
Subject: CONTTW.PRO (11 pages long)
Posted by [zawodny](#) on Fri, 11 Sep 1992 14:47:23 GMT
[View Forum Message](#) <> [Reply to Message](#)

```
;+
; NAME:
; CONTTW
; PURPOSE:
; Produce a color filled contour
; CATEGORY:
; Z4 - IMAGE PROCESSING, data display
; CALLING SEQUENCE:
; CONTTW,A,XOLD,YOLD {,keys}
; Calling sequence and keys are nearly identical to CONTOUR.
; INPUTS:
; A Array to be contoured
; XOLD Vector of X coordinates
; YOLD Vector of Y coordinates
; KEYWORD PARAMETERS:
; In this section (XYB) stands for X-axis, Y-axis, and Color Bar.
; The Keywords: (XYB)RANGE, (XYB)STYLE, (XYB)TICKS, (XYB)TICKV, (XYB)MINOR,
; (XYB)TITLE, (XYB)TYPE, (XYB)TICKNAME, TITLE, SUBTITLE, and CHARSIZE
; are similar to their implementations in CONTOUR. (defaults to associated
; !x, !y, or !p values)
;
; /RIGHT, /LEFT, /TOP, /BOTTOM, and /NOBAR determines the location of
; the optional color bar.
; TICKLEN, BLEN The default for this parameter is to draw the ticks
; inward. If outward tick marks are needed then this
; parameter should be set to a negative value. BLEN
; controls the color bar; defaults to ticklen*10.
; CHARSIZE Size of axis labels (default = 1).
; THICK Sets the thickness of all lines and characters
; (labels of contour lines cannot be thickened).
; LEV Vector of levels to contour.
; WINDOW=n Selects the window number to be used.
; /NEW If this keyword is present, highest unused window is
; used.
; WSIZE=[x,y] Sets the size of the window (default = [1024,800]).
; /SHOW If this keyword is present then the image will appear
; as a viewable window.
; /OVERLAY Controls whether line contours should be overplotted.
; /INVERSE White on black background.
; /DATE Dates the image along the bottom.
; TABLE Uses requested color table (default = 19).
; RES Determines the coarseness of the plot (default = 5).
; /PS Write postscript file. No screen output.
; /RESET Resets !X, !Y, !P, !QUIET at end of program.
```

```

;
; Color mapping is controlled by BRANGE, if present. Otherwise it is
; controlled by the first and last elements of LEV. If both are missing,
; then the program makes an attempt at auto-scaling (no promises here).
;
;
; OUTPUTS: Either creates a color graphic or gray scale on screen or
;           in a postscript file or creates a TEK-4693D compatible file
;           if the appropriate software is available.
; COMMON BLOCKS: CONTTW_1
; SIDE EFFECTS: Loads a color table.
; RESTRICTIONS: Assumes A is on an evenly spaced grid in both x and y. If
; A is not on a regular grid, the user should use
; POLY_2D or splines to place it on one.
; Change the default color scale (defcol) value when installed
; PROCEDURE: Nothing too fancy to require documentation.
; MODIFICATION HISTORY:
; Aug 14, 1989 J. M. Zawodny, NASA/LaRC, MS 475, Hampton VA, 23665.
; Jan 04, 1990 R. E. Boughner, modified for use with subarrays.
; Jul 27, 1990 J. M. Zawodny, Major upgrade to IDL version 2.
; Nov 27-Dec 13, 1991 A. C. Edwards, changed color bars, changed
; pixels, added xybtkv, xybtkname. Changed
; color bar procedure so tickmarks show. Added ability
; to make postscript files. Added ability to do
; gray scale plots after checking !d.n_colors
; AND reset plotting variables only on command.
; Jan-Feb, 1992 organizing the mess
; Feb 20, 1992 Deleted /HELP and /LOGO options so program would
; compile in default program area (.SIZE not needed).
; Aug 30, 1992 Changed to hardware fonts when in X.
;-

pro CONTTW,a,xold,yold $
, show=show, new=new, window=window, ps=ps, lev=lev, wsize=wsize $
, overlay=overlay, help=help, reset=reset, right=right, left=left, top=top $
, bottom=bottom, nobar=nobar, brange=brange, btitle=btitle, bticks=bticks $
, bminor=bminor, bstyle=bstyle, btype=btype, blen=blen, btickv=btickv $
, btickname=btickname, xrange=xrange, xtitle=xtitle, xticks=xticks, res=res $
, xminor=xminor, xstyle=xstyle, xtype=xtype, xtickv=xtickv, table=table $
, xtickname=xtickname, yrange=yrange, ytitle=ytitle, yticks=yticks $
, yminor=yminor, ystyle=ystyle, ytype=ytype, ytickv=ytickv, date=date $
, ytickname=ytickname, thick=thick, title=title, ticklen=ticklen $
, charsize=charsize, inverse=inverse, maxval=maxval, fillval=fillval

; Define Common Blocks
common CONTTW_1,pixf,text,bkgnd,time_flag,logo_flag
; Default color scale
defcol = 19

