```
pro modeldriver, inputfiles, npackets, seed, $
  quick=quick, overwrite=overwrite, generic=generic, local=local, $
  logfile=logfile, outputfile=outputfile, runmodel=runmodel

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;  Options
;;     * quick = 1/0 -- if running in quickmode, just does one run through of
;;         npackets to test a set of options. Otherwise, does the full check of
;;         what is already run. Quickmode implies overwrite=1 and generic=0
;;     * generic = 0/1/2
;;          = 0 => does not search through the existing generic files (runs
;;                 exact inputs specified)
;;          = 1 => Extracts specific inputs from generic files and runs more
;;                 generic files if needed.
;;          = 2 => Does not extract specific inputs and checks to see if there
;;                 are enough generic packets available. This is needed to prevent
;;                 extracting inputs when you want to run generic models only
;;
;;  Revision history
;;    3.8 -- 10/8/2011
;;      * moving the part where it searches for packets and extracts from generic files
;;        to a model_findpackets.pro
;;    3.7 -- 9/28/2011
;;      * Adding option to save models locally
;;    3.6 -- 9/16/2011
;;      * Changing the path method - using !model.basepath
;;    3.5 -- 5/10/2011
;;      * Changing the way the g-value is stored
;;    3.4 -- 12/17/2010
;;      * adding more options to make quick runs and streamlines easier
;;    3.3 -- 9/8/10
;;      * If generic=0, does not search through the existing generic files
;;    3.2 -- 8/31/10
;;      * added packets keyword to modeloutput_search
;;    3.0 -- 7/12/10
;;      * user no longer controls name of output file
;;      * want to look to see if the appropriate model already exists
;;      * need to move the streamline routines to separate programs
;;      * removing minimization option
;;    2.5 -- 4/19/10
;;      * added version to output
;;    2.4 -- replace destroy_XXX with destroy_structure
;;    2.3 -- added an option to minimize the structures to save memory
;;        -- Removed to_run option since only one integrator still works (5/27/09)
;;    2.2 -- added ability to use g-values from Killen et al. This will make it possible
;;      to compute radiation pressure for a number of species.
;;    2.1 -- revise modstream 1 to do everything all at once (if there aren't too
;;      many packets)
;;        -- need to add modstream 2
;;    2.0 -- 10/22/08
;;      Begin transition to version control
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;Load in the common blocks
common constants
common ratecoefs
common plasma

;; Turn off the logging and open a new logfile
if (!journal) then journal
w = file_test('logs/') & if ~w then spawn, 'mkdir logs'
if (n_elements(logfile) NE 1) then begin
  spawn, 'date +"%Y-%m-%dT%H:%M"', date
  logfile = 'logs/modelrun.' + date + '.pro'
endif

print, 'Opening log file: ' + logfile
journal, logfile

tstart = systime(1)
tittot = 0.

;; Set up the stuff structure
if (local EQ !null) then local = 0
stuff = {s:0, aplanet:0., vrplanet:0., radpres_v:ptr_new(0), $
  radpres_const:ptr_new(0), local:local, strstart:''}

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Determine run options
if (overwrite EQ !null) then overwrite = 0
if (generic EQ !null) then generic = 1
if (quick EQ !null) then quick = 0
if (runmodel EQ !null) then runmodel = 1

dogen = (generic EQ 1) or (generic EQ 2)
if (quick) then dogen = 0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Determine program version
readfmt, 'version.dat', /silent, 'A100', version
version = strtrim(version, 2)
if (n_elements(version) EQ 1) then stop
ntot = 0L

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; Loop over each inputfile
ninputs = n_elements(inputfiles)
for iii=0,ninputs-1 do begin
  trun0 = systime(1)
  stuff.strstart = 'Inputfile #' + strint(iii) + ': '
```

```

    inputfile = inputfiles[iii]
    print, '***********************'
    print, stuff.strstart + 'Starting ' + inputfile
    print, stuff.strstart + systime(0)

    totalpackets = 0L    ;; reset the number of packets
    runthrough = 0      ;; reset the run number
    owrite = overwrite  ;; reset the overwrite state
    ;; Run model until have specified number of packets

    while (totalpackets LT npackets) do begin
      print, stuff.strstart + 'Starting Simulation #' + strint(runthrough)

      ;; (a) Read in the inputs -- Reset to ensure no changes have been made
      input = inputs_restore(inputfile)
