```
function inputs_restore, file, geometry=geometry, sticking_info=sticking_info, $
  forces=forces, spatialdist=spatialdist, angulardist=angulardist, speeddist=speeddist, $
  perturbvel=perturbvel, options=options, plasma_info=plasma_info

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;  Revision History
;;    3.3: 7/7/11
;;     * Added PSD spatial distribution option
;;    3.2: 11/23/10
;;     * added size option to SO2 exosphere spatial dist
;;    3.0: 7/13/10
;;     * Put all the inputs into a single structure that is the output of the function
;;
;;    2.10 4/26/10
;;     * Added Earth/Moon options
;;    2.9  3/9/10
;;     * removed partial thermal accommodation option and added accom_factor to
;;        Maxwellian sticking function
;;    2.8: 1/13/10
;;      * added local temperature option to maxwellian speed ditsribution
;;         -- distribution fn depends on local surface temperature
;;      * added options.datapath and options.modelpath
;;    2.7: 11/20/09
;;      * added powerlaw/exponential options to exosphere spatial distribution
;;    2.6: 11/6/09
;;      * added Sticking_info.accom_factor option
;;    2.3: 2/12/09
;;      * added parameters for SpatialDist.type = 'exosphere'
;;          * SpatialDist.b -- powerlaw slope of density in exospehre
;;    2.2:
;;    2.1: 12/19/08
;;      * added fields geometry.subSolarLong and subSolarLat
;;      * These are currently used in determining the observation geometry
;;    2.0:
;;      * 12 November 2008: remade with new structure information
;;      * need to add in input validation to make sure there are no
;;          contradictions - for example - if using circular orbits, need a
;;          perturbation distribution but no angular distribution
;;      * need to check capitalization, etc.
;;    1.0: original
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

readcol, file, param, value, delim='=', format='A,A', /silent
param = strlowcase(strtrim(param, 2))

;; strip off any comments in the values
q = stregex(value, ';')
w = where(q NE -1, nq)
if (nq GT 0) then for i=0,nq-1 do $
```

```
    value[w[i]] = strmid(value[w[i]], 0, q[w[i]]-1)
value = strtrim(value, 2)

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Make the geometry structure
;; Fields:
;;   planet -- required
;;   StartPoint -- required
;;   phi -- required for Jupiter and Saturn with one for each object
;;       -- phi[0] = 0 always
;;       -- for Mercury, the input is ignored and phi set to 0.
;;   include -- optional with one for each object
;;           -- if not provided set to 1 for each object
;;           -- if included, must have correct number (Jup=5, Sat=10, Merc=1)
;;   CML -- required for Jupiter only
;;   taa -- required for Mercury only
;;   SubSolarLong -- default = 0 deg.
;;   SubSolLat -- default = 0 deg.
;;   aplanet ---| these are added to the structure but default value is deteremined
;;   vrplanet --| in modeldriver.
geom = where(strmatch(param, 'geometry*'))
gparam = strmid(param[geom], strlen('geometry.'))
gval = value[geom]

q = (where(gparam EQ 'planet'))[0]
planet = gval[q]

q = (where(gparam EQ 'subsolarlong', nq))[0]
subslong = (nq EQ 1) ? gval[q] : 0.

q = (where(gparam EQ 'subsolarlat', nq))[0]
subslat = (nq EQ 1) ? gval[q] : 0.

