```
function lifetime_setup, input

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Set up the default reactions and plasma for use later
;; Function returns loss_info which is only used to keep track of what
;; reactions were used in running the model
;;
;; Version History
;;   3.1: 12/9/2010
;;     * new rate coeficient structure
;;   3.0: 7/19/2010
;;     * Rewriting with input structure
;;   2.4: 4/26/2010
;;     * Added option for Earth - photoionization only
;;   2.3: 9/14/2009
;;     * moved load_plasma section to separate program
;;   2.2: 5/27/2009
;;     * replaced GetReactionList with create_lossinfo
;;     * added Jupiter plasma
;;   2.1: 4/23/2009
;;     * For Mercury, changing the routine so that I can bock photoionization
;;       in the shadow. Will keep the coef_photo structure and ionization rk5 will
;;       call ionization_rate
;;   2.0: 10/22/2008 (MHB)
;;     * Routine created.
;;     * Need to add Jupiter plasma
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

common constants
common ratecoefs
common plasma

;; Find the default reactions
restore, !model.basepath + 'Work/AtomicData/Defaults.Loss.sav'

;; Figure out which reactions to use
q = where(defaults.species EQ input.options.atom, nq)
if (nq EQ 0) then stop
loss_info = defaults[q]

pp = where(strcmp(loss_info.mechanism, 'photo', /fold), np)
ie = where(strcmp(loss_info.mechanism, 'Electron Impact', /fold), nie)
in = where(strcmp(loss_info.mechanism, 'Ion-Neutral', /fold), nin)
case (input.geometry.planet) of
  'Mercury': loss_info = reform(loss_info[pp]) ;; only use photoreactions for now
  'Earth': loss_info = reform(loss_info[pp]) ;; only use photoreactions for now
  else: begin ;; Load the plasma
    load_plasma, input.geometry.planet, input.plasma_info, plasma=plasma, $
      hotplasma=plasmahot
```

```
52      ionlist = [*plasma.ions, *plasmahot.ions]
53      ionlist = ionlist[uniq(ionlist, sort(ionlist))]
54    endelse
55  endcase
56  pp = where(strcmp(loss_info.mechanism, 'photo', /fold), np)
57  ie = where(strcmp(loss_info.mechanism, 'Electron Impact', /fold), nie)
58  in = where(strcmp(loss_info.mechanism, 'Ion-Neutral', /fold), nin)
59
60  ;; Load the photo-reaction rate coefficients
61  photrate = 0d
62  for i=0,np-1 do begin
63    restore, !model.basepath + strmid(loss_info[pp[i]].file, strlen('/Users/mburger/'))
64    photrate += ratecoef.kappa / stuff.aplanet^2
65    destroy_structure, ratecoef
66  endfor
67
68  ;; Load the electron impact rate coefficients
69  if (nie GT 0) then begin
70    electemp = 10.^(dindgen(41)/20.)  ;; electrons valid for 1 eV < t_e < 100 eV
71    eimpcoef = 0d
72    for i=0,nie-1 do begin
73      restore, !model.basepath + strmid(loss_info[ie[i]].file, strlen('/Users/mburger/'))
74      eimpcoef += loginterpol(*ratecoef.kappa, *ratecoef.t_e, electemp)
75      destroy_structure, ratecoef
76    endfor
77  endif else begin
78    electemp = 0. & eimpcoef = 0.
79  endelse
80
81  ;; Load the ion-neutral rate coefficeints
82  if (nin GT 0) then begin
83    vrel = dindgen(101)*2                ;; 0 km/s < v_rel < 200 km/s
84    iontemp = dindgen(101)
85    chxcoef = dblarr(101,101,n_elements(ionlist))
86    for i=0,nin-1 do begin
87      restore, !model.basepath + strmid(loss_info[in[i]].file, strlen('/Users/mburger/'))
88      q = (where(ionlist EQ ratecoef.ion, nq))[0]
89      if (nq GT 1) then stop ;; problem
90      if (nq EQ 1) then chxcoef[*,*,q] += interpolate_xy(*ratecoef.kappa, *ratecoef.t_i, $
91        *ratecoef.v_rel, iontemp, vrel, /grid)
92      print, ratecoef.reaction, '  ', q
93      destroy_structure, ratecoef
94    endfor
95  endif else begin
96    vrel = 0. & iontemp = 0. & chxcoef = 0. & ionlist = ''
97  endelse
98
99  kappa = {photo:(photrate NE 0), eimp:(nie GT 0), chx:(nin GT 0), $
100   kappa_photo:ptr_new(photrate), t_e:ptr_new(electemp), $
101   kappa_ei:ptr_new(eimpcoef), ions:ionlist, t_i:ptr_new(iontemp), $
102   v_rel:ptr_new(vrel), kappa_chx:ptr_new(chxcoef)}
```

```
103
104  return, loss_info
105
106  end
```