```
! Copyright (C) 2006 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU General Public License.
! See the file COPYING for license details.

!BOP
! !MODULE: modmain
! !DESCRIPTION:
!   Contains all the global variables required by the spacegroup code.
!
! !REVISION HISTORY:
!   Created October 2006 (JKD)
!EOP
!BOC
module modmain

!-------------------------------!
!     space group variables     !
!-------------------------------!
! Hermann-Mauguin symbol
character(20) hrmg
! space-group number
character(20) num
! Schoenflies symbol
character(20) schn
! Hall symbol
character(20) hall

!----------------------------!
!     lattice parameters     !
!----------------------------!
! number of unit cells
integer ncell(3)
! lattice vector lengths
real(8) a,b,c
! lattice vector angles
real(8) ab,ac,bc
! lattice vectors stored column-wise
real(8) avec(3,3)
! inverse of lattice vector matrix
real(8) ainv(3,3)
! any vector with length less than epslat is considered zero
real(8), parameter :: epslat=1.d-6

!-------------------------!
!     atomic variables    !
!-------------------------!
! maximum allowed species
integer, parameter :: maxspecies=20
! maximum allowed atoms per species
integer, parameter :: maxatoms=200000
! number of species
integer nspecies
! number of atoms for each species
integer natoms(maxspecies)
! total number of atoms
integer natmtot
! primcell is .true. if primitive unit cell is to be found automatically
logical primcell
! maximum allowed Wyckoff positions
integer, parameter :: maxwpos=100
! number of Wyckoff positions
integer nwpos(maxspecies)
! Wyckoff positions
real(8) wpos(3,maxwpos,maxspecies)
! atomic positions in lattice coordinates
real(8) atposl(3,maxatoms,maxspecies)
! atomic positions in Cartesian coordinates
real(8) atposc(3,maxatoms,maxspecies)
! magnetic fields
```



```
real(8) bfcmt0(3,maxatoms,maxspecies)

!-----------------------------------!
!     atomic species variables      !
!-----------------------------------!
! species symbol
character(256) spsymb(maxspecies)

!-------------------------------!
!     numerical constants       !
!-------------------------------!
real(8), parameter :: pi=3.14159265358979323850

!-----------------------------------!
!     miscellaneous variables       !
!-----------------------------------!
! code version
integer version(3)
data version / 1,2,0 /

end module
!EOC


! Copyright (C) 2006 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU General Public License.
! See the file COPYING for license details.

!BOP
! !ROUTINE: sgsymb
! !INTERFACE:
subroutine sgsymb(hrmg,num,schn,hall)
! !INPUT/OUTPUT PARAMETERS:
!   hrmg : Hermann-Mauguin symbol (in,character(20))
!   num  : space group number (out,character(20))
!   schn : Schoenflies symbol (out,character(20))
!   hall : Hall symbol (out,character(20))
! !DESCRIPTION:
!   Returns the space group number, Schoenflies and Hall symbols given the
!   Hermann-Mauguin symbol. The routine is case-sensitive. With acknowledgements
!   to Ralf W. Grosse-Kunstleve and the tables available at
!   {\tt http://cci.lbl.gov/sginfo/}.
!
! !REVISION HISTORY:
!   Created October 2006 (JKD)
!EOP
!BOC
implicit none
! arguments
character(20), intent(in) :: hrmg
character(20), intent(out) :: num
character(20), intent(out) :: schn
character(20), intent(out) :: hall
select case(trim(adjustl(hrmg)))
case('P1')
  num='1'
  schn='C1^1'
  hall='P 1'
case('P-1')
  num='2'
  schn='Ci^1'
  hall='-P 1'
case('P2:b')
  num='3:b'
  schn='C2^1'
  hall='P 2y'
case('P2:c')
  num='3:c'
  schn='C2^1'
```

```
  hall='P 2'
case('P2:a')
  num='3:a'
  schn='C2^1'
  hall='P 2x'
case('P21:b')
  num='4:b'
  schn='C2^2'
  hall='P 2yb'
case('P21:c')
  num='4:c'
  schn='C2^2'
  hall='P 2c'
case('P21:a')
  num='4:a'
  schn='C2^2'
  hall='P 2xa'
case('C2:b1')
  num='5:b1'
  schn='C2^3'
  hall='C 2y'
case('C2:b2')
  num='5:b2'
  schn='C2^3'
  hall='A 2y'
case('C2:b3')
  num='5:b3'
  schn='C2^3'
  hall='I 2y'
case('C2:c1')
  num='5:c1'
  schn='C2^3'
  hall='A 2'
case('C2:c2')
  num='5:c2'
  schn='C2^3'
  hall='B 2'
case('C2:c3')
  num='5:c3'
  schn='C2^3'
  hall='I 2'
case('C2:a1')
  num='5:a1'
  schn='C2^3'
  hall='B 2x'
case('C2:a2')
  num='5:a2'
  schn='C2^3'
  hall='C 2x'
case('C2:a3')
  num='5:a3'
  schn='C2^3'
  hall='I 2x'
case('Pm:b')
  num='6:b'
  schn='Cs^1'
  hall='P -2y'
case('Pm:c')
  num='6:c'
  schn='Cs^1'
  hall='P -2'
case('Pm:a')
  num='6:a'
  schn='Cs^1'
  hall='P -2x'
case('Pc:b1')
  num='7:b1'
  schn='Cs^2'
  hall='P -2yc'
```



```
case('Pc:b2')
  num='7:b2'
  schn='Cs^2'
  hall='P -2yac'
case('Pc:b3')
  num='7:b3'
  schn='Cs^2'
  hall='P -2ya'
case('Pc:c1')
  num='7:c1'
  schn='Cs^2'
  hall='P -2a'
case('Pc:c2')
  num='7:c2'
  schn='Cs^2'
  hall='P -2ab'
case('Pc:c3')
  num='7:c3'
  schn='Cs^2'
  hall='P -2b'
case('Pc:a1')
  num='7:a1'
  schn='Cs^2'
  hall='P -2xb'
case('Pc:a2')
  num='7:a2'
  schn='Cs^2'
  hall='P -2xbc'
case('Pc:a3')
  num='7:a3'
  schn='Cs^2'
  hall='P -2xc'
case('Cm:b1')
  num='8:b1'
  schn='Cs^3'
  hall='C -2y'
case('Cm:b2')
  num='8:b2'
  schn='Cs^3'
  hall='A -2y'
case('Cm:b3')
  num='8:b3'
  schn='Cs^3'
  hall='I -2y'
case('Cm:c1')
  num='8:c1'
  schn='Cs^3'
  hall='A -2'
case('Cm:c2')
  num='8:c2'
  schn='Cs^3'
  hall='B -2'
case('Cm:c3')
  num='8:c3'
  schn='Cs^3'
  hall='I -2'
case('Cm:a1')
  num='8:a1'
  schn='Cs^3'
  hall='B -2x'
case('Cm:a2')
  num='8:a2'
  schn='Cs^3'
  hall='C -2x'
case('Cm:a3')
  num='8:a3'
  schn='Cs^3'
  hall='I -2x'
case('Cc:b1')
```

```
  num='9:b1'
  schn='Cs^4'
  hall='C -2yc'
case('Cc:b2')
  num='9:b2'
  schn='Cs^4'
  hall='A -2yac'
case('Cc:b3')
  num='9:b3'
  schn='Cs^4'
  hall='I -2ya'
case('Cc:-b1')
  num='9:-b1'
  schn='Cs^4'
  hall='A -2ya'
case('Cc:-b2')
  num='9:-b2'
  schn='Cs^4'
  hall='C -2ybc'
case('Cc:-b3')
  num='9:-b3'
  schn='Cs^4'
  hall='I -2yc'
case('Cc:c1')
  num='9:c1'
  schn='Cs^4'
  hall='A -2a'
case('Cc:c2')
  num='9:c2'
  schn='Cs^4'
  hall='B -2bc'
case('Cc:c3')
  num='9:c3'
  schn='Cs^4'
  hall='I -2b'
case('Cc:-c1')
  num='9:-c1'
  schn='Cs^4'
  hall='B -2b'
case('Cc:-c2')
  num='9:-c2'
  schn='Cs^4'
  hall='A -2ac'
case('Cc:-c3')
  num='9:-c3'
  schn='Cs^4'
  hall='I -2a'
case('Cc:a1')
  num='9:a1'
  schn='Cs^4'
  hall='B -2xb'
case('Cc:a2')
  num='9:a2'
  schn='Cs^4'
  hall='C -2xbc'
case('Cc:a3')
  num='9:a3'
  schn='Cs^4'
  hall='I -2xc'
case('Cc:-a1')
  num='9:-a1'
  schn='Cs^4'
  hall='C -2xc'
case('Cc:-a2')
  num='9:-a2'
  schn='Cs^4'
  hall='B -2xbc'
case('Cc:-a3')
  num='9:-a3'
```



```
  schn='Cs^4'
  hall='I -2xb'
case('P2/m:b')
  num='10:b'
  schn='C2h^1'
  hall='-P 2y'
case('P2/m:c')
  num='10:c'
  schn='C2h^1'
  hall='-P 2'
case('P2/m:a')
  num='10:a'
  schn='C2h^1'
  hall='-P 2x'
case('P21/m:b')
  num='11:b'
  schn='C2h^2'
  hall='-P 2yb'
case('P21/m:c')
  num='11:c'
  schn='C2h^2'
  hall='-P 2c'
case('P21/m:a')
  num='11:a'
  schn='C2h^2'
  hall='-P 2xa'
case('C2/m:b1')
  num='12:b1'
  schn='C2h^3'
  hall='-C 2y'
case('C2/m:b2')
  num='12:b2'
  schn='C2h^3'
  hall='-A 2y'
case('C2/m:b3')
  num='12:b3'
  schn='C2h^3'
  hall='-I 2y'
case('C2/m:c1')
  num='12:c1'
  schn='C2h^3'
  hall='-A 2'
case('C2/m:c2')
  num='12:c2'
  schn='C2h^3'
  hall='-B 2'
case('C2/m:c3')
  num='12:c3'
  schn='C2h^3'
  hall='-I 2'
case('C2/m:a1')
  num='12:a1'
  schn='C2h^3'
  hall='-B 2x'
case('C2/m:a2')
  num='12:a2'
  schn='C2h^3'
  hall='-C 2x'
case('C2/m:a3')
  num='12:a3'
  schn='C2h^3'
  hall='-I 2x'
case('P2/c:b1')
  num='13:b1'
  schn='C2h^4'
  hall='-P 2yc'
case('P2/c:b2')
  num='13:b2'
  schn='C2h^4'
```

```
