```
function get_gvalue, atom, a, path=path

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; Looks up the gvalues given by Killen et al 2008. The function returns a
;; 2xn array with the velocities and the radiation pressure constant for the
;; species. Keywords can be used to get the g values for each line  which may
;; be used to calculate the emission.
;;
;; INPUTS:
;;    * a = distance from the sun (AU) -- must be a single value, not an array
;;    * atom = atom for which to look up g-values
;;
;; OUTPUTS
;;    * Function returns 2xn array with velocities and radiation pressure constant.
;;        units = km/s^2
;;    * lines = resonance transitions included
;;    * velocity = radial velocities (km/s)
;;    * gval = array containing g-value vs. velocity for each transition
;;
;;  Version History
;;    3.1: 10 May 2011
;;       * New way of saving g-values. Use set_up_gvals to save into structures
;;    2.1: 30 Jan 2009
;;      - added g-values for all species included in Killen et al. [C I, Ca I, Ca II,
;;         H I, He I, K I, Mg I, Mg II, Na I, OH, O I, S I
;;    2.0: original -- only looks up Na g values
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

if (n_elements(a) GT 1) then stop
if (a EQ !null) then a = 1.
if (n_elements(path) EQ 0) then $
  path = !model.basepath + 'Work/AtomicData/g-values/' + atom
if ~(file_test(path)) then stop

files = file_search(path, '*.sav') & nf = n_elements(files)

gval = {species:atom, $
        a:a, $                         ;; AU
        wavelength:ptr_new(0), $       ;; Angstrom
        v:ptr_new(0), $                ;; km/s
        g:ptr_new(0), $                ;; photons cm^-2 s^-1
        radaccel:ptr_new(0)}           ;; km^-2 s^-1
case (nf) of
  0: begin
     *gval.v = [0., 1.]
     *gval.g = [0., 0.]
     print, 'g-values not found. Radiation acceleration = 0'
     end
  1: begin
```

```
     restore, files[0]
     *gval.wavelength = gvalue.wavelength
     *gval.v = *gvalue.v
     *gval.g = *gvalue.g * gvalue.a^2/a^2 ;; normalize to specified distance
     end
  else: begin
     lambda = fltarr(nf)
     vv = ptrarr(nf, /allocate)
     gg = ptrarr(nf, /allocate)
     for i=0,nf-1 do begin
       restore, files[i]
       lambda[i] = gvalue.wavelength
       *vv[i] = *gvalue.v
       *gg[i] = *gvalue.g * gvalue.a^2/a^2
     endfor

     ;; Test if all wavelengths are unique
     u = uniq(lambda, sort(lambda))
     if (n_elements(u) NE nf) then begin
       ;; Need to decide which to use
       stop
     endif
     *gval.wavelength = lambda

     ;; Get a common velocity axis
     allv = !null
     for i=0,nf-1 do allv = [allv, *vv[i]]
     allv = allv[sort(allv)]
     *gval.v = allv[uniq(allv)]
     *gval.g = fltarr(n_elements(*gval.v),nf)
     for i=0,nf-1 do (*gval.g)[*,i] = interpol(*gg[i], *vv[i], *gval.v)
     end
endcase

;; radpres_const = h/(m*lambda) * g
rr = !const.h / atomicmass(atom) / (*gval.wavelength*1e-8)
qq = 0.
for i=0,nf-1 do qq += rr[i]*(*gval.g)[*,i]*1e-5  ;; km s^-2
*gval.radaccel = qq

return, gval

end
```