case (planet) of
  'Mercury': begin
    q = (where(gparam EQ 'taa', nq))[0]
    if (nq NE 1) then stop else taa = double(gval[q])
    q = (where(gparam EQ 'include', nq))[0]
    case (nq) of
      0: inc = 1
      1: inc = fix(gval[q]) NE 0
      else: stop
    endcase

    geometry = {planet:'Mercury', StartPoint:'Mercury', phi:ptr_new(0d), $
      include:ptr_new(inc), taa:taa, subsolarlong:subslong, subsolarlat:subslat}
    end
  'Earth': begin
    q = (where(gparam EQ 'startpoint', nq))[0]
    stpt = (nq EQ 1) ? gval[q] : stop
    q = where(gparam EQ 'phi', nq)
```

```
103       case (nq) of
104         1: phi = double([0., gval[q]])
105         2: phi = double(gval[q])
106         else: stop
107       endcase
108       q = where(gparam EQ 'include', nq)
109       case (nq) of
110         0: inc = [1,1]
111         2: inc = fix(gval[q]) NE 0
112         else: stop
113       endcase
114       geometry = {planet:'Earth', StartPoint:stpt, phi:ptr_new(phi), $
115         include:ptr_new(inc), subsolarlong:subslong, subsolarlat:subslat, taa:0.}
116       end
117     'Jupiter': begin
118       q = (where(gparam EQ 'startpoint'))[0]
119       stpt = gval[q]
120       q = where(gparam EQ 'phi', nq)
121       if (nq NE 5) then stop else phi = double(gval[q]) & phi[0] = 0d
122       q = (where(gparam EQ 'cml', nq))[0]
123       if (nq NE 1) then stop else cml = double(gval[q])
124
125       q = where(gparam EQ 'include', nq)
126       case (nq) of
127         0: inc = replicate(1, 5)
128         5: inc = fix(gval[q]) NE 0
129         else: stop
130       endcase
131
132       geometry = {planet:'Jupiter', StartPoint:stpt, phi:ptr_new(phi), $
133         include:ptr_new(inc), CML:CML, subsolarlong:subslong, $
134         subsolarlat:subslat, taa:0.}
135       end
136     'Saturn': begin
137       q = (where(gparam EQ 'startpoint'))[0]
138       stpt = gval[q]
139       q = where(gparam EQ 'phi', nq)
140       if (nq NE 10) then stop else phi = double(gval[q]) & phi[0] = 0d
141       q = where(gparam EQ 'include', nq)
142       case (nq) of
143         0: inc = replicate(1, 10)
144         10: inc = fix(gval[q]) NE 0
145         else: stop
146       endcase
147
148       geometry = {planet:'Saturn', StartPoint:stpt, phi:ptr_new(phi), $
149         include:ptr_new(inc), subsolarlong:subslong, subsolarlat:subslat, taa:0.}
150       end
151       else: stop
152   endcase
153
```

```
154  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
155  ;; Sticking_info
156  ;; This is optional - if not there, assumed that everything sticks to the surface
157  ;; fields:
158  ;;   * stickcoef  -- if stickcoef = 1. then there are no other options
159  ;;   * emitfn -- options = 'Maxwellian', 'elastic scattering'
160  ;;   * surftemp -- if emitfn = 'Maxwellian'
161  ;; -- will need to expand on this
162
163  st = where(strmatch(param, 'sticking_info*'), ns)
164  if (ns EQ 0) $
165    then stop $
166    else begin
167      sparam = strmid(param[st], strlen('sticking_info.'))
168      sval = value[st]
169
170      q = (where(sparam EQ 'stickcoef', nq))[0]
171      if (nq NE 1) then stop else stick = float(sval[q])
172      if (stick GE 1) $
173        then sticking_info = {stickcoef:1.} $
174        else begin
175          q = (where(sparam EQ 'emitfn'))[0]
176          fn = sval[q]
177          case strlowcase(fn) of
178            'maxwellian': begin
179              q = (where(sparam EQ 'tsurf', nq))[0]
180              if (nq EQ 1) then Tsurf = max([0d, double(sval[q])]) else Tsurf = 0
181
182              q = (where(sparam EQ 'accom_factor', nq))[0]
183              if (nq EQ 1) then accom_factor = double(sval[q]) else stop
184              if (accom_factor LT 0) then accom_factor = 0d
185              if (accom_factor GT 1) then accom_factor = 1d
186
187              sticking_info = {stickcoef:stick, emitfn:fn, Tsurf:tsurf, $
188                accom_factor:accom_factor}
189              end
190            'elastic scattering': sticking_info = {stickcoef:stick, emitfn:fn}
191            else: stop ;; not set up yet
192          endcase
193        endelse
194    endelse
195
196  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
197  ;;  Forces
198  ;;  Any force not explicitly set is turned off
199  ;;  options:
200  ;;    * gravity
201  ;;    * radpres
202  ;;    * lorentz
203  ;;
204  forces = {gravity:0, radpres:0, lorentz:0}
```

```
205  ff = where(strmatch(param, 'forces*'), ns)
206  if (ns NE 0) then begin
207    fparam = strmid(param[ff], strlen('forces.'))
