```
p o planet_dist, taa, SystemConsts, distance=distance, velocity=velocity

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Given a true anomaly angle, determine the distance and radial velocity of the planet
;; relative to the sun.
;;
;; Outputs: distance (AU) and radial velocity (km/s)
;;
;;  Version history
;;    2.0: 10/22/08 (MHB)
;;      * Computes distance and radial velocity for Mercury.
;;      * For other planets, just assumes circular orbits with dr/dt = 0.
;;      * Should change this, but would require a taa for all models
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

if (n_params() EQ 0) then begin
  print, 'planet_dat, taa, SystemConsts, distance=distance, velocity=velocity'
  return
endif

a = SystemConsts.aplan  ;AU             semi-major axis
eps = (SystemConsts.planet EQ 'Mercury') ? SystemConsts.epsplan : 0.

if (SystemConsts.planet EQ 'Mercury') then begin
  time = findgen(1001)/1000. ;; time = t/P

  ;; Mean anomaly
  M = (2*!pi*time) mod (2*!pi)

  ;; eccentric anomaly
  EEtemp = findgen(1001)/1000*2*!pi

  mm = EEtemp - eps*sin(EEtemp)
  EE = fltarr(n_elements(time))
  err = fltarr(n_elements(time))
  for i=0,n_elements(EE)-1 do ee[i] = interpol(eetemp, mm-m[i], 0)

  ;; true anomaly
  phi = (2*atan(sqrt((1+eps)/(1-eps)) * tan(EE/2)) + (2*!pi)) mod (2*!pi)
  r = a * (1-eps^2)/(1+eps*cos(phi))

  P = sqrt(a^3)
  drdt = deriv(time*!const.sec_year*0.241, r*!const.au/1e5)  ;; km/s

  distance = interpol(r, phi, taa)
  velocity = interpol(drdt, phi, taa)
endif else begin
  distance = a
  velocity = 0.
```

```
52  endelse
53
54  end
```