FILE: RW.F90

```fortran
program rw

   use global_env
   use calendar
   use walker
   use cell

   implicit none

   integer :: max_events
   integer :: action, opt1, opt2

   real(DP) :: time, time_limit, time_stat
   real(DP) :: sigma, rho, xx

   integer, dimension(:,:), allocatable :: p0

   real(DP), dimension(:), allocatable :: rn

   integer :: i, j, k, l, m, n
   integer :: n_errors

   character(len=10) :: arg

   debug = .false.

   ! The following if statement checks to see whether you supply any
   ! flags to the compiler. The possible flags are "debug" and
   ! "newseed". The debug flag will output information about the
   ! running of the program and will introduce additional stop
   ! commands. The newseed flag ensures that all the random numbers are
   ! generated from a different seed. This is something that we would
   ! like to turn off during testing.

   if(iargc()>0) then
      do i=1,iargc()
         call getarg(i,arg)
         select case (trim(arg))
         case ('-debug')
            debug = .true.
         case  ('-newseed')
            call newseed()
         end select
      end do
   end if

   ! Extract the needed data from the configuration file

   open(10,file='config.dat')
   read(10,*) n_walkers
   read(10,*) box_size
   read(10,*) cell_size
   read(10,*) k_3d, k_2d
   read(10,*) k_on, k_off
   read(10,*) time_limit
   read(10,*) time_stat
   close(10)

```

```
! open log file for output

open(fmain,file='rw.log')

write(fmain,*) '*** KMC FRAP *** '
write(fmain,*) 'Started at ',time_stamp()
write(fmain,'(/a,i5)')  'Number of Random Walkers=',n_walkers
write(fmain,'(3(a,i4))') 'Box size=',box_size(X),' x',box_size(Y),' x',box_size(Y)
write(fmain,'(3(a,i4))') 'Cell size=',cell_size(X),' x',cell_size(Y),' x',cell_size(Y)
write(fmain,'(2(a,f8.4),a)') 'Transition rate=',k_3d,'(3d) ',k_2d,'(2d)'
write(fmain,'(2(a,f8.4),a)') 'Transition rate=',k_on,'(on) ',k_off,'(off)'
write(fmain,'(a,e15.3)') 'Total simulation time=',time_limit
write(fmain,*)

! open a data file #1
write(fmain,*)  'outout: onoff.dat'
open(fdat1,file='onoff.dat')

! open a data file #2
write(fmain,*) 'output: diffusion.dat'
open(fdat2,file='diffusion.dat')

! open a file for debug information, iff the debug flag is issued
if(debug) then
   write(fmain,*) 'output: debug.dat'
   open(fdbg,file='debug.dat')
```

```
endif

! set the maximum number of walkers.
!(this determines the size of array)
max_walkers = 2*n_walkers

! set the maximum number of events recorded in the calendar at a time.
max_events = 2*max_walkers

! initialize the event calendar
call calendar_init(n_walkers,max_events)

! initialize the random walkers
call walker_init(n_walkers)

! initial position of walkers
!(for example uniformally random distribution)
allocate(rn(n_walkers)) ! Create a vector for the random numbers
call random_number(rn)
walker_pos(1:n_walkers,X) = ceiling(rn*box_size(X))
call random_number(rn)
walker_pos(1:n_walkers,Y) = ceiling(rn*box_size(Y))
call random_number(rn)
walker_pos(1:n_walkers,Z) = ceiling(rn*box_size(Z))
deallocate(rn) ! Not needed anymore, so there isn't any sense in
! letting rn take up any memory

! initialize the cell configuration
```

```
call cell_init(n_walkers,max_walkers,walker_pos)
! This is used for determining particle collisions

if(debug) call cell_test(n_walkers)  ! check the consistency in cell assignment

! predict the jump time for all walkers
do i=1,n_walkers
     call walker_predict_event(i)
end do

! save the initial position
allocate(p0(max_walkers,3))
p0 = walker_pos

! schedule the first data output
call calendar_schedule_event(11,0,0,time_stat)
call calendar_schedule_event(12,0,0,time_stat)

n_collisions = 0
time_current = 0.0_DP

do while (time_current<=time_limit)

   ! find the next event to happen
   call calendar_find_event(action,opt1,opt2)

   ! execute the event
   if (action<=10)  then   ! simple jump event

      ! walker 'opt1' jumps
      call walker_action(opt1,action)
      ! predict next jump of walker 'opt1'
      call walker_predict_event(opt1)

   else if(action<=20) then   ! non-physical events

      select case(action)
         case(11)
            !  evaluate surface density
            sigma = count(walker_pos(:,Z)>box_size(Z))
            rho = count(walker_pos(:,Z)==box_size(Z))
            write(fdat1,'(f10.2,2e13.5)') time_current, rho, sigma
            !  set the next evaluation time
            call calendar_schedule_event(11,0,0,time_current+time_stat)
         case(12)
            !  evaluate mean square displacement
            xx = real(sum((walker_pos-p0)**2),kind=DP)/real(n_walkers,kind=DP)
            write(fdat2,'(f10.2,e13.5)') time_current, xx
            !  set the next evaluation time
            call calendar_schedule_event(12,0,0,time_current+time_stat)
         case default
            ! unknown event
            write(fmain,'(f10.2,a)') time_current, 'unknown event'
            ! emergency stop
            stop
      end select

      else if(action>20) then
         ! binary event (\such as reaction
         write(fmain,'(f10.2,a)') time_current, 'binary event'
         ! binary events have not implimented yet.
      end if

   end do

   write(fmain,'(a,i10)') 'Number of collisions =',n_collisions

   if(debug) call cell_test(n_walkers)

   ! close output files

   close(fdat1)
   close(fdat2)

   !
   open(fdat1,file='walkers.dat')
   ! creation of particles has not been considered yet.
   write(fdat1,'(3i5)') (walker_pos(i,:), i=1,n_walkers)
   close(fdat1)

   if(debug) close(fdbg)

   write(fmain,*) '*** Done at ',time_stamp(), '***'
end program rw
```