208    fval = value[ff]
209    q = (where(fparam EQ 'gravity', nq))[0]
210    if (nq EQ 1) then forces.gravity = fix(fval(q))
211    q = (where(fparam EQ 'radpres', nq))[0]
212    if (nq EQ 1) then forces.radpres= fix(fval(q))
213    q = (where(fparam EQ 'lorentz', nq))[0]
214    if (nq EQ 1) then forces.lorentz= fix(fval(q))
215  endif
216
217  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
218  ;; SpatialDist
219  ;;   * surface -- default is evenly spread out over sphere with radius=1
220  ;;   * torus -- no default - r0,r1,r2 must be specified
221  spat = where(strmatch(param, 'spatialdist*'), ns)
222  sparam = strmid(param[spat], strlen('spatialdist.'))
223  sval = value[spat]
224  q = (where(sparam EQ 'type'))[0]
225  spatdist = sval[q]
226
227  case strlowcase(spatdist) of
228    'surface': begin
229      q = (where(sparam EQ 'use_map', nq))[0]
230      usemap = (nq EQ 1) ? fix(sval[q]) : 0
231
232      q = (where(sparam EQ 'exobase', nq))[0]
233      exobase = (nq EQ 1) ? double(sval[q]) : 1d
234
235      if (usemap) then begin
236        q = (where(sparam EQ 'mapfile'))[0]
237        mapfile = sval[q]
238        SpatialDist = {type:'surface', exobase:exobase, use_map:1, $
239          mapfile:mapfile}
240      endif else begin
241        q = where(sparam EQ 'longitude0', nq)
242        lon0 = (nq EQ 1) ? double(sval[q]) : 0d
243        q = where(sparam EQ 'longitude1', nq)
244        lon1 = (nq EQ 1) ? double(sval[q]) : 2*!dpi
245
246        q = where(sparam EQ 'latitude0', nq)
247        lat0 = (nq EQ 1) ? double(sval[q]) : -!dpi/2.
248        q = where(sparam EQ 'latitude1', nq)
249        lat1 = (nq EQ 1) ? double(sval[q]) : !dpi/2.
250
251        SpatialDist = {type:'surface', exobase:exobase, use_map:0, $
252          longitude:[lon0,lon1], latitude:[lat0,lat1]}
253      endelse
254      end
255    'torus': begin
```

```
256       q = (where(sparam EQ 'torus_radius0', nq))[0]
257       if (nq EQ 1) then r0 = double(sval[q]) else stop
258       q = (where(sparam EQ 'torus_radius1', nq))[0]
259       if (nq EQ 1) then r1 = double(sval[q]) else stop
260       q = (where(sparam EQ 'torus_radius2', nq))[0]
261       if (nq EQ 1) then r2 = double(sval[q]) else stop
262
263       SpatialDist = {type:'torus', torus_radii:[r0, r1, r2]}
264       end
265     'exosphere': begin
266       q= (where(sparam EQ 'exotype', nq))[0]
267       if (nq EQ 1) then exotype = sval[q] else stop
268       q = (where(sparam EQ 'b', nq))[0]
269       if (nq EQ 1) then b = float(sval[q]) else stop
270       q = (where(sparam EQ 'rmax', nq))[0]
271       rmax = (nq EQ 1) ? sval[q] : 10.
272       q = (where(sparam EQ 'block_shadow', nq))[0]
273       block_shadow = (nq EQ 1) ? fix(sval[q]) : 0
274
275       SpatialDist = {type:'exosphere', exotype:exotype, b:b, rmax:rmax, $
276         block_shadow:block_shadow}
277       end
278     'so2 exosphere': begin
279       q = (where(sparam EQ 'size', nq))[0]
280       case (1) of
281         stregex(sval[q], 'large', /fold, /bool): size = 'large'
282         stregex(sval[q], 'small', /fold, /bool): size = 'small'
283         else: stop
284       endcase
285       SpatialDist = {type:'SO2 exosphere', size:size}
286       end
287     'psd': begin
288       q = (where(sparam EQ 'exobase', nq))[0]
289       exobase = (nq EQ 1) ? double(sval[q]) : 1d
290
291       q = (where(sparam EQ 'diffusionlimit', nq))[0]
292       dlimit = (nq EQ 1) ? double(sval[q]) : 1d30 ; default = unlimited
293
294       q = (where(sparam EQ 'kappa', nq))[0]
295       kappa = (nq EQ 1) ? double(sval[q]) : 0d
296
297       if (kappa GT 0) then begin
298         q = (where(sparam EQ 'protonprecipfile', nq))[0]
299         if (nq EQ 1) then ff = sval[q] else stop
300       endif else ff = ''
301
302       SpatialDist = {type:'PSD', diffusionlimit:dlimit, kappa:kappa, $
303         ProtonPrecipFile:ff, exobase:exobase}
304       end
305     else: stop
306   endcase
```

```
307
308  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
309  ;;  VelocityDist: SpeedDist and AnglularDist
310  ;;
311  vdist = where(strmatch(param, 'speeddist*'))
312  vparam = strmid(param[vdist], strlen('speeddist.'))
