```
function ionization_rate, loc, input, magcoord

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Compute the ionozation rate of the species due to each possible process
;;
;; Version History
;; 3.3: 12/13/2010
;;   * rewriting with new kappa structure
;; 3.2: 7/21/2010
;;   * rewritten with new structure architecture
;; 3.1: 4/26/10
;;   * Added support for Earth (photoionization only)
;;   * Added check for moon's shadow -- before only checked to see if the packet
;;     was in the planet's shadow
;; 3.0: original based on neutlt
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

common constants
common ratecoefs

if (input.options.lifetime NE 0) $    ;;; Explicitly set
  then rate = 1./replicate(input.options.lifetime, n_elements(*loc.t)) $
  else begin
    num = n_elements(*loc.t)

    ;; Get plasma parameters
    dotherm = 0
    doener = 0
    case (input.geometry.planet) of
      'Mercury':
      'Earth':
      'Jupiter': begin
        JupiterPlasma, loc, *magcoord.M, *magcoord.zeta, input.plasma_info, $
          *magcoord.lam, ElecTherm=ElecTherm, ElecEner=ElecEner, IonTherm=IonTherm
        dotherm = 1 & doener = 1
        end
      'Saturn': begin
        SaturnPlasma, *magcoord.M, *magcoord.zeta, ElecTherm=ElecTherm, $
          IonTherm=IonTherm, ElecEner=ElecEner
        dotherm = 1
        doener = 0
        end
      else: stop
    endcase

    chxrate = dblarr(num)

    ;; Compute photo-loss rate
    if (kappa.photo) then begin
      if (n_elements(*magcoord.out_of_shadow) NE num) then stop
```

```
52        photorate = double(*magcoord.out_of_shadow * *kappa.kappa_photo)
53      endif else photorate = dblarr(num)
54      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
55
56      ;; Compute electron impact rate
57      eimprate = dblarr(num)
58      if ((kappa.eimp) and (dotherm)) then begin
59        K = loginterpol(*kappa.kappa_ei, *kappa.t_e, *ElecTherm.t_e)
60        w = where((*ElecTherm.n_e GE 0) and (K GT 0), ctw)
61        if (ctw NE 0) then eimprate[w] = (*ElecTherm.n_e)[w] * K[w]
62        destroy_structure, ElecTherm
63      endif
64
65      if ((kappa.eimp) and (doener)) then begin
66        K = loginterpol(*kappa.kappa_ei, *kappa.t_e, *ElecEner.t_e)
67        w = where((*ElecEner.n_e GE 0) and (K GT 0), ctw)
68        if (ctw NE 0) then eimprate[w] += (*ElecEner.n_e)[w] * K[w]
69        destroy_structure, ElecEner
70      endif
71      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
72
73      ;; Compute charge exchange rate
74      if (kappa.chx) then begin
75        ;; Calculate relative velocity
76        Bvx = -DipoleConsts.magrat * (*loc.x)[*,1]
77        Bvy = DipoleConsts.magrat * (*loc.x)[*,0]
78        vrel = (sqrt(((*loc.v)[*,0]-Bvx)^2 + ((*loc.v)[*,1]-Bvy)^2 + (*loc.v)[*,2]^2)) $
79          * SystemConsts.rplan
80        q = where(vrel GT max(*kappa.v_rel), nq)
81        if (nq NE 0) then vrel[q] = max(*kappa.v_rel)
82        q = where(vrel LT min(*kappa.v_rel), nq)
83        if (nq NE 0) then vrel[q] = min(*kappa.v_rel)
84
85        ;; Compute the rate
86        for kk=0,n_elements(kappa.ions)-1 do chxrate += (*IonTherm.n_i)[*,kk] * $
87          interpolate_xy((*kappa.kappa_chx)[*,*,kk], *kappa.t_i, *kappa.v_rel, $
88          (*IonTherm.t_i)[*,kk], vrel)
89        destroy_structure, IonTherm
90      endif
91
92      rate = photorate + eimprate + chxrate
93      q = where(rate EQ 0, nq) & if (nq NE 0) then rate[q] = 1d-30
94    endelse
95
96  ;Return the lifetimes
97  q = where(finite(rate) EQ 0) & if (q[0] NE -1) then stop
98  return, rate
99
100 end
101
```