  hall='-P 2yac'
case('P2/c:b3')
  num='13:b3'
  schn='C2h^4'
  hall='-P 2ya'
case('P2/c:c1')
  num='13:c1'
  schn='C2h^4'
  hall='-P 2a'
case('P2/c:c2')
  num='13:c2'
  schn='C2h^4'
  hall='-P 2ab'
case('P2/c:c3')
  num='13:c3'
  schn='C2h^4'
  hall='-P 2b'
case('P2/c:a1')
  num='13:a1'
  schn='C2h^4'
  hall='-P 2xb'
case('P2/c:a2')
  num='13:a2'
  schn='C2h^4'
  hall='-P 2xbc'
case('P2/c:a3')
  num='13:a3'
  schn='C2h^4'
  hall='-P 2xc'
case('P21/c:b1')
  num='14:b1'
  schn='C2h^5'
  hall='-P 2ybc'
case('P21/c:b2')
  num='14:b2'
  schn='C2h^5'
  hall='-P 2yn'
case('P21/c:b3')
  num='14:b3'
  schn='C2h^5'
  hall='-P 2yab'
case('P21/c:c1')
  num='14:c1'
  schn='C2h^5'
  hall='-P 2ac'
case('P21/c:c2')
  num='14:c2'
  schn='C2h^5'
  hall='-P 2n'
case('P21/c:c3')
  num='14:c3'
  schn='C2h^5'
  hall='-P 2bc'
case('P21/c:a1')
  num='14:a1'
  schn='C2h^5'
  hall='-P 2xab'
case('P21/c:a2')
  num='14:a2'
  schn='C2h^5'
  hall='-P 2xn'
case('P21/c:a3')
  num='14:a3'
  schn='C2h^5'
  hall='-P 2xac'
case('C2/c:b1')
  num='15:b1'
  schn='C2h^6'
  hall='-C 2yc'
```



```
case('C2/c:b2')
  num='15:b2'
  schn='C2h^6'
  hall='-A 2yac'
case('C2/c:b3')
  num='15:b3'
  schn='C2h^6'
  hall='-I 2ya'
case('C2/c:-b1')
  num='15:-b1'
  schn='C2h^6'
  hall='-A 2ya'
case('C2/c:-b2')
  num='15:-b2'
  schn='C2h^6'
  hall='-C 2ybc'
case('C2/c:-b3')
  num='15:-b3'
  schn='C2h^6'
  hall='-I 2yc'
case('C2/c:c1')
  num='15:c1'
  schn='C2h^6'
  hall='-A 2a'
case('C2/c:c2')
  num='15:c2'
  schn='C2h^6'
  hall='-B 2bc'
case('C2/c:c3')
  num='15:c3'
  schn='C2h^6'
  hall='-I 2b'
case('C2/c:-c1')
  num='15:-c1'
  schn='C2h^6'
  hall='-B 2b'
case('C2/c:-c2')
  num='15:-c2'
  schn='C2h^6'
  hall='-A 2ac'
case('C2/c:-c3')
  num='15:-c3'
  schn='C2h^6'
  hall='-I 2a'
case('C2/c:a1')
  num='15:a1'
  schn='C2h^6'
  hall='-B 2xb'
case('C2/c:a2')
  num='15:a2'
  schn='C2h^6'
  hall='-C 2xbc'
case('C2/c:a3')
  num='15:a3'
  schn='C2h^6'
  hall='-I 2xc'
case('C2/c:-a1')
  num='15:-a1'
  schn='C2h^6'
  hall='-C 2xc'
case('C2/c:-a2')
  num='15:-a2'
  schn='C2h^6'
  hall='-B 2xbc'
case('C2/c:-a3')
  num='15:-a3'
  schn='C2h^6'
  hall='-I 2xb'
case('P222')
```

```
  num='16'
  schn='D2^1'
  hall='P 2 2'
case('P2221')
  num='17'
  schn='D2^2'
  hall='P 2c 2'
case('P2122')
  num='17:cab'
  schn='D2^2'
  hall='P 2a 2a'
case('P2212')
  num='17:bca'
  schn='D2^2'
  hall='P 2 2b'
case('P21212')
  num='18'
  schn='D2^3'
  hall='P 2 2ab'
case('P22121')
  num='18:cab'
  schn='D2^3'
  hall='P 2bc 2'
case('P21221')
  num='18:bca'
  schn='D2^3'
  hall='P 2ac 2ac'
case('P212121')
  num='19'
  schn='D2^4'
  hall='P 2ac 2ab'
case('C2221')
  num='20'
  schn='D2^5'
  hall='C 2c 2'
case('A2122')
  num='20:cab'
  schn='D2^5'
  hall='A 2a 2a'
case('B2212')
  num='20:bca'
  schn='D2^5'
  hall='B 2 2b'
case('C222')
  num='21'
  schn='D2^6'
  hall='C 2 2'
case('A222')
  num='21:cab'
  schn='D2^6'
  hall='A 2 2'
case('B222')
  num='21:bca'
  schn='D2^6'
  hall='B 2 2'
case('F222')
  num='22'
  schn='D2^7'
  hall='F 2 2'
case('I222')
  num='23'
  schn='D2^8'
  hall='I 2 2'
case('I212121')
  num='24'
  schn='D2^9'
  hall='I 2b 2c'
case('Pmm2')
  num='25'
```



```
  schn='C2v^1'
  hall='P 2 -2'
case('P2mm')
  num='25:cab'
  schn='C2v^1'
  hall='P -2 2'
case('Pm2m')
  num='25:bca'
  schn='C2v^1'
  hall='P -2 -2'
case('Pmc21')
  num='26'
  schn='C2v^2'
  hall='P 2c -2'
case('Pcm21')
  num='26:ba-c'
  schn='C2v^2'
  hall='P 2c -2c'
case('P21ma')
  num='26:cab'
  schn='C2v^2'
  hall='P -2a 2a'
case('P21am')
  num='26:-cba'
  schn='C2v^2'
  hall='P -2 2a'
case('Pb21m')
  num='26:bca'
  schn='C2v^2'
  hall='P -2 -2b'
case('Pm21b')
  num='26:a-cb'
  schn='C2v^2'
  hall='P -2b -2'
case('Pcc2')
  num='27'
  schn='C2v^3'
  hall='P 2 -2c'
case('P2aa')
  num='27:cab'
  schn='C2v^3'
  hall='P -2a 2'
case('Pb2b')
  num='27:bca'
  schn='C2v^3'
  hall='P -2b -2b'
case('Pma2')
  num='28'
  schn='C2v^4'
  hall='P 2 -2a'
case('Pbm2')
  num='28:ba-c'
  schn='C2v^4'
  hall='P 2 -2b'
case('P2mb')
  num='28:cab'
  schn='C2v^4'
  hall='P -2b 2'
case('P2cm')
  num='28:-cba'
  schn='C2v^4'
  hall='P -2c 2'
case('Pc2m')
  num='28:bca'
  schn='C2v^4'
  hall='P -2c -2c'
case('Pm2a')
  num='28:a-cb'
  schn='C2v^4'
```

```
  hall='P -2a -2a'
case('Pca21')
  num='29'
  schn='C2v^5'
  hall='P 2c -2ac'
case('Pbc21')
  num='29:ba-c'
  schn='C2v^5'
  hall='P 2c -2b'
case('P21ab')
  num='29:cab'
  schn='C2v^5'
  hall='P -2b 2a'
case('P21ca')
  num='29:-cba'
  schn='C2v^5'
  hall='P -2ac 2a'
case('Pc21b')
  num='29:bca'
  schn='C2v^5'
  hall='P -2bc -2c'
case('Pb21a')
  num='29:a-cb'
  schn='C2v^5'
  hall='P -2a -2ab'
case('Pnc2')
  num='30'
  schn='C2v^6'
  hall='P 2 -2bc'
case('Pcn2')
  num='30:ba-c'
  schn='C2v^6'
  hall='P 2 -2ac'
case('P2na')
  num='30:cab'
  schn='C2v^6'
  hall='P -2ac 2'
case('P2an')
  num='30:-cba'
  schn='C2v^6'
  hall='P -2ab 2'
case('Pb2n')
  num='30:bca'
  schn='C2v^6'
  hall='P -2ab -2ab'
case('Pn2b')
  num='30:a-cb'
  schn='C2v^6'
  hall='P -2bc -2bc'
case('Pmn21')
  num='31'
  schn='C2v^7'
  hall='P 2ac -2'
case('Pnm21')
  num='31:ba-c'
  schn='C2v^7'
  hall='P 2bc -2bc'
case('P21mn')
  num='31:cab'
  schn='C2v^7'
  hall='P -2ab 2ab'
case('P21nm')
  num='31:-cba'
  schn='C2v^7'
  hall='P -2 2ac'
case('Pn21m')
  num='31:bca'
  schn='C2v^7'
  hall='P -2 -2bc'
```



```
case('Pm21n')
  num='31:a-cb'
  schn='C2v^7'
  hall='P -2ab -2'
case('Pba2')
  num='32'
  schn='C2v^8'
  hall='P 2 -2ab'
case('P2cb')
  num='32:cab'
  schn='C2v^8'
  hall='P -2bc 2'
case('Pc2a')
  num='32:bca'
  schn='C2v^8'
  hall='P -2ac -2ac'
case('Pna21')
  num='33'
  schn='C2v^9'
  hall='P 2c -2n'
case('Pbn21')
  num='33:ba-c'
  schn='C2v^9'
  hall='P 2c -2ab'
case('P21nb')
  num='33:cab'
  schn='C2v^9'
  hall='P -2bc 2a'
case('P21cn')
  num='33:-cba'
  schn='C2v^9'
  hall='P -2n 2a'
case('Pc21n')
  num='33:bca'
  schn='C2v^9'
  hall='P -2n -2ac'
case('Pn21a')
  num='33:a-cb'
  schn='C2v^9'
  hall='P -2ac -2n'
case('Pnn2')
  num='34'
  schn='C2v^10'
  hall='P 2 -2n'
case('P2nn')
  num='34:cab'
  schn='C2v^10'
  hall='P -2n 2'
case('Pn2n')
  num='34:bca'
  schn='C2v^10'
  hall='P -2n -2n'
case('Cmm2')
  num='35'
  schn='C2v^11'
  hall='C 2 -2'
case('A2mm')
  num='35:cab'
  schn='C2v^11'
  hall='A -2 2'
case('Bm2m')
  num='35:bca'
  schn='C2v^11'
  hall='B -2 -2'
case('Cmc21')
  num='36'
  schn='C2v^12'
  hall='C 2c -2'
case('Ccm21')
```

```
  num='36:ba-c'
  schn='C2v^12'
  hall='C 2c -2c'
case('A21ma')
  num='36:cab'
  schn='C2v^12'
  hall='A -2a 2a'
case('A21am')
  num='36:-cba'
  schn='C2v^12'
  hall='A -2 2a'
case('Bb21m')
  num='36:bca'
  schn='C2v^12'
  hall='B -2 -2b'
case('Bm21b')
  num='36:a-cb'
  schn='C2v^12'
  hall='B -2b -2'
case('Ccc2')
  num='37'
  schn='C2v^13'
  hall='C 2 -2c'
case('A2aa')
  num='37:cab'
  schn='C2v^13'
  hall='A -2a 2'
case('Bb2b')
  num='37:bca'
  schn='C2v^13'
  hall='B -2b -2b'
case('Amm2')
  num='38'
  schn='C2v^14'
  hall='A 2 -2'
case('Bmm2')
  num='38:ba-c'
  schn='C2v^14'
  hall='B 2 -2'
case('B2mm')
  num='38:cab'
  schn='C2v^14'
  hall='B -2 2'
case('C2mm')
  num='38:-cba'
  schn='C2v^14'
  hall='C -2 2'
case('Cm2m')
  num='38:bca'
  schn='C2v^14'
  hall='C -2 -2'
case('Am2m')
  num='38:a-cb'
  schn='C2v^14'
  hall='A -2 -2'
case('Abm2')
  num='39'
  schn='C2v^15'
  hall='A 2 -2c'
case('Bma2')
  num='39:ba-c'
  schn='C2v^15'
  hall='B 2 -2c'
case('B2cm')
  num='39:cab'
  schn='C2v^15'
  hall='B -2c 2'
case('C2mb')
  num='39:-cba'
```



```
  schn='C2v^15'
  hall='C -2b 2'
case('Cm2a')
  num='39:bca'
  schn='C2v^15'
  hall='C -2b -2b'
case('Ac2m')
  num='39:a-cb'
