```
pro JupiterPlasma, loc, M, zeta, plasma_info, lam, phi, $
  ElecTherm=ElecTherm, ElecEner=ElecEner, IonTherm=IonTherm, IonEner=IonEner

common constants
common ratecoefs
common plasma

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;  Determines the plasma density and temperature as a function of location in
;;  the IPT
;;
;;  OUTPUTS:
;;    ElecTherm: state of the thermal electrons
;;    IonTherm: state of thermal ions
;;    ElecEner: state of the energetic electrons
;;    IonEner: state of energetic ions
;;
;;  Version History
;;    3.1: 1/31/2011
;;    3.0: 12/6/2010
;;      * updating
;;    2.0: 5/27/2009
;;      Starting over from scratch. Removing all the varibility and using a simple
;;      offset, tilted dipole. Can add other effects in later.
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

num = n_elements(*loc.t)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; State of the thermal electrons
if ((kappa.eimp) and (plasma_info.thermal)) then begin
  n_e = interpol(*plasma.n_e, *plasma.L, M)
  t_e = interpol(*plasma.t_e, *plasma.L, M)
  H = interpol(*plasma.h_e, *plasma.L, M)

  hq = where(n_e LE 0, hct) & if (hct NE 0) then n_e[hq] = 0.
  hq = where(t_e LE 0.01, hct) & if (hct NE 0) then t_e[hq] = 0.01
  hq = where(H LE .1, hct)  & if (hct NE 0) then H[hq] = .1

  q = where(M GT max(*plasma.L), nq)
  if (nq NE 0) then begin
    t_e[q] = 0.01
    H[q] = .1
  endif

  n_e = n_e * exp( -(zeta/H)^2 )
  ElecTherm = {n_e:ptr_new(n_e), t_e:ptr_new(t_e)}
endif

```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; State of the energetic electrons
if ((kappa.eimp) and (plasma_info.energetic)) then begin
  n_ehot = interpol(*plasmahot.n_e, *plasmahot.L, M)
  t_ehot = interpol(*plasmahot.t_e, *plasmahot.L, M)
  Hhot = interpol(*plasmahot.h_e, *plasmahot.L, M)

  hq = where(n_ehot LE 0, hct) & if (hct NE 0) then n_ehot[hq] = 0.
  hq = where(t_ehot LE 0.01, hct) & if (hct NE 0) then t_ehot[hq] = 0.01
  hq = where(HHot LE .1, hct) & if (hct NE 0) then HHot[hq] = .1

  q = where(M GT max(*plasmahot.L), nq)
  if (nq NE 0) then begin
    t_ehot[q] = 0.01
    Hhot[q] = .1
  endif

  n_ehot = n_ehot * exp( -(zeta/Hhot)^2 )
  ElecEner = {n_e:ptr_new(n_ehot), t_e:ptr_new(t_ehot)}
endif

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; State of the Thermal ions
if ((kappa.chx) and (plasma_info.thermal)) then begin
  nion = n_elements(kappa.ions)
  ThermDen = fltarr(num, nion)
  ThermH = fltarr(num, nion)
  for i=0,nion-1 do begin
    ;; For each ion in the rate coefficient, determine the ion density and scale height
    q = (where(*plasma.ions EQ (kappa.ions)[i]))[0]
    if (q NE -1) then begin
      ThermDen[*,i] = interpol((*plasma.n_i)[*,q], *plasma.L, M)
      ThermH[*,i] = interpol((*plasma.h_i)[*,q], *plasma.L, M)
    endif
  endfor
  hq = where(ThermDen LE 0, hct) & if (hct NE 0) then ThermDen[hq] = 0
  hq = where(ThermH LE 0.1, hct) & if (hct NE 0) then ThermH[hq] = 0.1

  ii = replicate(1., nion)
  ThermDen = ThermDen * exp(-((zeta#ii)/ThermH)^2)
  ThermT = ThermH*0.  ;; Currently do not include effects of ion thermal motion

  IonTherm = {n_i:ptr_new(ThermDen), t_i:ptr_new(ThermT)}
endif

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; State of the Energetic ions
;; Don't include charge exchange with hot ions
;;if ((chx) and (plasma_info.energetic)) then begin
;;  IonEner = {n_i:ptr_new(0), t_i:ptr_new(0)}
;;endif
```

```
103
104  end
105
```