```

```

; Save some Plotting Things here
save_p = !p & save_x = !x & save_y = !y & save_noerase = !p.noerase

; Test for sufficient information
if n_params() lt 3 then begin
  print,'CONTTW called with too few parameters'
  return
endif

; Find the min and max of array a
mia = min(a) & maa = max(a)
; Do we try to auto-scale? (Don't try too hard!)
n_lev = n_elements(lev)
if (not keyword_set(brange) and (n_lev eq 0)) then begin
  if !quiet eq 0 then print,' Auto-scale requested '
  dma = (maa-mia)/10.
  BRANGE = [mia,maa]
  if keyword_set(overlay) then lev = findgen(11)*dma+mia
  n_lev = n_elements(lev)
endif

; If axis parameters have not been set then make defaults
if not keyword_set(xstyle) then xstyle = !x.style or 1
if not keyword_set(ystyle) then ystyle = !y.style or 1
if not keyword_set(bstyle) then bstyle = 1
if not keyword_set(xtype) then xtype = !x.type
if not keyword_set(ytype) then ytype = !y.type
if not keyword_set(btype) then btype = 0
if not keyword_set(xticks) then xticks = !x.ticks
if not keyword_set(yticks) then yticks = !y.ticks
if not keyword_set(bticks) then bticks = n_lev-1
if not keyword_set(ytickv) then ytickv = !y.tickv
if not keyword_set(btickv) then btickv = [0,0]
if not keyword_set(xtickname) then xtickname = ''
if not keyword_set(ytickname) then ytickname = ''
if not keyword_set(btickname) then btickname = ''
if not keyword_set(xminor) then xminor = !x.minor
if not keyword_set(yminor) then yminor = !y.minor
if not keyword_set(bminor) then bminor = 0
if not keyword_set(xtitle) then xtitle = !x.title
if not keyword_set(ytitle) then ytitle = !y.title
if not keyword_set(btitle) then btitle = ''
if not keyword_set(brange) then brange = [lev(0),lev(n_lev-1)]
if not keyword_set(xrange) then xrange = !x.range
if not keyword_set(yrange) then yrange = !y.range

; If plot parameters have not been set then make defaults
if not keyword_set(maxval) then maxval = 1.e33