  schn='C2v^15'
  hall='A -2c -2c'
case('Ama2')
  num='40'
  schn='C2v^16'
  hall='A 2 -2a'
case('Bbm2')
  num='40:ba-c'
  schn='C2v^16'
  hall='B 2 -2b'
case('B2mb')
  num='40:cab'
  schn='C2v^16'
  hall='B -2b 2'
case('C2cm')
  num='40:-cba'
  schn='C2v^16'
  hall='C -2c 2'
case('Cc2m')
  num='40:bca'
  schn='C2v^16'
  hall='C -2c -2c'
case('Am2a')
  num='40:a-cb'
  schn='C2v^16'
  hall='A -2a -2a'
case('Aba2')
  num='41'
  schn='C2v^17'
  hall='A 2 -2ac'
case('Bba2')
  num='41:ba-c'
  schn='C2v^17'
  hall='B 2 -2bc'
case('B2cb')
  num='41:cab'
  schn='C2v^17'
  hall='B -2bc 2'
case('C2cb')
  num='41:-cba'
  schn='C2v^17'
  hall='C -2bc 2'
case('Cc2a')
  num='41:bca'
  schn='C2v^17'
  hall='C -2bc -2bc'
case('Ac2a')
  num='41:a-cb'
  schn='C2v^17'
  hall='A -2ac -2ac'
case('Fmm2')
  num='42'
  schn='C2v^18'
  hall='F 2 -2'
case('F2mm')
  num='42:cab'
  schn='C2v^18'
  hall='F -2 2'
case('Fm2m')
  num='42:bca'
  schn='C2v^18'
```

```
  hall='F -2 -2'
case('Fdd2')
  num='43'
  schn='C2v^19'
  hall='F 2 -2d'
case('F2dd')
  num='43:cab'
  schn='C2v^19'
  hall='F -2d 2'
case('Fd2d')
  num='43:bca'
  schn='C2v^19'
  hall='F -2d -2d'
case('Imm2')
  num='44'
  schn='C2v^20'
  hall='I 2 -2'
case('I2mm')
  num='44:cab'
  schn='C2v^20'
  hall='I -2 2'
case('Im2m')
  num='44:bca'
  schn='C2v^20'
  hall='I -2 -2'
case('Iba2')
  num='45'
  schn='C2v^21'
  hall='I 2 -2c'
case('I2cb')
  num='45:cab'
  schn='C2v^21'
  hall='I -2a 2'
case('Ic2a')
  num='45:bca'
  schn='C2v^21'
  hall='I -2b -2b'
case('Ima2')
  num='46'
  schn='C2v^22'
  hall='I 2 -2a'
case('Ibm2')
  num='46:ba-c'
  schn='C2v^22'
  hall='I 2 -2b'
case('I2mb')
  num='46:cab'
  schn='C2v^22'
  hall='I -2b 2'
case('I2cm')
  num='46:-cba'
  schn='C2v^22'
  hall='I -2c 2'
case('Ic2m')
  num='46:bca'
  schn='C2v^22'
  hall='I -2c -2c'
case('Im2a')
  num='46:a-cb'
  schn='C2v^22'
  hall='I -2a -2a'
case('Pmmm')
  num='47'
  schn='D2h^1'
  hall='-P 2 2'
case('Pnnn:1')
  num='48:1'
  schn='D2h^2'
  hall='P 2 2 -1n'
```



```
case('Pnnn:2')
  num='48:2'
  schn='D2h^2'
  hall='-P 2ab 2bc'
case('Pccm')
  num='49'
  schn='D2h^3'
  hall='-P 2 2c'
case('Pmaa')
  num='49:cab'
  schn='D2h^3'
  hall='-P 2a 2'
case('Pbmb')
  num='49:bca'
  schn='D2h^3'
  hall='-P 2b 2b'
case('Pban:1')
  num='50:1'
  schn='D2h^4'
  hall='P 2 2 -1ab'
case('Pban:2')
  num='50:2'
  schn='D2h^4'
  hall='-P 2ab 2b'
case('Pncb:1')
  num='50:1cab'
  schn='D2h^4'
  hall='P 2 2 -1bc'
case('Pncb:2')
  num='50:2cab'
  schn='D2h^4'
  hall='-P 2b 2bc'
case('Pcna:1')
  num='50:1bca'
  schn='D2h^4'
  hall='P 2 2 -1ac'
case('Pcna:2')
  num='50:2bca'
  schn='D2h^4'
  hall='-P 2a 2c'
case('Pmma')
  num='51'
  schn='D2h^5'
  hall='-P 2a 2a'
case('Pmmb')
  num='51:ba-c'
  schn='D2h^5'
  hall='-P 2b 2'
case('Pbmm')
  num='51:cab'
  schn='D2h^5'
  hall='-P 2 2b'
case('Pcmm')
  num='51:-cba'
  schn='D2h^5'
  hall='-P 2c 2c'
case('Pmcm')
  num='51:bca'
  schn='D2h^5'
  hall='-P 2c 2'
case('Pmam')
  num='51:a-cb'
  schn='D2h^5'
  hall='-P 2 2a'
case('Pnna')
  num='52'
  schn='D2h^6'
  hall='-P 2a 2bc'
case('Pnnb')
```

```
  num='52:ba-c'
  schn='D2h^6'
  hall='-P 2b 2n'
case('Pbnn')
  num='52:cab'
  schn='D2h^6'
  hall='-P 2n 2b'
case('Pcnn')
  num='52:-cba'
  schn='D2h^6'
  hall='-P 2ab 2c'
case('Pncn')
  num='52:bca'
  schn='D2h^6'
  hall='-P 2ab 2n'
case('Pnan')
  num='52:a-cb'
  schn='D2h^6'
  hall='-P 2n 2bc'
case('Pmna')
  num='53'
  schn='D2h^7'
  hall='-P 2ac 2'
case('Pnmb')
  num='53:ba-c'
  schn='D2h^7'
  hall='-P 2bc 2bc'
case('Pbmn')
  num='53:cab'
  schn='D2h^7'
  hall='-P 2ab 2ab'
case('Pcnm')
  num='53:-cba'
  schn='D2h^7'
  hall='-P 2 2ac'
case('Pncm')
  num='53:bca'
  schn='D2h^7'
  hall='-P 2 2bc'
case('Pman')
  num='53:a-cb'
  schn='D2h^7'
  hall='-P 2ab 2'
case('Pcca')
  num='54'
  schn='D2h^8'
  hall='-P 2a 2ac'
case('Pccb')
  num='54:ba-c'
  schn='D2h^8'
  hall='-P 2b 2c'
case('Pbaa')
  num='54:cab'
  schn='D2h^8'
  hall='-P 2a 2b'
case('Pcaa')
  num='54:-cba'
  schn='D2h^8'
  hall='-P 2ac 2c'
case('Pbcb')
  num='54:bca'
  schn='D2h^8'
  hall='-P 2bc 2b'
case('Pbab')
  num='54:a-cb'
  schn='D2h^8'
  hall='-P 2b 2ab'
case('Pbam')
  num='55'
```



```
  schn='D2h^9'
  hall='-P 2 2ab'
case('Pmcb')
  num='55:cab'
  schn='D2h^9'
  hall='-P 2bc 2'
case('Pcma')
  num='55:bca'
  schn='D2h^9'
  hall='-P 2ac 2ac'
case('Pccn')
  num='56'
  schn='D2h^10'
  hall='-P 2ab 2ac'
case('Pnaa')
  num='56:cab'
  schn='D2h^10'
  hall='-P 2ac 2bc'
case('Pbnb')
  num='56:bca'
  schn='D2h^10'
  hall='-P 2bc 2ab'
case('Pbcm')
  num='57'
  schn='D2h^11'
  hall='-P 2c 2b'
case('Pcam')
  num='57:ba-c'
  schn='D2h^11'
  hall='-P 2c 2ac'
case('Pmca')
  num='57:cab'
  schn='D2h^11'
  hall='-P 2ac 2a'
case('Pmab')
  num='57:-cba'
  schn='D2h^11'
  hall='-P 2b 2a'
case('Pbma')
  num='57:bca'
  schn='D2h^11'
  hall='-P 2a 2ab'
case('Pcmb')
  num='57:a-cb'
  schn='D2h^11'
  hall='-P 2bc 2c'
case('Pnnm')
  num='58'
  schn='D2h^12'
  hall='-P 2 2n'
case('Pmnn')
  num='58:cab'
  schn='D2h^12'
  hall='-P 2n 2'
case('Pnmn')
  num='58:bca'
  schn='D2h^12'
  hall='-P 2n 2n'
case('Pmmn:1')
  num='59:1'
  schn='D2h^13'
  hall='P 2 2ab -1ab'
case('Pmmn:2')
  num='59:2'
  schn='D2h^13'
  hall='-P 2ab 2a'
case('Pnmm:1')
  num='59:1cab'
  schn='D2h^13'
```

```
  hall='P 2bc 2 -1bc'
case('Pnmm:2')
  num='59:2cab'
  schn='D2h^13'
  hall='-P 2c 2bc'
case('Pmnm:1')
  num='59:1bca'
  schn='D2h^13'
  hall='P 2ac 2ac -1ac'
case('Pmnm:2')
  num='59:2bca'
  schn='D2h^13'
  hall='-P 2c 2a'
case('Pbcn')
  num='60'
  schn='D2h^14'
  hall='-P 2n 2ab'
case('Pcan')
  num='60:ba-c'
  schn='D2h^14'
  hall='-P 2n 2c'
case('Pnca')
  num='60:cab'
  schn='D2h^14'
  hall='-P 2a 2n'
case('Pnab')
  num='60:-cba'
  schn='D2h^14'
  hall='-P 2bc 2n'
case('Pbna')
  num='60:bca'
  schn='D2h^14'
  hall='-P 2ac 2b'
case('Pcnb')
  num='60:a-cb'
  schn='D2h^14'
  hall='-P 2b 2ac'
case('Pbca')
  num='61'
  schn='D2h^15'
  hall='-P 2ac 2ab'
case('Pcab')
  num='61:ba-c'
  schn='D2h^15'
  hall='-P 2bc 2ac'
case('Pnma')
  num='62'
  schn='D2h^16'
  hall='-P 2ac 2n'
case('Pmnb')
  num='62:ba-c'
  schn='D2h^16'
  hall='-P 2bc 2a'
case('Pbnm')
  num='62:cab'
  schn='D2h^16'
  hall='-P 2c 2ab'
case('Pcmn')
  num='62:-cba'
  schn='D2h^16'
  hall='-P 2n 2ac'
case('Pmcn')
  num='62:bca'
  schn='D2h^16'
  hall='-P 2n 2a'
case('Pnam')
  num='62:a-cb'
  schn='D2h^16'
  hall='-P 2c 2n'
```



```
case('Cmcm')
  num='63'
  schn='D2h^17'
  hall='-C 2c 2'
case('Ccmm')
  num='63:ba-c'
  schn='D2h^17'
  hall='-C 2c 2c'
case('Amma')
  num='63:cab'
  schn='D2h^17'
  hall='-A 2a 2a'
case('Amam')
  num='63:-cba'
  schn='D2h^17'
  hall='-A 2 2a'
case('Bbmm')
  num='63:bca'
  schn='D2h^17'
  hall='-B 2 2b'
case('Bmmb')
  num='63:a-cb'
  schn='D2h^17'
  hall='-B 2b 2'
case('Cmca')
  num='64'
  schn='D2h^18'
  hall='-C 2bc 2'
case('Ccmb')
  num='64:ba-c'
  schn='D2h^18'
  hall='-C 2bc 2bc'
case('Abma')
  num='64:cab'
  schn='D2h^18'
  hall='-A 2ac 2ac'
case('Acam')
  num='64:-cba'
  schn='D2h^18'
  hall='-A 2 2ac'
case('Bbcm')
  num='64:bca'
  schn='D2h^18'
  hall='-B 2 2bc'
case('Bmab')
  num='64:a-cb'
  schn='D2h^18'
  hall='-B 2bc 2'
case('Cmmm')
  num='65'
  schn='D2h^19'
  hall='-C 2 2'
case('Ammm')
  num='65:cab'
  schn='D2h^19'
  hall='-A 2 2'
case('Bmmm')
  num='65:bca'
  schn='D2h^19'
  hall='-B 2 2'
case('Cccm')
  num='66'
  schn='D2h^20'
  hall='-C 2 2c'
case('Amaa')
  num='66:cab'
  schn='D2h^20'
  hall='-A 2a 2'
case('Bbmb')
```

```
  num='66:bca'
  schn='D2h^20'
  hall='-B 2b 2b'
case('Cmma')
  num='67'
  schn='D2h^21'
  hall='-C 2b 2'
case('Cmmb')
  num='67:ba-c'