      SystemConstants, input.geometry.planet, SystemConsts, DipoleConsts
      geninput = inputs_restore(inputfile)

      ;; (b) Determine how many packets have already been run
      if ~(quick) then begin
        owrite = (runthrough EQ 0) ? overwrite : 0
        totalpackets = model_findpackets(input, generic=generic, overwrite=owrite)
        ntodo = npackets - totalpackets
      endif else ntodo = npackets ;; quick

      ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
      ;; If runmodel = 0, then just doing the packet extraction part
      if (runmodel) and (ntodo GT 0) then begin
        if (dogen) then input = temporary(geninput)

        ;; Read in the constants
        stuff.s = (where(strlowcase(*SystemConsts.Objects) EQ $
          strlowcase(input.geometry.StartPoint)))[0]

        ;; Determine distance and radial velocity of planet relative to the sun
        planet_dist, input.geometry.taa, SystemConsts, distance=dd, velocity=vv
        stuff.aplanet = dd
        stuff.vrplanet = vv/SystemConsts.rplan

        ;; find the default reactions and datasets
        if (input.options.lifetime EQ 0) $
          then loss_info = lifetime_setup(input) $
          else loss_info = !null

        ;; Set up the radiation pressure
        if (input.forces.radpres) then begin
          q = get_gvalue(input.options.atom, stuff.aplanet)
          ;;q /= SystemConsts.rplan ;; v in rplan/s, a in rplan/s^2
          *stuff.radpres_v = *q.v
          *stuff.radpres_const = *q.radaccel/SystemConsts.rplan
    endif else begin
      *stuff.radpres_v = 0.
      *stuff.radpres_const = 0.
    endelse

    ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
    ;; Determine how to run the additional packets that are needed
    ;; run 10 iterations of 100,000 packets to produce a run of 1,000,000 packets
    ;; Each outputfile produced will have 1,000,000 packets
    maxpack = 1000000L
    nruns = ceil(ntodo/maxpack)
    nits = 10
    packs_per_it = 100000L
    if (dogen) $
      then print, stuff.strstart + 'Running generic model.' $
      else print, stuff.strstart + 'Running Model'
    print, stuff.strstart + 'Will complete ' + strint(nruns) + $
      ' runs of 1,000,000 packets.'

    for i=0,nruns-1 do begin
      outputfile = output_filename(input)
      print, stuff.strstart + 'Outputfile = ' + outputfile
      for j=0,nits-1 do begin
        tit0 = systime(1)
        print, '** Starting run #' + strint(i+1) + ' of ' + strint(nruns)
        print, '** Starting iteration #' + strint(j+1) + ' of 10'

        ;; Determine the initial source distribution
        source_distribution, input, packs_per_it, seed, output=output
        if (input.options.lifetime EQ 0) $
          then output.loss_info = {reactions:ptr_new(loss_info.reaction), $
            files:ptr_new(loss_info.file), type:ptr_new(loss_info.type)}

        ;; Run the model
        print, '**** RUNNING *****'
        driver, input, output, seed=seed
        save, output, input, version, file=outputfile+'.' + strint(j)
        destroy_structure, output

        ;; Concluding stuff
        tit1 = systime(1)
        ntot++
        tittot += (tit1-tit0)
        print, stuff.strstart + 'Iteration time = ' + strint(tit1-tit0) + ' seconds'
        print, stuff.strstart + 'Mean Iteration time = ' + strint(tittot/ntot) + $
          ' seconds'
      endfor
      combine_iterations, outputfile, /delete
      print, 'Finished ' + outputfile
    endfor  ;; runs
```

```
      ;; Cleanup the memory
      destroy_structure, input

      destroy_structure, plasma
      destroy_structure, plasmahot
      destroy_structure, plasma

      destroy_structure, coef_eimp
      destroy_structure, coef_chx
      destroy_structure, coef_photo

      print, stuff.strstart + 'Finishing Simulation #' + strint(runthrough)
      runthrough++
    endif else totalpackets = npackets ;; not running model, so end while loop

    ;; If not running quick version, always want to do another runthrough
    ;; to extract packets from newly available generic files
    ;; If running quick version then stop running
    if (quick) then totalpackets = npackets
    destroy_structure, SystemConsts
  endwhile

  trun1 = systime(1)
  print, stuff.strstart + 'Finishing ' + inputfile
  print, stuff.strstart + 'Time for inputfile: ' + strint((trun1-trun0)/3600.) + ' hours'
  print, stuff.strstart + 'Total elaspsed time: ' + strint((trun1-tstart)/3600.) + ' hours'
  print, '  ' + inputfile
  print
endfor ; iii

;; close the logfile
journal
print, 'Closed log file ' + logfile

end
```