```

```

if not keyword_set(fillval) then fillval = 1.e34
if not keyword_set(title) then title = !p.title
if not keyword_set(subtitle) then subtitle = !p.subtitle
if not keyword_set(ticklen) then ticklen = !p.ticklen
if not keyword_set(blen) then blen = ticklen*10
if not keyword_set(charsize) then charsize = !p.charsize
if(n_elements(show) eq 0) then show = 1
if keyword_set(overlay) then overlay = 0 else overlay = 1
if keyword_set(ps) then set_graph,/cps,/land
!p.title="

; Set axis and plot parameters to thick
if keyword_set(thick) then begin
  !x.thick = thick & !y.thick = thick & !p.thick = thick
  !p.charthick = thick
endif else thick = 0

; if !D.N_COLORS ne 256 (color) or 16 (gray scale) then try to reset IDL
if (!d.n_colors ne 256 and !d.n_colors ne 16) then device,/close_displ

; Determine whether to use gray scale or full color values
if (!d.n_colors eq 16 and (not keyword_set(ps))) then begin
; **** These should be changed to suit your needs ****
white = 15 & black = 0
topcolor = 13 & offset = 1B
totalcolors = 16 & ncolors = 14
endif else begin
; **** These should be changed to suit your needs ****
white = 255 & black = 251
topcolor = 250 & offset = 0B
totalcolors = 256 & ncolors = 251
endelse

; Skip this if doing a postscript plot
if not keyword_set(ps) then begin
  ; Set common block defaults
  if keyword_set(res) then pixf = res else pixf = 5
  if keyword_set(date) then time_flag = 1 else time_flag = 0
  if keyword_set(logo) then logo_flag = 1 else logo_flag = 0
endif

if keyword_set(inverse) then begin
  ; Reverse screen colors
  text = white & bkgnd = black
endif else begin
  text = black & bkgnd = white
endelse

```

```

; Find out where color bar goes
barlocs = ['/NOBAR','/RIGHT','/LEFT','TOP','/BOTTOM']
barloc = [keyword_set(nobar),keyword_set(right),keyword_set(left), $
          keyword_set(top),keyword_set(bottom)]
barloc = where(barloc,len)
if (len gt 1) then begin
  print,'CONTTW called with incompatible keywords'
  print,barlocs(barloc)
  return
endif

```

```

; Default to /NOBAR
if (len eq 0) then barloc=0 else barloc=barloc(0)

```

```

; Well we are now safe!
quiet = !QUIET & !QUIET = 1

```

```

; SKIP THIS WINDOW STUFF IF POSTSCRIPT IS CALLED
if not keyword_set(ps) then begin

```

```

  ; Set window, plot, and bar sizes
  if keyword_set(wsize) then begin
    wdx = wsize(0) & wdy = wsize(1)
  endif else begin
    wdx = 935 & wdy = 825
  endelse

```

```

  ; If a new window is requested or no window has ever been used.
  ; WINDOW keyword overrides /NEW.

```

```

  if((not keyword_set(window)) and (keyword_set(new) or $
    !d.window eq -1)) then begin

```

```

    if keyword_set(show) then begin
      ; If we need to show the window
      window,0,xsize=wdx,ysize=wdy,title="", $
        xpos=2,ypos=2,retain=2,color=totalcolors
      if not keyword_set(show) then wshow,!d.window,0
    ; otherwise make it a pixmap
    endif else window,0,xsize=wdx,ysize=wdy,/pixmap,$
      color=totalcolors
    endif else begin

```

```

      ; Using an old window or a specific window
      if keyword_set(window) then wnum=window else wnum=!d.window
      if keyword_set(show) then begin
        ; If we need to show the window
        window,wnum,xsize=wdx,ysize=wdy,title="", $
          xpos=2,ypos=2,retain=2,color=totalcolors
        if not keyword_set(show) then wshow,!d.window,0
      ; otherwise make it a pixmap

```

```

endif else window,wnum,xsize=wdx,ysize=wdy,/pixmap,$
color=totalcolors
endif

```

```

; **** These should be changed to suit your needs ****

```

```

;set font
!p.font=0 & device,font='times_roman18'

```

```

;"Zero" workspace to background color
tv,replicate(bkgnd,wdx,wdy)

```

```

; We need to Overlay Multiple Objects

```

```

!P.NOERASE = 1

```

```

endif

```

```

; **** These should be changed to suit your needs ****

```

```

; Select a color table

```

```

if keyword_set(table) then coltab=table else coltab = defcol

```

```

if totalcolors eq 15 then loadct,0 else loadct,coltab

```

```

; Reset !COLOR for line drawing

```

```

if keyword_set(ps) then !p.color = black else !p.color = text

```

```

; Make extra room for large values along bar

```

```

if((abs(brange(0)) gt 100) or (abs(brange(1)) gt 100))then xb=1 else xb=0

```

```

; Make extra room for negative signs

```

```

if((brange(0) lt 0) or (brange(1) lt 0))then xn=1 else xn=0

```

```

; Take back extra room if there is not a bar title

```

```

if(btitle eq "") then xt=2 else xt=0

```

```

; Set position of plot and bar regions and plot color bar

```

```

bar = byte(indgen(ncolors,2) mod ncolors)+offset

```

```

case barloc of

```

```

1: begin ; Bar on Right

```

```

    preg    = [.0,.0,.85,1.]