  schn='D2h^21'
  hall='-C 2b 2b'
case('Abmm')
  num='67:cab'
  schn='D2h^21'
  hall='-A 2c 2c'
case('Acmm')
  num='67:-cba'
  schn='D2h^21'
  hall='-A 2 2c'
case('Bmcm')
  num='67:bca'
  schn='D2h^21'
  hall='-B 2 2c'
case('Bmam')
  num='67:a-cb'
  schn='D2h^21'
  hall='-B 2c 2'
case('Ccca:1')
  num='68:1'
  schn='D2h^22'
  hall='C 2 2 -1bc'
case('Ccca:2')
  num='68:2'
  schn='D2h^22'
  hall='-C 2b 2bc'
case('Cccb:1')
  num='68:1ba-c'
  schn='D2h^22'
  hall='C 2 2 -1bc'
case('Cccb:2')
  num='68:2ba-c'
  schn='D2h^22'
  hall='-C 2b 2c'
case('Abaa:1')
  num='68:1cab'
  schn='D2h^22'
  hall='A 2 2 -1ac'
case('Abaa:2')
  num='68:2cab'
  schn='D2h^22'
  hall='-A 2a 2c'
case('Acaa:1')
  num='68:1-cba'
  schn='D2h^22'
  hall='A 2 2 -1ac'
case('Acaa:2')
  num='68:2-cba'
  schn='D2h^22'
  hall='-A 2ac 2c'
case('Bbcb:1')
  num='68:1bca'
  schn='D2h^22'
  hall='B 2 2 -1bc'
case('Bbcb:2')
  num='68:2bca'
  schn='D2h^22'
  hall='-B 2bc 2b'
case('Bbab:1')
  num='68:1a-cb'
```



```
  schn='D2h^22'
  hall='B 2 2 -1bc'
case('Bbab:2')
  num='68:2a-cb'
  schn='D2h^22'
  hall='-B 2b 2bc'
case('Fmmm')
  num='69'
  schn='D2h^23'
  hall='-F 2 2'
case('Fddd:1')
  num='70:1'
  schn='D2h^24'
  hall='F 2 2 -1d'
case('Fddd:2')
  num='70:2'
  schn='D2h^24'
  hall='-F 2uv 2vw'
case('Immm')
  num='71'
  schn='D2h^25'
  hall='-I 2 2'
case('Ibam')
  num='72'
  schn='D2h^26'
  hall='-I 2 2c'
case('Imcb')
  num='72:cab'
  schn='D2h^26'
  hall='-I 2a 2'
case('Icma')
  num='72:bca'
  schn='D2h^26'
  hall='-I 2b 2b'
case('Ibca')
  num='73'
  schn='D2h^27'
  hall='-I 2b 2c'
case('Icab')
  num='73:ba-c'
  schn='D2h^27'
  hall='-I 2a 2b'
case('Imma')
  num='74'
  schn='D2h^28'
  hall='-I 2b 2'
case('Immb')
  num='74:ba-c'
  schn='D2h^28'
  hall='-I 2a 2a'
case('Ibmm')
  num='74:cab'
  schn='D2h^28'
  hall='-I 2c 2c'
case('Icmm')
  num='74:-cba'
  schn='D2h^28'
  hall='-I 2 2b'
case('Imcm')
  num='74:bca'
  schn='D2h^28'
  hall='-I 2 2a'
case('Imam')
  num='74:a-cb'
  schn='D2h^28'
  hall='-I 2c 2'
case('P4')
  num='75'
  schn='C4^1'
```

```
  hall='P 4'
case('P41')
  num='76'
  schn='C4^2'
  hall='P 4w'
case('P42')
  num='77'
  schn='C4^3'
  hall='P 4c'
case('P43')
  num='78'
  schn='C4^4'
  hall='P 4cw'
case('I4')
  num='79'
  schn='C4^5'
  hall='I 4'
case('I41')
  num='80'
  schn='C4^6'
  hall='I 4bw'
case('P-4')
  num='81'
  schn='S4^1'
  hall='P -4'
case('I-4')
  num='82'
  schn='S4^2'
  hall='I -4'
case('P4/m')
  num='83'
  schn='C4h^1'
  hall='-P 4'
case('P42/m')
  num='84'
  schn='C4h^2'
  hall='-P 4c'
case('P4/n:1')
  num='85:1'
  schn='C4h^3'
  hall='P 4ab -1ab'
case('P4/n:2')
  num='85:2'
  schn='C4h^3'
  hall='-P 4a'
case('P42/n:1')
  num='86:1'
  schn='C4h^4'
  hall='P 4n -1n'
case('P42/n:2')
  num='86:2'
  schn='C4h^4'
  hall='-P 4bc'
case('I4/m')
  num='87'
  schn='C4h^5'
  hall='-I 4'
case('I41/a:1')
  num='88:1'
  schn='C4h^6'
  hall='I 4bw -1bw'
case('I41/a:2')
  num='88:2'
  schn='C4h^6'
  hall='-I 4ad'
case('P422')
  num='89'
  schn='D4^1'
  hall='P 4 2'
```



```
case('P4212')
  num='90'
  schn='D4^2'
  hall='P 4ab 2ab'
case('P4122')
  num='91'
  schn='D4^3'
  hall='P 4w 2c'
case('P41212')
  num='92'
  schn='D4^4'
  hall='P 4abw 2nw'
case('P4222')
  num='93'
  schn='D4^5'
  hall='P 4c 2'
case('P42212')
  num='94'
  schn='D4^6'
  hall='P 4n 2n'
case('P4322')
  num='95'
  schn='D4^7'
  hall='P 4cw 2c'
case('P43212')
  num='96'
  schn='D4^8'
  hall='P 4nw 2abw'
case('I422')
  num='97'
  schn='D4^9'
  hall='I 4 2'
case('I4122')
  num='98'
  schn='D4^10'
  hall='I 4bw 2bw'
case('P4mm')
  num='99'
  schn='C4v^1'
  hall='P 4 -2'
case('P4bm')
  num='100'
  schn='C4v^2'
  hall='P 4 -2ab'
case('P42cm')
  num='101'
  schn='C4v^3'
  hall='P 4c -2c'
case('P42nm')
  num='102'
  schn='C4v^4'
  hall='P 4n -2n'
case('P4cc')
  num='103'
  schn='C4v^5'
  hall='P 4 -2c'
case('P4nc')
  num='104'
  schn='C4v^6'
  hall='P 4 -2n'
case('P42mc')
  num='105'
  schn='C4v^7'
  hall='P 4c -2'
case('P42bc')
  num='106'
  schn='C4v^8'
  hall='P 4c -2ab'
case('I4mm')
```

```
  num='107'
  schn='C4v^9'
  hall='I 4 -2'
case('I4cm')
  num='108'
  schn='C4v^10'
  hall='I 4 -2c'
case('I41md')
  num='109'
  schn='C4v^11'
  hall='I 4bw -2'
case('I41cd')
  num='110'
  schn='C4v^12'
  hall='I 4bw -2c'
case('P-42m')
  num='111'
  schn='D2d^1'
  hall='P -4 2'
case('P-42c')
  num='112'
  schn='D2d^2'
  hall='P -4 2c'
case('P-421m')
  num='113'
  schn='D2d^3'
  hall='P -4 2ab'
case('P-421c')
  num='114'
  schn='D2d^4'
  hall='P -4 2n'
case('P-4m2')
  num='115'
  schn='D2d^5'
  hall='P -4 -2'
case('P-4c2')
  num='116'
  schn='D2d^6'
  hall='P -4 -2c'
case('P-4b2')
  num='117'
  schn='D2d^7'
  hall='P -4 -2ab'
case('P-4n2')
  num='118'
  schn='D2d^8'
  hall='P -4 -2n'
case('I-4m2')
  num='119'
  schn='D2d^9'
  hall='I -4 -2'
case('I-4c2')
  num='120'
  schn='D2d^10'
  hall='I -4 -2c'
case('I-42m')
  num='121'
  schn='D2d^11'
  hall='I -4 2'
case('I-42d')
  num='122'
  schn='D2d^12'
  hall='I -4 2bw'
case('P4/mmm')
  num='123'
  schn='D4h^1'
  hall='-P 4 2'
case('P4/mcc')
  num='124'
```



```
  schn='D4h^2'
  hall='-P 4 2c'
case('P4/nbm:1')
  num='125:1'
  schn='D4h^3'
  hall='P 4 2 -1ab'
case('P4/nbm:2')
  num='125:2'
  schn='D4h^3'
  hall='-P 4a 2b'
case('P4/nnc:1')
  num='126:1'
  schn='D4h^4'
  hall='P 4 2 -1n'
case('P4/nnc:2')
  num='126:2'
  schn='D4h^4'
  hall='-P 4a 2bc'
case('P4/mbm')
  num='127'
  schn='D4h^5'
  hall='-P 4 2ab'
case('P4/mnc')
  num='128'
  schn='D4h^6'
  hall='-P 4 2n'
case('P4/nmm:1')
  num='129:1'
  schn='D4h^7'
  hall='P 4ab 2ab -1ab'
case('P4/nmm:2')
  num='129:2'
  schn='D4h^7'
  hall='-P 4a 2a'
case('P4/ncc:1')
  num='130:1'
  schn='D4h^8'
  hall='P 4ab 2n -1ab'
case('P4/ncc:2')
  num='130:2'
  schn='D4h^8'
  hall='-P 4a 2ac'
case('P42/mmc')
  num='131'
  schn='D4h^9'
  hall='-P 4c 2'
case('P42/mcm')
  num='132'
  schn='D4h^10'
  hall='-P 4c 2c'
case('P42/nbc:1')
  num='133:1'
  schn='D4h^11'
  hall='P 4n 2c -1n'
case('P42/nbc:2')
  num='133:2'
  schn='D4h^11'
  hall='-P 4ac 2b'
case('P42/nnm:1')
  num='134:1'
  schn='D4h^12'
  hall='P 4n 2 -1n'
case('P42/nnm:2')
  num='134:2'
  schn='D4h^12'
  hall='-P 4ac 2bc'
case('P42/mbc')
  num='135'
  schn='D4h^13'
```

```
  hall='-P 4c 2ab'
case('P42/mnm')
  num='136'
  schn='D4h^14'
  hall='-P 4n 2n'
case('P42/nmc:1')
  num='137:1'
  schn='D4h^15'
  hall='P 4n 2n -1n'
case('P42/nmc:2')
  num='137:2'
  schn='D4h^15'
  hall='-P 4ac 2a'
case('P42/ncm:1')
  num='138:1'
  schn='D4h^16'
  hall='P 4n 2ab -1n'
case('P42/ncm:2')
  num='138:2'
  schn='D4h^16'
  hall='-P 4ac 2ac'
case('I4/mmm')
  num='139'
  schn='D4h^17'
  hall='-I 4 2'
case('I4/mcm')
  num='140'
  schn='D4h^18'
  hall='-I 4 2c'
case('I41/amd:1')
  num='141:1'
  schn='D4h^19'
  hall='I 4bw 2bw -1bw'
case('I41/amd:2')
  num='141:2'
  schn='D4h^19'
  hall='-I 4bd 2'
case('I41/acd:1')
  num='142:1'
  schn='D4h^20'
  hall='I 4bw 2aw -1bw'
case('I41/acd:2')
  num='142:2'
  schn='D4h^20'
  hall='-I 4bd 2c'
case('P3')
  num='143'
  schn='C3^1'
  hall='P 3'
case('P31')
  num='144'
  schn='C3^2'
  hall='P 31'
case('P32')
  num='145'
  schn='C3^3'
  hall='P 32'
case('R3:H')
  num='146:H'
  schn='C3^4'
  hall='R 3'
case('R3:R')
  num='146:R'
  schn='C3^4'
  hall='P 3*'
case('P-3')
  num='147'
  schn='C3i^1'
  hall='-P 3'
```



```
case('R-3:H')
  num='148:H'
  schn='C3i^2'
  hall='-R 3'
case('R-3:R')
  num='148:R'
  schn='C3i^2'
  hall='-P 3*'
case('P312')
  num='149'
  schn='D3^1'
  hall='P 3 2'
case('P321')
  num='150'
  schn='D3^2'
  hall='P 3 2"'
case('P3112')
  num='151'
  schn='D3^3'
  hall='P 31 2c (0 0 1)'
case('P3121')
