```
function produce_results, inputtemp, formattemp, data=data, npackets=npackets, $
  savefile=savefile, local=local

common constants
common results
time0 = systime(1)

if (local EQ !null) then local = 0

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;; For instructions, see: modelpro_2.0/Docs/produce_results.tex
;;
;; Given and inputfile and an output format file, produce the
;; desired output.
;;
;; All positions and angles need to be given in a reference frame with
;; the +y axis pointed away from the sun -- i.e. in the model reference frame
;;
;; Inputs:
;;  inputtemp - can be
;;    (a) inputfile - restore input and search for outputfiles
;;    (b) input structure - search for outputfiles
;;    (c) outputfile - restore
;;  formattemp = either a format structure or a file with the format
;;
;; Keyword Inputs:
;;   * npackets = minimum number of packets that are needed to continue.
;;
;; Version History:
;;   4.0: 25 Jan 2011
;;     * Original based on previous routines
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

if (npackets EQ !null) then npackets = 0 ;; If not specified, only need 1 packet
if (data EQ !null) then data = -1

fname = 'produce_results: '
stuff = {aplanet:0d, vrplanet:0d, atoms_per_packet:0d, mod_rate:0d, totalsource:0d, $
  local:local}

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; restore the inputs and determine outputfiles to use
ss = size(inputtemp, /type)
case (1) of
  (ss EQ 7) and (stregex(inputtemp[0], '.output', /fold, /bool)): begin
    ;; A list of output files has been given
    ofile = obj_new('IDL_savefile', inputtemp[0])
    ofile.restore, 'input'
    obj_destroy, ofile
```

```
      files = inputtemp
      ;SystemConstants, input.geometry.planet, SystemConsts
      end
    (ss EQ 7) and (stregex(inputtemp, '.input', /fold, /bool)): begin
      ;; the name of an input file is given
      input = inputs_restore(inputtemp)
      SystemConstants, input.geometry.planet, SystemConsts
      files = modeloutput_search(input, nfiles=n0)
      end
    (ss EQ 8): begin
      ;; an input structure is given
      input = inputtemp
      SystemConstants, input.geometry.planet, SystemConsts
      files = modeloutput_search(input, nfiles=n0)
      end
    else: stop
  endcase
  if (size(input, /type) NE 8) then stop
  nfiles = (files[0] EQ '') ? 0 : n_elements(files)
  print, fname + strint(nfiles) + ' output files found.'

  ;; Restore the system constants
  planet_dist, input.geometry.taa, SystemConsts, distance=aplanet, velocity=vrplanet
  stuff.aplanet = aplanet
  stuff.vrplanet = vrplanet

  ;; Determine the number of packets available
  if (nfiles GT 0) then begin
    pack = extract_parameter('savedpackets', files)
    totalpackets = long(total((pack.values()).ToArray(type='long')))
    pack = 0  ; get around an IDL bug
  endif else totalpackets = 0L
  print, fname + strint(totalpackets) + ' packets found.'

  ;; If there are enough packets, process the result
  if (totalpackets GT npackets) then begin
    ;; Restore the results format file
    case (size(formattemp, /type)) of
      7: format = read_resultformat(formattemp)
      8: format = formattemp
      else: stop
    endcase
    if (size(format, /type) NE 8) then stop

    ;; Determine the packet conversion
    tt = extract_parameter('totalsource', files)
    stuff.totalsource = total((tt.values()).ToArray(type='double'))
    tt = 0  ; get around an IDL bug

    stuff.mod_rate = stuff.totalsource / input.options.endtime ;; packets ejected per sec
    stuff.atoms_per_packet = (format.strength *1e26) / stuff.mod_rate
      print, fname + strint(stuff.mod_rate) + ' packets ejected per second'
      print, fname + strint(stuff.atoms_per_packet) + ' atoms per packet'

      ;;;;;;;;;;;;;;;;
      ;; Set up intensity if needed
      if (format.quantity EQ 'intensity') then results_intensity_setup

      ;; take different path for each result type
      case strlowcase(format.type) of
        'image': result = produce_image(files, savefile=savefile)
        'voronoi image': result = produce_voronoi_image(files, savefile=savefile)
      'los': result = produce_los(files, data)
      'points': result = produce_density(files, data, savefile=savefile)
      else: stop
      endcase
    endif else begin ;; (totalpackest < npackets)
  print, fname + 'Too few packets found.'
  result = -1
  endelse

  time1 = systime(1)
  print, 'Total runtime = ' + strint(round(time1-time0)) + ' seconds'

  return, result

  end

```