```

```

    pmarx    = [8,1]

```

```

    pmary    = [4,2]

```

```

    !p.region = [.85,.0,1.,1.]

```

```

    !x.margin = [0,9.5+xb+xn-xt]

```

```

    !y.margin = [8,5]

```

```

    ; Locate bar area

```

```

    plot,[0,1],[0,1],/nodata,xstyle=4,ystyle=4, $

```

```

    ytit=btitle,chars=charsize-.5,back=white

```

```

    ; Make a Color Bar

```

```

    bar = transpose(bar)

```

```

    if keyword_set(ps) then begin

```

```

; Postscript color bar
bardevx=[!p.clip(0),!p.clip(2)]
bardevy=[!p.clip(1),!p.clip(3)]
barnorm=convert_coord(bardevx,bardevy,$
  /to_normal,/device)
barnx=barnorm(0,1)-barnorm(0,0)
barny=barnorm(1,1)-barnorm(1,0)
tv,bar,barnorm(0,0),barnorm(1,0),$
  /normal,xsize=barnx,ysize=barny
endif else begin
; Screen color bar
barw = 28 ; .25:
barl = !p.clip(3)-!p.clip(1)+1
expand,bar,barw,barl,rbar
tv,rbar,!p.clip(0)+28,!p.clip(1)
endelse
rbar = 0 ; save memory
; Draw color bar axis
axis,yaxis=1,ystyle=bstyle,yran=brange,ytitle=btittle, $
  ytype=btype,yticks=bticks,yminor=bminor,ticklen=blen, $
  charsize=charsize-.5,thick=thick,/nocli, $
  ytickv=btickv,yticknam=btickname
end
2: begin ; Bar on Left
preg    = [.15,.0,1.,1.]
pmarx   = [6,1.5]
pmary   = [4,2]
!p.region = [.0,.0,.15,1.0]
!x.margin = [9+xb+xn-xt,0]
!y.margin = [8,5]
  ; Locate bar area
plot,[0,1],[0,1],/nodata,xstyle=4,ystyle=4,ytit=btittle, $
  chars=charsize-.5,back=white
; Make a Color Bar
bar = transpose(bar)
if keyword_set(ps) then begin
; Postscript color bar
bardevx=[!p.clip(0),!p.clip(2)]
bardevy=[!p.clip(1),!p.clip(3)]
barnorm=convert_coord(bardevx,bardevy,$
  /to_normal,/device)
barnx=barnorm(0,1)-barnorm(0,0)
barny=barnorm(1,1)-barnorm(1,0)
tv,bar,barnorm(0,0),barnorm(1,0),$
  /normal,xsize=barnx,ysize=barny
endif else begin
; Screen color bar
barw = 28 ; .25"

