```
function read_resultformat, formatfile

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Read in the result format file.
;;
;; Version History:
;;   4.2 1 Dec 2011
;;     * A few updates
;;   4.1: 24 Oct 2011
;;     * Reworking this
;;   4.0: 25 Jan 2011
;;     * Original
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

readcol, formatfile, 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 format structure
form = where(strmatch(param, 'format*'))
fparam = strmid(param[form], strlen('format.'))
fval = value[form]

q = (where(fparam EQ 'type', nq))[0]
if (nq EQ 1) then type = fval[q] else stop

q = (where(fparam EQ 'quantity', nq))[0]
if (nq EQ 1) then quantity = fval[q] else stop

q = (where(fparam EQ 'strength', nq))[0]
strength = (nq EQ 1) ? double(fval[q]) : 1.

;; Test these:
if ((type NE 'image') and (type NE 'voronoi image') and (type NE 'los') and $
  (type NE 'points')) then begin
    print, 'Not a valid result type.'
    print, 'Valid options are: image, voronoi, los, points'
    stop
endif

if ((quantity NE 'column') and (quantity NE 'intensity') and (quantity NE 'density')) $
  then begin
```

```
      print, 'Not a valid result quantity.'
      print, 'Valid options are: column, intensity, density.'
      stop
    endif

  if (strength LE 0) then begin
    print, 'Strength must be >0.'
    stop
  endif

  ;;;;;;;;;;;;;;;;;
  ;; Make the geometry structure
  geo = where(strmatch(param, 'geometry*'))
  gparam = strmid(param[geo], strlen('geometry.'))
  gval = value[geo]

  q = (where(gparam EQ 'origin', nq))[0]
  if (nq EQ 1) then origin = gval[q] else stop

  case (1) of
    (type EQ 'image') or (type EQ 'voronoi image'): begin
      q = (where(gparam EQ 'dims', nq))[0]
      if (nq EQ 1) then begin
        dims = strcompress(gval[q], /remove_all)
        dims = fix(strsplit(dims, ',', /extract))
      endif else stop

      q = (where(gparam EQ 'center', nq))[0]
      if (nq EQ 1) then begin
        center = strcompress(gval[q], /remove_all)
        center = float(strsplit(center, ',', /extract))
      endif else stop

      q = (where(gparam EQ 'width', nq))[0]
      if (nq EQ 1) then begin
        width = strcompress(gval[q], /remove_all)
        width = float(strsplit(width, ',', /extract))
      endif else stop

      q = (where(gparam EQ 'subobslongitude', nq))[0]
      if (nq EQ 1) then subobslong = float(gval[q]) else stop
      if ((subobslong LT 0) or (subobslong GT 2*!dpi)) then begin
        print, 'Sub-Observer Longitude must be between 0 and 2¿'
        stop
      endif

      q = (where(gparam EQ 'subobslatitude', nq))[0]
      if (nq EQ 1) then subobslat = float(gval[q]) else stop
      if ((subobslat LT -!dpi/2) or (subobslat GT !dpi/2)) then begin
        print, 'Sub-Observer Latitude must be between -¿/2 and ¿/2'
        stop
  endif

  q = (where(gparam EQ 'polarangle', nq))[0]
  if (nq EQ 1) then polarangle = float(gval[q]) else stop
  if ((polarangle LT 0) or (polarangle GT 2*!dpi)) then begin
    print, 'Polar angle must be between 0 and 2¿'
    stop
  endif

  geometry = {origin:origin, dims:dims, center:center, width:width, $
    subobslongitude:subobslong, subobslatitude:subobslat, $
    polarangle:polarangle}
  end
(type EQ 'los') or (type EQ 'density'): begin
  ;; Note: dr can be either in format or geometry part
  q = (where(fparam EQ 'dr', nq))[0]
  if (nq EQ 1) $
    then dr = double(fval[q]) $
    else begin
      q = (where(gparam EQ 'dr', nq))[0]
      if (nq EQ 1) then dr = double(gval[q]) else stop
    endelse

  q = (where(gparam EQ 'usedata', nq))[0]
  usedata = (nq EQ 1) ? fix(gval[q]) : 1

  if (usedata) then begin
    q = (where(gparam EQ 'spacecraft', nq))[0]
    spacecraft = gval[q]

    if (type EQ 'density') then begin
      q = (where(gparam EQ 'dt', nq))[0]
      dt = (nq EQ 1) ? double(gval[q]) : 0.
    endif else dt = 0.

    q = (where(gparam EQ 'orbit', nq))[0]
    case (nq) of
      0: begin  ;; tstart, tend specified
         q = (where(gparam EQ 'tstart', nq))[0]
         if (nq EQ 1) then tstart = gval[q] else stop

         q = (where(gparam EQ 'tend', nq))[0]
         if (nq EQ 1) then tend= gval[q] else stop
         geometry = {origin:origin, dr:dr, spacecraft:spacecraft, usedata:usedata, $
           tstart:tstart, tend:tend, usedata:usedata, dt:dt}
        end
      1: begin  ;; orbit # specified
         orbit = fix(gval[q])
         geometry = {origin:origin, dr:dr, spacecraft:spacecraft, orbit:orbit, $
           usedata:usedata, dt:dt}
         end
        else: stop
      endcase
    endif
    end
  else: stop ;; problem
endcase

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Make the emission structure if necessary
if (quantity EQ 'intensity') then begin
  emi = where(strmatch(param, 'emission*'))
  eparam = strmid(param[emi], strlen('emission.'))
  eval = value[emi]

  q = (where(eparam EQ 'mechanism', nq))[0]
  if (nq EQ 1) then mech = eval[q] else stop
  mech = strsplit(mech, ',', /extract)
  if (n_elements(mech) EQ 1) then mech = mech[0]

  q = (where(eparam EQ 'line', nq))[0]
  if (nq EQ 1) then line = eval[q] else stop
  line = float(strsplit(line, ',', /extract))
  if (n_elements(line) EQ 1) then line = line[0]

  emission = {mechanism:mech, line:line}
endif else emission = !null

format = {type:type, quantity:quantity, strength:strength, geometry:geometry, $
  emission:emission}

return, format

end
```