  num='152'
  schn='D3^4'
  hall='P 31 2"'
case('P3212')
  num='153'
  schn='D3^5'
  hall='P 32 2c (0 0 -1)'
case('P3221')
  num='154'
  schn='D3^6'
  hall='P 32 2"'
case('R32:H')
  num='155:H'
  schn='D3^7'
  hall='R 3 2"'
case('R32:R')
  num='155:R'
  schn='D3^7'
  hall='P 3* 2'
case('P3m1')
  num='156'
  schn='C3v^1'
  hall='P 3 -2"'
case('P31m')
  num='157'
  schn='C3v^2'
  hall='P 3 -2'
case('P3c1')
  num='158'
  schn='C3v^3'
  hall='P 3 -2"c'
case('P31c')
  num='159'
  schn='C3v^4'
  hall='P 3 -2c'
case('R3m:H')
  num='160:H'
  schn='C3v^5'
  hall='R 3 -2"'
case('R3m:R')
  num='160:R'
  schn='C3v^5'
  hall='P 3* -2'
case('R3c:H')
  num='161:H'
  schn='C3v^6'
  hall='R 3 -2"c'
case('R3c:R')
```

```
  num='161:R'
  schn='C3v^6'
  hall='P 3* -2n'
case('P-31m')
  num='162'
  schn='D3d^1'
  hall='-P 3 2'
case('P-31c')
  num='163'
  schn='D3d^2'
  hall='-P 3 2c'
case('P-3m1')
  num='164'
  schn='D3d^3'
  hall='-P 3 2"'
case('P-3c1')
  num='165'
  schn='D3d^4'
  hall='-P 3 2"c'
case('R-3m:H')
  num='166:H'
  schn='D3d^5'
  hall='-R 3 2"'
case('R-3m:R')
  num='166:R'
  schn='D3d^5'
  hall='-P 3* 2'
case('R-3c:H')
  num='167:H'
  schn='D3d^6'
  hall='-R 3 2"c'
case('R-3c:R')
  num='167:R'
  schn='D3d^6'
  hall='-P 3* 2n'
case('P6')
  num='168'
  schn='C6^1'
  hall='P 6'
case('P61')
  num='169'
  schn='C6^2'
  hall='P 61'
case('P65')
  num='170'
  schn='C6^3'
  hall='P 65'
case('P62')
  num='171'
  schn='C6^4'
  hall='P 62'
case('P64')
  num='172'
  schn='C6^5'
  hall='P 64'
case('P63')
  num='173'
  schn='C6^6'
  hall='P 6c'
case('P-6')
  num='174'
  schn='C3h^1'
  hall='P -6'
case('P6/m')
  num='175'
  schn='C6h^1'
  hall='-P 6'
case('P63/m')
  num='176'
```



```
  schn='C6h^2'
  hall='-P 6c'
case('P622')
  num='177'
  schn='D6^1'
  hall='P 6 2'
case('P6122')
  num='178'
  schn='D6^2'
  hall='P 61 2 (0 0 -1)'
case('P6522')
  num='179'
  schn='D6^3'
  hall='P 65 2 (0 0 1)'
case('P6222')
  num='180'
  schn='D6^4'
  hall='P 62 2c (0 0 1)'
case('P6422')
  num='181'
  schn='D6^5'
  hall='P 64 2c (0 0 -1)'
case('P6322')
  num='182'
  schn='D6^6'
  hall='P 6c 2c'
case('P6mm')
  num='183'
  schn='C6v^1'
  hall='P 6 -2'
case('P6cc')
  num='184'
  schn='C6v^2'
  hall='P 6 -2c'
case('P63cm')
  num='185'
  schn='C6v^3'
  hall='P 6c -2'
case('P63mc')
  num='186'
  schn='C6v^4'
  hall='P 6c -2c'
case('P-6m2')
  num='187'
  schn='D3h^1'
  hall='P -6 2'
case('P-6c2')
  num='188'
  schn='D3h^2'
  hall='P -6c 2'
case('P-62m')
  num='189'
  schn='D3h^3'
  hall='P -6 -2'
case('P-62c')
  num='190'
  schn='D3h^4'
  hall='P -6c -2c'
case('P6/mmm')
  num='191'
  schn='D6h^1'
  hall='-P 6 2'
case('P6/mcc')
  num='192'
  schn='D6h^2'
  hall='-P 6 2c'
case('P63/mcm')
  num='193'
  schn='D6h^3'
```

```
  hall='-P 6c 2'
case('P63/mmc')
  num='194'
  schn='D6h^4'
  hall='-P 6c 2c'
case('P23')
  num='195'
  schn='T^1'
  hall='P 2 2 3'
case('F23')
  num='196'
  schn='T^2'
  hall='F 2 2 3'
case('I23')
  num='197'
  schn='T^3'
  hall='I 2 2 3'
case('P213')
  num='198'
  schn='T^4'
  hall='P 2ac 2ab 3'
case('I213')
  num='199'
  schn='T^5'
  hall='I 2b 2c 3'
case('Pm-3')
  num='200'
  schn='Th^1'
  hall='-P 2 2 3'
case('Pn-3:1')
  num='201:1'
  schn='Th^2'
  hall='P 2 2 3 -1n'
case('Pn-3:2')
  num='201:2'
  schn='Th^2'
  hall='-P 2ab 2bc 3'
case('Fm-3')
  num='202'
  schn='Th^3'
  hall='-F 2 2 3'
case('Fd-3:1')
  num='203:1'
  schn='Th^4'
  hall='F 2 2 3 -1d'
case('Fd-3:2')
  num='203:2'
  schn='Th^4'
  hall='-F 2uv 2vw 3'
case('Im-3')
  num='204'
  schn='Th^5'
  hall='-I 2 2 3'
case('Pa-3')
  num='205'
  schn='Th^6'
  hall='-P 2ac 2ab 3'
case('Ia-3')
  num='206'
  schn='Th^7'
  hall='-I 2b 2c 3'
case('P432')
  num='207'
  schn='O^1'
  hall='P 4 2 3'
case('P4232')
  num='208'
  schn='O^2'
  hall='P 4n 2 3'
```



```
case('F432')
  num='209'
  schn='O^3'
  hall='F 4 2 3'
case('F4132')
  num='210'
  schn='O^4'
  hall='F 4d 2 3'
case('I432')
  num='211'
  schn='O^5'
  hall='I 4 2 3'
case('P4332')
  num='212'
  schn='O^6'
  hall='P 4acd 2ab 3'
case('P4132')
  num='213'
  schn='O^7'
  hall='P 4bd 2ab 3'
case('I4132')
  num='214'
  schn='O^8'
  hall='I 4bd 2c 3'
case('P-43m')
  num='215'
  schn='Td^1'
  hall='P -4 2 3'
case('F-43m')
  num='216'
  schn='Td^2'
  hall='F -4 2 3'
case('I-43m')
  num='217'
  schn='Td^3'
  hall='I -4 2 3'
case('P-43n')
  num='218'
  schn='Td^4'
  hall='P -4n 2 3'
case('F-43c')
  num='219'
  schn='Td^5'
  hall='F -4c 2 3'
case('I-43d')
  num='220'
  schn='Td^6'
  hall='I -4bd 2c 3'
case('Pm-3m')
  num='221'
  schn='Oh^1'
  hall='-P 4 2 3'
case('Pn-3n:1')
  num='222:1'
  schn='Oh^2'
  hall='P 4 2 3 -1n'
case('Pn-3n:2')
  num='222:2'
  schn='Oh^2'
  hall='-P 4a 2bc 3'
case('Pm-3n')
  num='223'
  schn='Oh^3'
  hall='-P 4n 2 3'
case('Pn-3m:1')
  num='224:1'
  schn='Oh^4'
  hall='P 4n 2 3 -1n'
case('Pn-3m:2')
```

```
  num='224:2'
  schn='Oh^4'
  hall='-P 4bc 2bc 3'
case('Fm-3m')
  num='225'
  schn='Oh^5'
  hall='-F 4 2 3'
case('Fm-3c')
  num='226'
  schn='Oh^6'
  hall='-F 4c 2 3'
case('Fd-3m:1')
  num='227:1'
  schn='Oh^7'
  hall='F 4d 2 3 -1d'
case('Fd-3m:2')
  num='227:2'
  schn='Oh^7'
  hall='-F 4vw 2vw 3'
case('Fd-3c:1')
  num='228:1'
  schn='Oh^8'
  hall='F 4d 2 3 -1cd'
case('Fd-3c:2')
  num='228:2'
  schn='Oh^8'
  hall='-F 4cvw 2vw 3'
case('Im-3m')
  num='229'
  schn='Oh^9'
  hall='-I 4 2 3'
case('Ia-3d')
  num='230'
  schn='Oh^10'
  hall='-I 4bd 2c 3'
case default
  write(*,*)
  write(*,'("Error(sgsymb): Hermann-Mauguin symbol ''",A,"'' not found")') &
   trim(adjustl(hrmg))
  write(*,*)
  stop
end select
return
end subroutine
!EOC



! Copyright (C) 2006 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU General Public License.
! See the file COPYING for license details.

logical function seitzeq(eps,sr1,st1,sr2,st2)
implicit none
! arguments
real(8), intent(in) :: eps
real(8), intent(in) :: sr1(3,3)
real(8), intent(in) :: st1(3)
real(8), intent(in) :: sr2(3,3)
real(8), intent(in) :: st2(3)
! local variables
integer j
real(8) v1(3),v2(3)
seitzeq=.false.
do j=1,3
  v1(:)=sr1(:,j)+st1(:)
  v2(:)=sr2(:,j)+st2(:)
  if ((abs(v1(1)-v2(1)).gt.eps).or. &
      (abs(v1(2)-v2(2)).gt.eps).or. &
```



```
      (abs(v1(3)-v2(3)).gt.eps)) return
end do
seitzeq=.true.
return
end function



! Copyright (C) 2006 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU General Public License.
! See the file COPYING for license details.

subroutine seitzgen(hall,ngen,srgen,stgen)
implicit none
character(20), intent(in) :: hall
integer, intent(out) :: ngen
real(8), intent(out) :: srgen(3,3,12)
real(8), intent(out) :: stgen(3,12)
! local variables
logical pr
integer i,m,n,no,nop
integer axis,id(3)
! zero vector tolerance
real(8), parameter :: eps=1.d-6
real(8) av(3),r(3,3),t1
real(8) v1(3),v2(3),v3(3)
real(8) tmpmat(3,3)
character(20) str1,str2,str3
str1=trim(adjustl(hall))//''
no=0
nop=0
axis=0
n=0
10 continue
! check for origin shift vector
if (scan(str1,'(').eq.1) then
  if (index(str1,'(0 0 1)').ne.0) then
    v1(1)=0.d0; v1(2)=0.d0; v1(3)=1.d0
  else if (index(str1,'(0 0 -1)').ne.0) then
    v1(1)=0.d0; v1(2)=0.d0; v1(3)=-1.d0
  else
    write(*,*)
    write(*,'("Error(seitzgen): origin-shift not available : ",A)') trim(str1)
    write(*,*)
    stop
  end if
  v1(:)=v1(:)/12.d0
! apply vector shift to all Seitz matrices
  do i=1,ngen
    v3(:)=-v1(:)
!   call r3mv(srgen(:,:,i),v3,v2)
    tmpmat(:,:)=srgen(:,:,i) ; call r3mv(tmpmat,v3,v2)
    v2(:)=v2(:)+stgen(:,i)
    stgen(:,i)=v2(:)+v1(:)
  end do
  goto 20
end if
m=scan(str1,' ')
if (m.le.1) goto 20
str2=str1(1:m-1)
n=n+1
!------------------------------!