```

```

    barl = !p.clip(3)-!p.clip(1)+1
    expand,bar,barw,barl,rbar
    tv,rbar,!p.clip(0),!p.clip(1)
endelse
rbar = 0 ; save memory
; Draw color bar axis
axis,yaxis=0,ystyle=bstyle,yran=brange,ytitle=bttitle, $
  ytype=btype,yticks=bticks,yminor=bminor,ticklen=blen, $
  charsize=charsize-.5,thick=thick,/noclip, $
  ytickv=btickv,ytickname=btickname
end
3: begin ; Bar on Top
  preg    = [.0,.0,1.,.85]
  pmarx    = [8,1.5]
  pmary    = [3.5,2]
  !p.region = [.0,.85,1.,1.]
  !x.margin = [15,6]
  !y.margin = [0,4-xt]
    ; Locate bar area
  plot,[0,1],[0,1],/nodata,xstyle=4,ystyle=4,xtit=bttitle, $
    chars=charsize-.5,back=white
; Make a Color Bar
if keyword_set(ps) then begin
; Postscript color bar
  bardevx=[!p.clip(0),!p.clip(2)]
  bardevy=[!p.clip(1),!p.clip(3)]
  barnorm=convert_coord(bardevx,bardevy,$
    /to_normal,/device)
  barnx=barnorm(0,1)-barnorm(0,0)
  barny=barnorm(1,1)-barnorm(1,0)
  tv,bar,barnorm(0,0),barnorm(1,0),$
    /normal,xsize=barnx,ysize=barny
endif else begin
; Screen color bar
  barw = !p.clip(2)-!p.clip(0)+1
  barl = 28 ;calculated to be .25"
  expand,bar,barw,barl,rbar
  tv,rbar,!p.clip(0),(!p.clip(3)-28)
endelse
rbar = 0 ; save memory
; Draw color bar axis
axis,xaxis=1,xstyle=bstyle,xran=brange,xtitle=bttitle, $
  xtype=btype,xticks=bticks,xminor=bminor,ticklen=blen, $
  charsize=charsize-.5,thick=thick,/noclip, $
  xtickv=btickv,xtickname=btickname
end
4: begin ; Bar on Bottom
  preg    = [.0,.15,1.,1.]

```

```

pmarx   = [8,1.5]
pmary   = [3.5,3]
!p.region = [.0,.0,1,..15]
!x.margin = [15,6]
!y.margin = [4.5-xt,-.5]
; Locate color bar
plot,[0,1],[0,1],/nodata,xstyle=4,ystyle=4,xtit=btittle, $
  chars=charsize-.5,back=white
; Make a Color Bar
if keyword_set(ps) then begin
; Postscript color bar
bardevx=[!p.clip(0),!p.clip(2)]
bardevy=[!p.clip(1),!p.clip(3)]
barnorm=convert_coord(bardevx,bardevy,$
  /to_normal,/device)
barnx=barnorm(0,1)-barnorm(0,0)
barny=barnorm(1,1)-barnorm(1,0)
tv,bar,barnorm(0,0),barnorm(1,0),$
  /normal,xsize=barnx,ysize=barny
endif else begin
; Screen color bar
barw = !p.clip(2)-!p.clip(0)+1
barl = 28 ; .25"
expand,bar,barw,barl,rbar
tv,rbar,!p.clip(0),!p.clip(1)
endelse
rbar = 0 ; save memory
; Draw color bar axis
axis,xaxis=0,xstyle=bstyle,xran=brange,xtitle=btittle, $
  xtype=btype,xticks=bticks,xminor=bminor,ticklen=blen, $
  charsize=charsize-.5,thick=thick,/noclip, $
  xtickv=btickv,xtickname=btickname
end
ELSE: begin ; No Bar preg = [.0,.0,1,.1.]
  pmarx = [8,1]
  pmary = [4,2]
  goto,nobar
end
endcase
!p.noerase = 1

NOBAR:
; Set the position of the contour box
!p.region = preg
!x.margin = pmarx & !y.margin = pmary

; Make the axes invisible
xss = xstyle or 4 & yss = ystyle or 4