313  vval = value[vdist]
314
315  q = (where(vparam EQ 'type'))[0]
316  spd = strlowcase(vval[q])
317
318  case (spd) of
319    'gaussian': begin   ;; vprob, sigma
320      q = (where(vparam EQ 'vprob'))[0]
321      vprob = double(vval[q])
322      q = (where(vparam EQ 'sigma'))[0]
323      sigma = double(vval[q])
324      speeddist = {type:'gaussian', vprob:vprob, sigma:sigma}
325      end
326    'trigaussian': begin ;; vxprob, vxsigma, vyprob, vysigma, vzprob, vzsigma
327      q = (where(vparam EQ 'vxprob'))[0]
328      vxprob = double(vval[q])
329      q = (where(vparam EQ 'vxsigma'))[0]
330      vxsigma = double(vval[q])
331
332      q = (where(vparam EQ 'vyprob'))[0]
333      vyprob = double(vval[q])
334      q = (where(vparam EQ 'vysigma'))[0]
335      vysigma = double(vval[q])
336
337      q = (where(vparam EQ 'vzprob'))[0]
338      vzprob = double(vval[q])
339      q = (where(vparam EQ 'vzsigma'))[0]
340      vzsigma = double(vval[q])
341
342      speeddist = {type:'trigaussian', vxprob:vxprob, vxsigma:vxsigma, vyprob:vyprob, $
343        vysigma:vysigma, vzprob:vzprob, vzsigma:vzsigma}
344      end
345    'dolsfunction': begin  ;; dols0, dols1
346      q = (where(vparam EQ 'dols0'))[0]
347      dols0 = vval[q]
348      q =  (where(vparam EQ 'dols1'))[0]
349      dols1 = vval[q]
350      speeddist = {type:'dolsfunction', dols0:dols0, dols1:dols1}
351      end
352    'sputtering': begin  ;; U, alpha, beta
353      q = (where(vparam EQ 'u'))[0]
354      U = double(vval[q])
355      q = (where(vparam EQ 'alpha'))[0]
356      alpha = double(vval[q])
357      q = (where(vparam EQ 'beta'))[0]
```

```
358       beta = double(vval[q])
359       speeddist = {type:'sputtering', U:U, alpha:alpha, beta:beta}
360       end
361     'maxwellian': begin ;; temperature
362       q = (where(vparam EQ 'temperature'))[0]
363       temp = double(vval[q])
364       speeddist = {type:'maxwellian', temperature:temp}
365       end
366     'flat': begin  ;; vprob, delv
367       q = (where(vparam EQ 'vprob'))[0]
368       vprob = double(vval[q])
369       q = (where(vparam EQ 'delv'))[0]
370       delv = double(vval[q])
371       speeddist = {type:'flat', vprob:vprob, delv:delv}
372       end
373     'circular orbits': speeddist = {type:'circular orbits'}  ;; no options
374     'user defined': begin
375       q = (where(vparam EQ 'distfile', nq))[0]
376       if (nq NE 0) then distfile = vval[q] else stop
377       speeddist = {type:'user defined', distfile:distfile}
378       end
379     else: stop
380   endcase
381
382   vdist = where(strmatch(param, 'angulardist*'))
383   vparam = strmid(param[vdist], strlen('angulardist.'))
384   vval = value[vdist]
385
386   q = (where(vparam EQ 'type'))[0]
387   ang = strlowcase(vval[q])
388   if (SpeedDist.type EQ 'circular orbits') then ang = 'none'
389
390   case (ang) of  ;; none, radial, isotropic, costheta
391     'none': angulardist = {type:'none'}
392     'radial': angulardist = {type:'radial'}
393     'isotropic': begin
394       ;; For distributions starting at the surface, make sure the packets are pointed
395       ;; outward
396       case (SpatialDist.type) of
397         'surface': altmin = 0.