!     lattice translations     !
!------------------------------!
if (n.eq.1) then
  stgen(:,1)=0.d0
  if (scan(str2,'P').ne.0) then
    ngen=1
  else if (scan(str2,'A').ne.0) then
```

```
    stgen(1,2)=0.d0
    stgen(2,2)=0.5d0
    stgen(3,2)=0.5d0
    ngen=2
  else if (scan(str2,'B').ne.0) then
    stgen(1,2)=0.5d0
    stgen(2,2)=0.d0
    stgen(3,2)=0.5d0
    ngen=2
  else if (scan(str2,'C').ne.0) then
    stgen(1,2)=0.5d0
    stgen(2,2)=0.5d0
    stgen(3,2)=0.d0
    ngen=2
  else if (scan(str2,'I').ne.0) then
    stgen(:,2)=0.5d0
    ngen=2
  else if (scan(str2,'R').ne.0) then
    stgen(1,2)=0.66666666666666667d0
    stgen(2,2)=0.33333333333333333d0
    stgen(3,2)=0.33333333333333333d0
    stgen(1,3)=0.33333333333333333d0
    stgen(2,3)=0.66666666666666667d0
    stgen(3,3)=0.66666666666666667d0
    ngen=3
  else if (scan(str2,'S').ne.0) then
    stgen(1,2)=0.33333333333333333d0
    stgen(2,2)=0.33333333333333333d0
    stgen(3,2)=0.66666666666666667d0
    stgen(1,3)=0.66666666666666667d0
    stgen(2,3)=0.66666666666666667d0
    stgen(3,3)=0.33333333333333333d0
    ngen=3
  else if (scan(str2,'T').ne.0) then
    stgen(1,2)=0.33333333333333333d0
    stgen(2,2)=0.66666666666666667d0
    stgen(3,2)=0.33333333333333333d0
    stgen(1,3)=0.66666666666666667d0
    stgen(2,3)=0.33333333333333333d0
    stgen(3,3)=0.66666666666666667d0
    ngen=3
  else if (scan(str2,'F').ne.0) then
    stgen(1,2)=0.d0
    stgen(2,2)=0.5d0
    stgen(3,2)=0.5d0
    stgen(1,3)=0.5d0
    stgen(2,3)=0.d0
    stgen(3,3)=0.5d0
    stgen(1,4)=0.5d0
    stgen(2,4)=0.5d0
    stgen(3,4)=0.d0
    ngen=4
  else
    write(*,*)
    write(*,'("Error(seitzgen): Lattice symbol ''",A,"'' not found")') &
     trim(str2)
    write(*,*)
    stop
  end if
! set the rotations to the identity
  do i=1,ngen
    srgen(1,1,i)=1.d0; srgen(1,2,i)=0.d0; srgen(1,3,i)=0.d0
    srgen(2,1,i)=0.d0; srgen(2,2,i)=1.d0; srgen(2,3,i)=0.d0
    srgen(3,1,i)=0.d0; srgen(3,2,i)=0.d0; srgen(3,3,i)=1.d0
  end do
! check if lattice is centrosymmetric
  if (scan(str2,'-').ne.0) then
    do i=ngen+1,2*ngen
      srgen(:,:,i)=-srgen(:,:,i-ngen)
```



```
      stgen(:,i)=stgen(:,i-ngen)
    end do
    ngen=2*ngen
  end if
end if
!------------------------------!
!     rotation-translations     !
!------------------------------!
if (n.ge.2) then
! determine if rotation is proper or improper
  if (scan(str2,'-').eq.1) then
    pr=.false.
! remove the minus sign
    str3=str2(2:)
    str2=str3
  else
    pr=.true.
  end if
! determine the order of rotation
  if (scan(str2,'1').eq.1) then
    no=1
  else if (scan(str2,'2').eq.1) then
    no=2
  else if (scan(str2,'3').eq.1) then
    no=3
  else if (scan(str2,'4').eq.1) then
    no=4
  else if (scan(str2,'6').eq.1) then
    no=6
  else
    write(*,*)
    write(*,'("Error(seitzgen): invalid rotation order for Hall symbol ''",A,&
   &"''")') trim(hall)
    write(*,*)
    stop
  end if
! determine the axis of rotation
  if (scan(str2,'x').ne.0) then
! a axis
    axis=1
  else if (scan(str2,'y').ne.0) then
! b axis
    axis=2
  else if (scan(str2,'z').ne.0) then
! c axis
    axis=3
  else if (scan(str2,'"').ne.0) then
! a+b
    axis=5
  else if (scan(str2,'*').ne.0) then
! a+b+c axis
    axis=6
  else if (n.eq.2) then
! default first rotation is along c
    axis=3
  else if ((n.eq.3).and.(no.eq.2)) then
! default second rotation
    if ((nop.eq.2).or.(nop.eq.4)) then
! a axis
      axis=1
    else if ((nop.eq.3).or.(nop.eq.6)) then
! a-b axis
      axis=4
    else
      write(*,*)
      write(*,'("Error(seitzgen): malformed Hall symbol ''",A,"''")') trim(hall)
      write(*,'(" for default second rotation")')
      write(*,*)
      stop
```

```
    end if
  else if ((n.eq.4).and.(no.eq.3)) then
! third rotation around a+b+c axis
    axis=6
  else if (no.eq.1) then
! arbitrary axis for identity
    axis=1
  else
    write(*,*)
    write(*,'("Error(seitzgen): malformed Hall symbol ''",A,"''")') trim(hall)
    write(*,*)
    stop
  end if
! determine axis vector
  av(:)=0.d0
  if (axis.eq.1) then
! a axis
    av(1)=1.d0
  else if (axis.eq.2) then
! b axis
    av(2)=1.d0
  else if (axis.eq.3) then
! c axis
    av(3)=1.d0
  else if (axis.eq.4) then
! a-b axis
    av(1)=1.d0
    av(2)=-1.d0
  else if (axis.eq.5) then
! a+b axis
    av(1)=1.d0
    av(2)=1.d0
  else if (axis.eq.6) then
! a+b+c axis
    av(:)=1.d0
  end if
! compute the rotation part of the Seitz matrix
  if (axis.eq.1) then
! a axis
    if (no.eq.1) then
      r(1,1)= 1.d0; r(1,2)= 0.d0; r(1,3)= 0.d0
      r(2,1)= 0.d0; r(2,2)= 1.d0; r(2,3)= 0.d0
      r(3,1)= 0.d0; r(3,2)= 0.d0; r(3,3)= 1.d0
    else if (no.eq.2) then
      r(1,1)= 1.d0; r(1,2)= 0.d0; r(1,3)= 0.d0
      r(2,1)= 0.d0; r(2,2)=-1.d0; r(2,3)= 0.d0
      r(3,1)= 0.d0; r(3,2)= 0.d0; r(3,3)=-1.d0
    else if (no.eq.3) then
      r(1,1)= 1.d0; r(1,2)= 0.d0; r(1,3)= 0.d0
      r(2,1)= 0.d0; r(2,2)= 0.d0; r(2,3)=-1.d0
      r(3,1)= 0.d0; r(3,2)= 1.d0; r(3,3)=-1.d0
    else if (no.eq.4) then
      r(1,1)= 1.d0; r(1,2)= 0.d0; r(1,3)= 0.d0
      r(2,1)= 0.d0; r(2,2)= 0.d0; r(2,3)=-1.d0
      r(3,1)= 0.d0; r(3,2)= 1.d0; r(3,3)= 0.d0
    else if (no.eq.6) then
      r(1,1)= 1.d0; r(1,2)= 0.d0; r(1,3)= 0.d0
      r(2,1)= 0.d0; r(2,2)= 1.d0; r(2,3)=-1.d0
      r(3,1)= 0.d0; r(3,2)= 1.d0; r(3,3)= 0.d0
    end if
  else if (axis.eq.2) then
! b axis
    if (no.eq.1) then
      r(1,1)= 1.d0; r(1,2)= 0.d0; r(1,3)= 0.d0
      r(2,1)= 0.d0; r(2,2)= 1.d0; r(2,3)= 0.d0
      r(3,1)= 0.d0; r(3,2)= 0.d0; r(3,3)= 1.d0
    else if (no.eq.2) then
      r(1,1)=-1.d0; r(1,2)= 0.d0; r(1,3)= 0.d0
      r(2,1)= 0.d0; r(2,2)= 1.d0; r(2,3)= 0.d0
```



```
      r(3,1)= 0.d0; r(3,2)= 0.d0; r(3,3)=-1.d0
    else if (no.eq.3) then
      r(1,1)=-1.d0; r(1,2)= 0.d0; r(1,3)= 1.d0
      r(2,1)= 0.d0; r(2,2)= 1.d0; r(2,3)= 0.d0
      r(3,1)=-1.d0; r(3,2)= 0.d0; r(3,3)= 0.d0
    else if (no.eq.4) then
      r(1,1)= 0.d0; r(1,2)= 0.d0; r(1,3)= 1.d0
      r(2,1)= 0.d0; r(2,2)= 1.d0; r(2,3)= 0.d0
      r(3,1)=-1.d0; r(3,2)= 0.d0; r(3,3)= 0.d0
    else if (no.eq.6) then
      r(1,1)= 0.d0; r(1,2)= 0.d0; r(1,3)= 1.d0
      r(2,1)= 0.d0; r(2,2)= 1.d0; r(2,3)= 0.d0
      r(3,1)=-1.d0; r(3,2)= 0.d0; r(3,3)= 1.d0
    end if
  else if (axis.eq.3) then
! c axis
    if (no.eq.1) then
      r(1,1)= 1.d0; r(1,2)= 0.d0; r(1,3)= 0.d0
      r(2,1)= 0.d0; r(2,2)= 1.d0; r(2,3)= 0.d0
      r(3,1)= 0.d0; r(3,2)= 0.d0; r(3,3)= 1.d0
    else if (no.eq.2) then
      r(1,1)=-1.d0; r(1,2)= 0.d0; r(1,3)= 0.d0
      r(2,1)= 0.d0; r(2,2)=-1.d0; r(2,3)= 0.d0
      r(3,1)= 0.d0; r(3,2)= 0.d0; r(3,3)= 1.d0
    else if (no.eq.3) then
      r(1,1)= 0.d0; r(1,2)=-1.d0; r(1,3)= 0.d0
      r(2,1)= 1.d0; r(2,2)=-1.d0; r(2,3)= 0.d0
      r(3,1)= 0.d0; r(3,2)= 0.d0; r(3,3)= 1.d0
    else if (no.eq.4) then
      r(1,1)= 0.d0; r(1,2)=-1.d0; r(1,3)= 0.d0
      r(2,1)= 1.d0; r(2,2)= 0.d0; r(2,3)= 0.d0
      r(3,1)= 0.d0; r(3,2)= 0.d0; r(3,3)= 1.d0
    else if (no.eq.6) then
      r(1,1)= 1.d0; r(1,2)=-1.d0; r(1,3)= 0.d0
      r(2,1)= 1.d0; r(2,2)= 0.d0; r(2,3)= 0.d0
      r(3,1)= 0.d0; r(3,2)= 0.d0; r(3,3)= 1.d0
    end if
  else if (axis.eq.4) then
! a-b axis
    r(1,1)= 0.d0; r(1,2)=-1.d0; r(1,3)= 0.d0
    r(2,1)=-1.d0; r(2,2)= 0.d0; r(2,3)= 0.d0
    r(3,1)= 0.d0; r(3,2)= 0.d0; r(3,3)=-1.d0
  else if (axis.eq.5) then
! a+b axis
    r(1,1)= 0.d0; r(1,2)= 1.d0; r(1,3)= 0.d0
    r(2,1)= 1.d0; r(2,2)= 0.d0; r(2,3)= 0.d0
    r(3,1)= 0.d0; r(3,2)= 0.d0; r(3,3)=-1.d0
  else if (axis.eq.6) then
! a+b+c axis
    r(1,1)= 0.d0; r(1,2)= 0.d0; r(1,3)= 1.d0
    r(2,1)= 1.d0; r(2,2)= 0.d0; r(2,3)= 0.d0
    r(3,1)= 0.d0; r(3,2)= 1.d0; r(3,3)= 0.d0
  end if
! check if axis is invariant with respect to rotation
  call r3mv(r,av,v1)
  t1=sum(abs(av(:)-v1(:)))
  if (t1.gt.eps) then
    write(*,*)
    write(*,'("Error(seitzgen): axis not invariant with respect to rotation")')
    write(*,'(" for Hall symbol ''",A,"''")') trim(hall)
    write(*,*)
