```
      subroutine qm_mm(x,iatoms,nquant,jqatms,chrgs,mlabel,
     $           nlink,qmlink,iread,forces,
     $           escf,mark,icalcf,iqmres)
c
c     Argument list variables:
c
c     x(*)        - Cartesian coordinates for all atoms.
c     iatoms      - Total number of atoms.
c     nquant      - Number of quantum atoms.
c     jqatms(*)   - Atom numbers for quantum atoms.
c     chrgs(*)    - Atomic charges for mm atoms.
c     mlabel(*)   - Atomic numbers for qm atoms.
c     nlink       - Number of link atoms.
c     qmlink(*)   - Cartesian coordinates of link atoms.
c     iread       - Read in density matrix if iread=1, new calculation
c                   if iread=0.
c     forces(*)   - Atomic forces.
c     escf        - Heat of formation from MOPAC.
c     mark(*)     - Center of geometry atom list.  For each atom j in a
c                   residue, mark(j) is the atom number in that residue
c                   that is closest to the center of geometry for the residue.
c                   Used for periodic boundary conditions.
c     icalcf      - Calculate forces if icalcf=0, don't calculate forces
c                   otherwise.  Forces are skipped for rollin/rolout step
c                   of PMF or FEP.
c
c     iqmres(*)   - Atom-based interaction list for quantum atoms:
c                   ...,m,i1,i2,...,im,n,j1,j2,...,jn,...
c                   Atom lists contain the numbers (m,n,...) of non-qm
c                   atoms that interact with each qm atom, followed by
c                   the list of interacting non-qm atoms.
c
c     Locally defined arrays:
c
c     qcords(3,numatm)  - Cartesian coordinates of quantum atoms.
c                         (Extracted from x by subroutine tran)
c     dxyzqm(3,numatm)  - Quantum mechanical derivatives from qm-mm
c                         interactions.
c     dxyzcl(3,mxatom)   - Classical derivatives from qm-mm interaction.
c
c
      implicit double precision (a-h,o-z)
#include "SIZES"
      character keywrd*241
C
      COMMON /ATOMIC/ EISOL(107),EHEAT(107)
      COMMON /KEYWRD/ KEYWRD
      COMMON /GEOVAR/ NVAR,LOC(2,MAXPAR), IDUMY, XPARAM(MAXPAR)
      COMMON /GEOSYM/ NDEP,LOCPAR(MAXPAR),IDEPFN(MAXPAR),LOCDEP(MAXPAR)
      COMMON /GEOKST/ NATOMS,LABELS(NUMATM),
     1                NA(NUMATM),NB(NUMATM),NC(NUMATM)
      COMMON /GEOM  / GEO(3,NUMATM)
      COMMON /GRADNT/ GRAD(MAXPAR),GNORM
      common /istat/istat
      COMMON /TIMEa / TIMEb
      COMMON /MOLKST/ NUMAT,NAT(NUMATM),NFIRST(NUMATM),NMIDLE(NUMATM),
     2                NLAST(NUMATM), NORBS, NELECS,NALPHA,NBETA,
     3                NCLOSE,NOPEN,NDUMY,FRACT
c     COMMON /DIPSTO/ UX,UY,UZ,CH(NUMATM)
#include "../sizesqm.h"
      dimension qcords(3,mxatom),jqatms(*),mlabel(*),x(*),chrgs(*)
      dimension qmlink(*),forces(*)
      dimension dxyzqm(3,mxatom),dxyzcl(3,mxatom)
      dimension mark(*),iqmres(*)
c
```

```
      common/setbox/box(3),beta,boxh(3),boxhm,boxhm2,
     $              boxoh,boxoq,cosb,cosb2,ntm,ntb
c
      logical skip,there
      real*4 wwww
      data jkll /0/
      data itimes1 /0/
      data jjvkll /0/
      save jkll
      save itimes1
      save jjvkll
      save inesp,istesp,there,iesp,iesp2
c
c     FEP common block:
c
#include "../feprun.h"
c
      skip = .false.
      kount = 1
 9292 format(f8.4)
      call tran(x,nquant,jqatms,mlabel,nlink,qmlink,
     $                          qcords)
      if(itimes1.eq.0) then
          CALL MPSETUP(qcords)
          itimes1 = itimes1 + 1
          INQUIRE(FILE='cntrl.dat',EXIST=THERE)
          IF(THERE) THEN
              open(63,file='cntrl.dat',status='old')
              read(63,*) inesp,istesp
              close(63)
              iesp=-1
              iesp2=istesp-1
          else
              inesp=1.0d8
              istesp=1.0d8
              iesp=0
              iesp2=0
          endif
      end if
      iesp = iesp + 1
      if(iesp.ge.inesp) iesp2= iesp2+ 1
c
c     If this is an FEP run to zero out QM vdW parameters, and the
c     MM charges have already been removed, then there is no need
c     to do the scf calculation.
c
      if(.not.do_scf)then
        escf = 0.0d0
        go to 5000
      endif

      if( index(keywrd, 'GAUSSIAN') .ne. 0 ) then !wpp
        call gaussfg(XPARAM, .TRUE., ESCF, .TRUE., GRAD, .TRUE.,
     $            qcords,nquant,x,chrgs,iread,dxyzqm,
     $            dxyzcl,icalcf,iqmres,mark,jqatms)
      else !wpp

        CALL COMPFG(XPARAM, .TRUE., ESCF, .TRUE., GRAD, .TRUE.,
     $            qcords,nquant,x,chrgs,iread,dxyzqm,
     $            dxyzcl,icalcf,iqmres,mark,jqatms)
        if (index(keywrd,'MULLIK').ne.0) then
            call atmchg
            call wrtchg
        endif

```

```
131         endif !wpp
132   c
133   c     calculate electrostatic potential
134   c
135   C ESP
136   c
137         if(mod(iesp2,istesp).eq.0.and.iesp.ge.inesp) then
138             write(6,*) 'Calculating ESP charges'
139             call esp(qcords)
140         end if
141         nquant = nquant - nlink
142         istat = 1
143         if(icalcf.ne.0) go to 9999
144       191 continue
145       189 continue
146       400  continue
147   c
148   c     put calculated forces into force array
149   c
150         do 300 i=1,nquant
151             m = jqatms(i)
152             do 320 j=1,3
153                 forces((m-1)*3+j) = forces((m-1)*3+j) - dxyzqm(j,i)
154   c             forces((m-1)*3+j) = 0.0d0
155       320       continue
156       300  continue
157         do 330 i=1,iatoms
158             do 340 j=1,3
159                 forces((i-1)*3+j) = forces((i-1)*3+j) - dxyzcl(j,i)
160   c             forces((i-1)*3+j) = 0.0d0
161       340       continue
162       330  continue
163         if(nlink.ne.0) then
164             ij = iatoms * 3
165             do 350 i=1,nlink
166                 do 360 j=1,3
167                     forces((i-1)*3+j+ij) = -dxyzqm(j,i+nquant)
168   c                 forces((i-1)*3+j+ij) = 0.0d0
169       360           continue
170       350       continue
171         end if
172       9999 continue
173       5000 return
174         end
```