398         'torus': altimin = -!dpi/2.
399         'exosphere': altmin = -!dpi/2.
400         'SO2 exosphere': altmin = -!dpi/2.
401         else: stop
402       endcase
403
404       q = where(vparam EQ 'azimuth0', nq)
405       az0 = (nq EQ 1) ? double(vval[q]) : 0d
406       q = where(vparam EQ 'azimuth1', nq)
407       az1 = (nq EQ 1) ? double(vval[q]) : 2*!dpi
408
```

```
      q = where(vparam EQ 'altitude0', nq)
      alt0 = (nq EQ 1) ? double(vval[q]) : altmin
      q = where(vparam EQ 'altitude1', nq)
      alt1 = (nq EQ 1) ? double(vval[q]) : !dpi/2.

      angulardist = {type:'isotropic', azimuth:[az0, az1], altitude:[alt0, alt1]}
      end
    'costheta': begin
      q = where(vparam EQ 'azimuth0', nq)
      az0 = (nq EQ 1) ? double(vval[q]) : 0d
      q = where(vparam EQ 'azimuth1', nq)
      az1 = (nq EQ 1) ? double(vval[q]) : 2*!dpi

      q = where(vparam EQ 'altitude0', nq)
      alt0 = (nq EQ 1) ? double(vval[q]) : 0.
      q = where(vparam EQ 'altitude1', nq)
      alt1 = (nq EQ 1) ? double(vval[q]) : !dpi/2.

      q = (where(vparam EQ 'n', nq))[0]
      n = (nq EQ 1) ? double(vval[q]): 1.

      angulardist = {type:'costheta', azimuth:[az0, az1], altitude:[alt0, alt1], n:n}
      end
  endcase

  ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
  ;; PerturbVel -- not set up
  pdist = where(strmatch(param, 'perturbvel*'), npert)
  if (npert NE 0) then begin
    pparam = strmid(param[pdist], strlen('perturbvel.'))
    pval = value[pdist]

    q = (where(pparam EQ 'type'))[0]
    type = pval[q]

    case (type) of
      'none': PerturbVel = {type:'none'}
      'gaussian': begin
        q = (where(pparam EQ 'vprob'))[0]
        vprob = pval[q]
        q = (where(pparam EQ 'sigma'))[0]
        sigma = pval[q]

        PerturbVel = {type:'gaussian', vprob:vprob, sigma:sigma}
        end
      'trigaussian': begin ;; vxprob, vxsigma, vyprob, vysigma, vzprob, vzsigma
        q = (where(pparam EQ 'vxprob'))[0]
        vxprob = double(pval[q])
        q = (where(pparam EQ 'vxsigma'))[0]
        vxsigma = double(pval[q])

```

```
460         q = (where(pparam EQ 'vyprob'))[0]
461         vyprob = double(pval[q])
462         q = (where(pparam EQ 'vysigma'))[0]
463         vysigma = double(pval[q])
464
465         q = (where(pparam EQ 'vzprob'))[0]
466         vzprob = double(pval[q])
467         q = (where(pparam EQ 'vzsigma'))[0]
468         vzsigma = double(pval[q])
469
470         PerturbVel = {type:'trigaussian', vxprob:vxprob, vxsigma:vxsigma, vyprob:vyprob, $
471           vysigma:vysigma, vzprob:vzprob, vzsigma:vzsigma}
472         end
473       'charge exchange': begin ;; flowvel
474         q = (where(pparam EQ 'flowvel'))[0]
475         flowvel = double(pval[q])
476         PerturbVel = {type:'charge exchange', flowvel:flowvel}
477         end
478       else: stop
479     endcase
480   endif else PerturbVel = {type:'none'}
481
482   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
483   ;; Plasma_info
484   case (planet) of
485     'Mercury': plasma_info = {type:'none'}
486     'Earth': plasma_info = {type:'none'}
487     'Jupiter': begin
488       ;; These are the current defaults, but need to review this.