    stop
  end if
! apply inverse for improper rotation
  if (.not.pr) r(:,:)=-r(:,:)
! increment Seitz matrix count
  ngen=ngen+1
! store rotation in main array
  srgen(:,:,ngen)=r(:,:)
```

```
! remove rotation symbol
  str3=str2(2:)
  str2=str3
! determine translations
  stgen(:,ngen)=0.d0
  if (scan(str2,'a').ne.0) then
    stgen(1,ngen)=stgen(1,ngen)+0.5d0
  end if
  if (scan(str2,'b').ne.0) then
    stgen(2,ngen)=stgen(2,ngen)+0.5d0
  end if
  if (scan(str2,'c').ne.0) then
    stgen(3,ngen)=stgen(3,ngen)+0.5d0
  end if
  if (scan(str2,'n').ne.0) then
    stgen(:,ngen)=stgen(:,ngen)+0.5d0
  end if
  if (scan(str2,'u').ne.0) then
    stgen(1,ngen)=stgen(1,ngen)+0.25d0
  end if
  if (scan(str2,'v').ne.0) then
    stgen(2,ngen)=stgen(2,ngen)+0.25d0
  end if
  if (scan(str2,'w').ne.0) then
    stgen(3,ngen)=stgen(3,ngen)+0.25d0
  end if
  if (scan(str2,'d').ne.0) then
    stgen(:,ngen)=stgen(:,ngen)+0.25d0
  end if
  if (scan(str2,'1').ne.0) then
    if (no.eq.3) then
      stgen(:,ngen)=stgen(:,ngen)+0.33333333333333333d0*av(:)
    else if (no.eq.4) then
      stgen(:,ngen)=stgen(:,ngen)+0.25d0*av(:)
    else if (no.eq.6) then
      stgen(:,ngen)=stgen(:,ngen)+0.166666666666666667d0*av(:)
    end if
  else if (scan(str2,'2').ne.0) then
    if (no.eq.3) then
      stgen(:,ngen)=stgen(:,ngen)+0.666666666666666667d0*av(:)
    else if (no.eq.6) then
      stgen(:,ngen)=stgen(:,ngen)+0.33333333333333333d0*av(:)
    end if
  else if (scan(str2,'3').ne.0) then
    if (no.eq.4) then
      stgen(:,ngen)=stgen(:,ngen)+0.75d0*av(:)
    end if
  else if (scan(str2,'4').ne.0) then
    if (no.eq.6) then
      stgen(:,ngen)=stgen(:,ngen)+0.666666666666666667d0*av(:)
    end if
  else if (scan(str2,'5').ne.0) then
    if (no.eq.6) then
      stgen(:,ngen)=stgen(:,ngen)+0.833333333333333333d0*av(:)
    end if
  end if
end if
str3=adjustl(str1(m:))
str1=str3
nop=no
goto 10
20 continue
! map translations to [0,1)
do i=1,ngen
  call r3frac(eps,stgen(:,i),id)
end do
return
end subroutine
```



```
! Copyright (C) 2006 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU General Public License.
! See the file COPYING for license details.

subroutine seitzmul(eps,sr1,st1,sr2,st2,sr3,st3)
implicit none
! arguments
real(8), intent(in) :: eps
real(8), intent(in) :: sr1(3,3)
real(8), intent(in) :: st1(3)
real(8), intent(in) :: sr2(3,3)
real(8), intent(in) :: st2(3)
real(8), intent(out) :: sr3(3,3)
real(8), intent(out) :: st3(3)
! local variables
integer id(3)
call r3mv(sr1,st2,st3)
st3(:)=st3(:)+st1(:)
call r3frac(eps,st3,id)
call r3mm(sr1,sr2,sr3)
return
end subroutine
! Copyright (C) 2006 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU General Public License.
! See the file COPYING for license details.

subroutine gengroup(ngen,srgen,stgen,ngrp,srgrp,stgrp)
implicit none
! arguments
integer, intent(in) :: ngen
real(8), intent(in) :: srgen(3,3,ngen)
real(8), intent(in) :: stgen(3,ngen)
integer, intent(out) :: ngrp
real(8), intent(out) :: srgrp(3,3,192)
real(8), intent(out) :: stgrp(3,192)
! local variables
integer i,j,k
real(8), parameter :: eps=1.d-6
real(8) sr(3,3),st(3)
! external functions
logical seitzeq
external seitzeq
! store the identity
ngrp=1
srgrp(1,1,1)=1.d0; srgrp(1,2,1)=0.d0; srgrp(1,3,1)=0.d0
srgrp(2,1,1)=0.d0; srgrp(2,2,1)=1.d0; srgrp(2,3,1)=0.d0
srgrp(3,1,1)=0.d0; srgrp(3,2,1)=0.d0; srgrp(3,3,1)=1.d0
stgrp(:,1)=0.d0
10 continue
! right multiply by the generators
do i=1,ngen
  do j=1,ngrp
    call seitzmul(eps,srgrp(:,:,j),stgrp(:,j),srgen(:,:,i),stgen(:,i),sr,st)
! check if the new element already exists
    do k=1,ngrp
      if (seitzeq(eps,srgrp(:,:,k),stgrp(:,k),sr,st)) goto 20
    end do
    goto 40
20 continue
  end do
end do
! left multiply by the generators
do i=1,ngen
  do j=1,ngrp
    call seitzmul(eps,srgen(:,:,i),stgen(:,i),srgrp(:,:,j),stgrp(:,j),sr,st)
! check if the new element already exists
    do k=1,ngrp
```

```
      if (seitzeq(eps,srgrp(:,:,k),stgrp(:,k),sr,st)) goto 30
    end do
    goto 40
30 continue
  end do
end do
! all elements accounted for
return
40 continue
! add new element
ngrp=ngrp+1
if (ngrp.gt.192) then
  write(*,*)
  write(*,'("Error(gengroup): more than 192 group elements")')
  write(*,*)
  stop
end if
srgrp(:,:,ngrp)=sr(:,:)
stgrp(:,ngrp)=st(:)
goto 10
return
end subroutine



subroutine gencrystal
use modmain
implicit none
! local variables
integer is,ia,ip,i,j
integer i1,i2,i3
integer id(3),ngen,ngrp
real(8) abr,acr,bcr
real(8) sab,cab,cac,cbc
real(8) v1(3),v2(3),t1
! space group generator Seitz matrices
real(8) srgen(3,3,12),stgen(3,12)
! space group Seitz matrices
real(8) srgrp(3,3,192),stgrp(3,192)
real(8) tmpmat(3,3)
! convert angles from degrees to radians
abr=ab*(pi/180.d0)
acr=ac*(pi/180.d0)
bcr=bc*(pi/180.d0)
! setup lattice vectors
sab=sin(abr)
if (abs(sab).lt.epslat) then
  write(*,*)
  write(*,'("Error(gencrystal): degenerate lattice vectors")')
  write(*,*)
  stop
end if
cab=cos(abr)
cac=cos(acr)
cbc=cos(bcr)
avec(1,1)=a
avec(2,1)=0.d0
avec(3,1)=0.d0
avec(1,2)=b*cab
avec(2,2)=b*sab
avec(3,2)=0.d0
avec(1,3)=c*cac
avec(2,3)=c*(cbc-cab*cac)/sab
avec(3,3)=c*sqrt(sab**2-cac**2+2.d0*cab*cac*cbc-cbc**2)/sab
do i=1,3
  do j=1,3
    if (abs(avec(i,j)).lt.epslat) avec(i,j)=0.d0
  end do
end do
```



```
! scale lattice vectors by the number of unit cells
do i=1,3
  avec(:,i)=avec(:,i)*dble(ncell(i))
end do
! determine the Hall symbol from the Hermann-Mauguin symbol
call sgsymb(hrmg,num,schn,hall)
! determine the space group generators
call seitzgen(hall,ngen,srgen,stgen)
! compute the space group operations
call gengroup(ngen,srgen,stgen,ngrp,srgrp,stgrp)
! compute the equivalent atomic positions
do is=1,nspecies
  natoms(is)=0
  do ip=1,nwpos(is)
    do j=1,ngrp
! apply the space group operation
!     call r3mv(srgrp(:,1,j),wpos(:,ip,is),v1)
      tmpmat(:,:)=srgrp(:,:,j) ; call r3mv(tmpmat,wpos(:,ip,is),v1)
      v1(:)=v1(:)+stgrp(:,j)
      do i1=0,ncell(1)-1
        do i2=0,ncell(2)-1
          do i3=0,ncell(3)-1
            v2(1)=(v1(1)+dble(i1))/dble(ncell(1))
            v2(2)=(v1(2)+dble(i2))/dble(ncell(2))
            v2(3)=(v1(3)+dble(i3))/dble(ncell(3))
            call r3frac(epslat,v2,id)
! check if new position already exists
            do ia=1,natoms(is)
              t1=sum(abs(v2(:)-atposl(:,ia,is)))
              if (t1.lt.epslat) goto 30
            end do
! add new position to list
            natoms(is)=natoms(is)+1
            if (natoms(is).gt.maxatoms) then
              write(*,*)
              write(*,'("Error(gencrystal): natoms too large")')
              write(*,'(" for species ",I4)') is
              write(*,'("Adjust maxatoms and recompile code")')
              write(*,*)
              stop
            end if
            atposl(:,natoms(is),is)=v2(:)
          end do
        end do
      end do
30 continue
    end do
  end do
  natmtot=natmtot+natoms(is)
end do
! set magnetic fields to zero
bfcmt0(:,:,:)=0.d0
! reduce conventional cell to primitive cell if required
if (primcell) call findprim
! find the total number of atoms
natmtot=0
do is=1,nspecies
  natmtot=natmtot+natoms(is)
end do
! determine the Cartesian atomic coordinates
do is=1,nspecies
  do ia=1,natoms(is)
    call r3mv(avec,atposl(:,ia,is),atposc(:,ia,is))
  end do
end do
return
end subroutine
```

```
! Copyright (C) 2007 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU General Public License.
! See the file COPYING for license details.

!BOP
! !ROUTINE: findprim
! !INTERFACE:
subroutine findprim
! !USES:
use modmain
! !DESCRIPTION:
!   This routine finds the smallest primitive cell which produces the same
!   crystal structure as the conventional cell. This is done by searching
!   through all the vectors which connect atomic positions and finding those
!   which leave the crystal structure invariant. Of these, the three shortest
!   which produce a non-zero unit cell volume are chosen.
!