489       ;; plamsa parameters:
490       ;;   eps - default = 0.14/5.7
491       ;;   thermal: default = 1
492       ;;   energetic: default = 1
493       pdist = where(strmatch(param, 'plasma*'))
494       pparam = strmid(param[pdist], strlen('plasma.'))
495       pval = value[pdist]
496
497       q = (where(pparam EQ 'eps', nq))[0]
498       eps = (nq EQ 1) ? double(pval[q]) : 0.14/5.7
499
500       q = (where(pparam EQ 'thermal', nq))[0]
501       th = (nq EQ 1) ? fix(pval[q]) : 1
502       q = (where(pparam EQ 'energetic', nq))[0]
503       en = (nq EQ 1) ? fix(pval[q]) : 1
504
505       plasma_info = {eps:eps, thermal:th, energetic:en}
506       end
507     'Saturn': begin
508       ;;   ElecDenMod: default = 1
509       ;;   ElecTempMod: default = 1
510       pdist = where(strmatch(param, 'plasma*'),np)
```

```
511       if (np NE 0) then begin
512         pparam = strmid(param[pdist], strlen('plasma.'))
513         pval = value[pdist]
514       endif else begin
515         pparam = ''
516         pval = 0.
517       endelse
518
519       q = (where(pparam EQ 'elecdenmod', nq))[0]
520       denmod = (nq EQ 1) ? double(pval[q]) : 1d
521
522       q = (where(pparam EQ 'electempmod', nq))[0]
523       tmod = (nq EQ 1) ? double(pval[q]) : 1d
524
525       plasma_info = {ElecDenMod:denmod, ElecTempMod:tmod}
526       end
527   endcase
528
529   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
530   ;; Options
531   ;;   * endtime
532   ;;   * resolution - default = 1e-6
533   ;;   * motion - default = 1 for Jupiter and Saturn, no effect for Mercury
534   ;;   * lifetime - default = 0
535   ;;   * atom
536   ;;   * at_once - default = 0 but reset when doing streamlines
537   ;;   * fullsystem - default = 1 for Jupiter and Saturn, default=0 for Mercury
538   ;;   * outeredge - if fullsystem set to 0, default = 20
539   ;;
540
541   odist = where(strmatch(param, 'options*'))
542   oparam = strmid(param[odist], strlen('options.'))
543   oval = value[odist]
544
545   q = (where(oparam EQ 'endtime', nq))[0]
546   if (nq NE 1) then stop else endtime = double(oval[q])
547
548   q = (where(oparam EQ 'resolution', nq))[0]
549   res = (nq EQ 1) ? double(oval[q]) : 1d-6
550
551   q = (where(oparam EQ 'at_once', nq))[0]
552   at = (nq EQ 1) ? fix(oval[q]) : 0
553
554   q = (where(oparam EQ 'atom', nq))[0]
555   if (nq NE 1) then stop else atom = oval[q]
556
557   q = (where(oparam EQ 'lifetime', nq))[0]
558   life = (nq EQ 1) ? double(oval[q]) : 0d
559
560   f = (where(oparam EQ 'fullsystem', nf))[0]
561   if (planet EQ 'Mercury') then begin
```

```
562     full = (nf EQ 1) ? fix(oval[f]) : 0
563     m = 0
564   endif else begin
565     full = (nf EQ 1) ? fix(oval[f]) : 1
566
567     q = (where(oparam EQ 'motion', nq))[0]
568     m = (nq EQ 1) ? fix(oval[q]) : 1
569   endelse
570
571   if ~(full) then begin
572     q = (where((oparam EQ 'outeredge'), nq))[0]
573     outer = (nq EQ 1) ? double(oval[q]) : 20d
574   endif else outer = 0.
575
576   q = (where(oparam EQ 'trackloss', nq))[0]
577   trackloss = (nq EQ 1) ? fix(oval[q]) : 0
578
579   options = {endtime:endtime, resolution:res, motion:m, lifetime:life, $
580     atom:atom, at_once:at, fullsystem:full, outeredge:outer, trackloss:trackloss}
581
582   ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
583   ;; Put all the inputs into a single structure
584   input = {geometry:geometry, sticking_info:sticking_info, forces:forces, $
585     spatialdist:spatialdist, angulardist:angulardist, speeddist:speeddist, $
586     options:options, perturbvel:perturbvel, plasma_info:plasma_info}
587
588   return, input
589
590   end
```