! !REVISION HISTORY:
!   Created April 2007 (JKD)
!EOP
!BOC
implicit none
! local variables
integer is,js,ia,ja,ka,na
integer i1,i2,i3,iv(3),i,j,n
real(8) v1(3),v2(3),v3(3)
real(8) t1,t2
! allocatable arrays
real(8), allocatable :: dp(:)
real(8), allocatable :: vp(:,:)
do is=1,nspecies
  do ia=1,natoms(is)
! make sure all atomic coordinates are in [0,1)
    call r3frac(epslat,atposl(:,ia,is),iv)
! determine atomic Cartesian coordinates
    call r3mv(avec,atposl(:,ia,is),atposc(:,ia,is))
  end do
end do
! find the smallest set of atoms
is=1
do js=1,nspecies
! if a species has only one atom the cell must be primitive
  if (natoms(js).eq.1) return
  if (natoms(js).lt.natoms(is)) is=js
end do
n=27*natoms(is)
allocate(dp(n),vp(3,n))
! generate set of possible lattice vectors
n=0
do ia=1,natoms(is)
  v1(:)=atposl(:,ia,is)-atposl(:,1,is)
  do i1=-1,1
    v2(1)=v1(1)+dble(i1)
    do i2=-1,1
      v2(2)=v1(2)+dble(i2)
      do i3=-1,1
        v2(3)=v1(3)+dble(i3)
        t1=abs(v2(1))+abs(v2(2))+abs(v2(3))
        if (t1.lt.epslat) goto 20
! check if vector v2 leaves conventional cell invariant
        do js=1,nspecies
          do ja=1,natoms(js)
            v3(:)=atposl(:,ja,js)+v2(:)
            call r3frac(epslat,v3,iv)
            do ka=1,natoms(js)
! check both positions and magnetic fields are the same
              t1=sum(abs(atposl(:,ka,js)-v3(:)))
              t2=sum(abs(bfcmt0(:,ja,js)-bfcmt0(:,ka,js)))
```



```
              if ((t1.lt.epslat).and.(t2.lt.epslat)) goto 10
            end do
! atom ja has no equivalent under translation by v2
            goto 20
10 continue
          end do
        end do
! cell invariant under translation by v2, so add to list
        n=n+1
        call r3mv(avec,v2,vp(:,n))
        dp(n)=sqrt(vp(1,n)**2+vp(2,n)**2+vp(3,n)**2)
20 continue
      end do
    end do
  end do
end do
if (n.eq.0) then
  write(*,*)
  write(*,'("Error(findprim): cannot find any lattice vectors")')
  write(*,*)
  stop
end if
! find the shortest lattice vector
j=1
t1=1.d8
do i=1,n
  if (dp(i).lt.t1+epslat) then
    j=i
    t1=dp(i)
  end if
end do
avec(:,1)=vp(:,j)
! find the next shortest lattice vector not parallel to the first
j=1
t1=1.d8
do i=1,n
  call r3cross(avec(:,1),vp(:,i),v1)
  t2=sqrt(v1(1)**2+v1(2)**2+v1(3)**2)
  if (t2.gt.epslat) then
    if (dp(i).lt.t1+epslat) then
      j=i
      t1=dp(i)
    end if
  end if
end do
avec(:,2)=vp(:,j)
! find the next shortest lattice vector which gives non-zero unit cell volume
call r3cross(avec(:,1),avec(:,2),v1)
j=1
t1=1.d8
do i=1,n
  t2=dot_product(vp(:,i),v1(:))
  if (abs(t2).gt.epslat) then
    if (dp(i).lt.t1+epslat) then
      j=i
      t1=dp(i)
    end if
  end if
end do
avec(:,3)=vp(:,j)
call r3minv(avec,ainv)
! remove redundant atoms
do is=1,nspecies
  na=0
  do ia=1,natoms(is)
    call r3mv(ainv,atposc(:,ia,is),v1)
    call r3frac(epslat,v1,iv)
    do ja=1,na
      t1=sum(abs(atposl(:,ja,is)-v1(:)))
```

```
      if (t1.lt.epslat) goto 30
    end do
    na=na+1
    atposl(:,na,is)=v1(:)
    call r3mv(avec,atposl(:,na,is),atposc(:,na,is))
    bfcmt0(:,na,is)=bfcmt0(:,ia,is)
30 continue
  end do
  natoms(is)=na
end do
deallocate(dp,vp)
return
end subroutine
!EOC
! Copyright (C) 2002-2005 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU Lesser General Public
! License. See the file COPYING for license details.

!BOP
! !ROUTINE: r3cross
! !INTERFACE:
subroutine r3cross(x,y,z)
! !INPUT/OUTPUT PARAMETERS:
!   x : input vector 1 (in,real(3))
!   y : input vector 2 (in,real(3))
!   z : output cross-product (out,real(3))
! !DESCRIPTION:
!   Returns the cross product of two real 3-vectors.
!
! !REVISION HISTORY:
!   Created September 2002 (JKD)
!EOP
!BOC
implicit none
! arguments
real(8), intent(in) :: x(3)
real(8), intent(in) :: y(3)
real(8), intent(out) :: z(3)
z(1)=x(2)*y(3)-x(3)*y(2)
z(2)=x(3)*y(1)-x(1)*y(3)
z(3)=x(1)*y(2)-x(2)*y(1)
return
end subroutine
!EOC


! Copyright (C) 2002-2005 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU Lesser General Public
! License. See the file COPYING for license details.

!BOP
! !ROUTINE: r3frac
! !INTERFACE:
subroutine r3frac(eps,v,iv)
! !INPUT/OUTPUT PARAMETERS:
!   eps : zero component tolerance (in,real)
!   v   : input vector (inout,real(3))
!   iv  : integer parts of v (out,integer(3))
! !DESCRIPTION:
!   Finds the fractional part of each component of a real 3-vector using the
!   function ${\rm frac}\,(x)=x-\lfloor x\rfloor$. A component is taken to be
!   zero if it lies within the intervals $[0,\epsilon)$ or $(1-\epsilon,1]$.
!   The integer components of {\tt v} are returned in the variable {\tt iv}.
!
! !REVISION HISTORY:
!   Created January 2003 (JKD)
!EOP
!BOC
implicit none
```



```
! arguments
real(8), intent(in) :: eps
real(8), intent(inout) :: v(3)
integer, intent(out) :: iv(3)
! local variables
integer i
do i=1,3
  iv(i)=int(v(i))
  v(i)=v(i)-dble(iv(i))
  if (v(i).lt.0.d0) then
    v(i)=v(i)+1.d0
    iv(i)=iv(i)-1
  end if
  if (1.d0-v(i).lt.eps) then
    v(i)=0.d0
    iv(i)=iv(i)+1
  end if
  if (v(i).lt.eps) then
    v(i)=0.d0
  end if
end do
return
end subroutine
!EOC


! Copyright (C) 2002-2005 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU Lesser General Public
! License. See the file COPYING for license details.

!BOP
! !ROUTINE: r3minv
! !INTERFACE:
subroutine r3minv(a,b)
! !INPUT/OUTPUT PARAMETERS:
!   a : input matrix (in,real(3,3))
!   b : output matrix (in,real(3,3))
! !DESCRIPTION:
!   Computes the inverse of a real $3\times 3$ matrix.
!
! !REVISION HISTORY:
!   Created April 2003 (JKD)
!EOP
!BOC
implicit none
! arguments
real(8), intent(in) :: a(3,3)
real(8), intent(out) :: b(3,3)
! local variables
real(8) t1
t1=a(1,2)*a(2,3)*a(3,1)-a(1,3)*a(2,2)*a(3,1)+a(1,3)*a(2,1)*a(3,2) &
  -a(1,1)*a(2,3)*a(3,2)+a(1,1)*a(2,2)*a(3,3)-a(1,2)*a(2,1)*a(3,3)
if (abs(t1).lt.1.d-40) then
  write(*,*)
  write(*,'("Error(r3minv): singular matrix")')
  write(*,*)
  stop
end if
t1=1.d0/t1
b(1,1)=(a(2,2)*a(3,3)-a(2,3)*a(3,2))*t1
b(1,2)=(a(1,3)*a(3,2)-a(1,2)*a(3,3))*t1
b(1,3)=(a(1,2)*a(2,3)-a(1,3)*a(2,2))*t1
b(2,1)=(a(2,3)*a(3,1)-a(2,1)*a(3,3))*t1
b(2,2)=(a(1,1)*a(3,3)-a(1,3)*a(3,1))*t1
b(2,3)=(a(1,3)*a(2,1)-a(1,1)*a(2,3))*t1
b(3,1)=(a(2,1)*a(3,2)-a(2,2)*a(3,1))*t1
b(3,2)=(a(1,2)*a(3,1)-a(1,1)*a(3,2))*t1
b(3,3)=(a(1,1)*a(2,2)-a(1,2)*a(2,1))*t1
return
```

```
end subroutine
!EOC


! Copyright (C) 2002-2005 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU Lesser General Public
! License. See the file COPYING for license details.

!BOP
! !ROUTINE: r3mm
! !INTERFACE:
subroutine r3mm(a,b,c)
! !INPUT/OUTPUT PARAMETERS:
!   a : input matrix 1 (in,real(3,3))
!   b : input matrix 2 (in,real(3,3))
!   c : output matrix (out,real(3,3))
! !DESCRIPTION:
!   Multiplies two real $3\times 3$ matrices.
!
! !REVISION HISTORY:
!   Created April 2003 (JKD)
!EOP
!BOC
implicit none
! arguments
real(8), intent(in) :: a(3,3)
real(8), intent(in) :: b(3,3)
real(8), intent(out) :: c(3,3)
c(1,1)=a(1,1)*b(1,1)+a(1,2)*b(2,1)+a(1,3)*b(3,1)
c(2,1)=a(2,1)*b(1,1)+a(2,2)*b(2,1)+a(2,3)*b(3,1)
c(3,1)=a(3,1)*b(1,1)+a(3,2)*b(2,1)+a(3,3)*b(3,1)
c(1,2)=a(1,1)*b(1,2)+a(1,2)*b(2,2)+a(1,3)*b(3,2)
c(2,2)=a(2,1)*b(1,2)+a(2,2)*b(2,2)+a(2,3)*b(3,2)
c(3,2)=a(3,1)*b(1,2)+a(3,2)*b(2,2)+a(3,3)*b(3,2)
c(1,3)=a(1,1)*b(1,3)+a(1,2)*b(2,3)+a(1,3)*b(3,3)
c(2,3)=a(2,1)*b(1,3)+a(2,2)*b(2,3)+a(2,3)*b(3,3)
c(3,3)=a(3,1)*b(1,3)+a(3,2)*b(2,3)+a(3,3)*b(3,3)
return
end subroutine
!EOC


! Copyright (C) 2002-2005 J. K. Dewhurst, S. Sharma and C. Ambrosch-Draxl.
! This file is distributed under the terms of the GNU Lesser General Public
! License. See the file COPYING for license details.

!BOP
! !ROUTINE: r3mv
! !INTERFACE:
subroutine r3mv(a,x,y)
! !INPUT/OUTPUT PARAMETERS:
!   a : input matrix (in,real(3,3))
!   x : input vector (in,real(3))
!   y : output vector (out,real(3))
! !DESCRIPTION:
!   Multiplies a real $3\times 3$ matrix with a vector.
!
! !REVISION HISTORY:
!   Created January 2003 (JKD)
!EOP
!BOC
implicit none
! arguments
real(8), intent(in) :: a(3,3)
real(8), intent(in) :: x(3)
real(8), intent(out) :: y(3)
y(1)=a(1,1)*x(1)+a(1,2)*x(2)+a(1,3)*x(3)
y(2)=a(2,1)*x(1)+a(2,2)*x(2)+a(2,3)*x(3)
y(3)=a(3,1)*x(1)+a(3,2)*x(2)+a(3,3)*x(3)
```



```
return
end subroutine
!EOC
```