```

```
; Set up the contour box (This is done to set !p.clip)
contour,a,xold,yold,/nodata,XRANGE=xrange,XSTYLE=xss,XTITLE= xtitle, $
CHARSIZE=charsize,YRANGE=yrange,YSTYLE=yss,YTICKS=yticks, $
YTITLE=ytitle,TITLE=title,SUBTITLE=subtitle,TICKLEN=ticklen
```

```
; Calculate the dimensions for the image
xcmi = min(!x.crange) & ycmi = min(!y.crange)
xcma = max(!x.crange) & ycma = max(!y.crange)
xmin = min(xold)>xcmi & ymin = min(yold)>ycmi
xmax = max(xold)<xcma & ymax = max(yold)<ycma
```

```
; Determine whether to flip a before plotting
sizex = size(xold) & sizey = size(yold)
lastx = sizex(1)-1 & lasty = sizey(1)-1
xorder = (xrange(1)-xrange(0))*(xold(lastx)-xold(0))
yorder = (yrange(1)-yrange(0))*(yold(lasty)-yold(0))
```

```
; if xorder, yorder are negative, then image should be rotated 180
if(xorder lt 0 and yorder lt 0) then begin
  anew = rotate(a,2)
  x  = reverse(xold)
  y  = reverse(yold)
endif
```

```
; if xorder is negative, then image should be flipped left to right
if(xorder lt 0 and yorder gt 0) then begin
  anew = rotate(a,5)
  x  = reverse(xold)
  y  = yold
endif
```

```
; if yorder is negative, then image should be flipped top to bottom
if(xorder gt 0 and yorder lt 0) then begin
  anew = rotate(a,7)
  x  = xold
  y  = reverse(yold)
endif
```

```
; if xorder, yorder are positive, then no change
if(xorder gt 0 and yorder gt 0) then begin
  anew = a
  x  = xold
  y  = yold
endif
```

```
; Determine the size of the plot box
pdx = !p.clip(2)-!p.clip(0)+1 & pdy = !p.clip(3)-!p.clip(1)+1
```

```
; Calculate the x-limits of the array
if(!x.crange(0) le !x.crange(1)) then begin
```

```

loc = where(x ge xmin)
x0 = loc(0)
xmin = x(x0)
loc = where((x gt xmax),count)
if (count eq 0) then begin
  x1 = n_elements(x)-1
endif else begin
  x1 = loc(0)-1
  xmax = x(x1)
endelse
xout = xmin
endif else begin
  loc = where(x le xmax)
  x0 = loc(0)
  xmax = x(x0)
  loc = where((x lt xmin),count)
  if (count eq 0) then begin
    x1 = n_elements(x)-1
  endif else begin
    x1 = loc(0)-1
    xmin = x(x1)
  endelse
  xout = xmax
endelse

; Calculate the y-limits of the array
if(!y.crangle(0) le !y.crangle(1)) then begin
  loc = where(y ge ymin)
  y0 = loc(0)
  ymin = y(y0)
  loc = where((y gt ymax),count)
  if (count eq 0) then begin
    y1 = n_elements(y)-1
  endif else begin
    y1 = loc(0)-1
    ymax = y(y1)
  endelse
  yout = ymin
endif else begin
  loc = where(y le ymax)
  y0 = loc(0)
  ymax = y(y0)
  loc = where((y lt ymin),count)
  if (count eq 0) then begin
    y1 = n_elements(y)-1
  endif else begin
    y1 = loc(0)-1
    ymin = y(y1)
  endelse
endelse

```

```

endelse
yout = ymax
endelse

; If set for postscript, output to postscript file convert ranges from data
; to normal for postscript otherwise output to screen.
if keyword_set(ps) then begin
  idx   = 300
  idy   = 225
  normcoor = convert_coord([xmin,xmax],[ymin,ymax],/to_normal,/data)
  nx     = normcoor(0,1)-normcoor(0,0)
  ny     = normcoor(1,1)-normcoor(1,0)
  expand, anew(x0:x1,y0:y1), idx, idy, work, maxval=maxval, fillval= fillval

  ; Check for nodata
  m = where(work eq fillval, num)

  ; Scale it to the color scale
  if (btype eq 0) then begin
    image = byte(topcolor/float(brange(1)-brange(0))* $
      (((work>brange(0))<brange(1))-brange(0)))+offset
  endif else begin
    logb = alog10(brange)
    image = byte(topcolor/(logb(1)-logb(0))*(((alog10(work) $
      >logb(0))<logb(1))-logb(0)))+offset
  endif
endelse

; Fill nodata with proper color
image(m) = bkgnd

endif else begin ; screen output

; Is the X-axis linear or logarithmic
if((xtype and 1) eq 0) then begin ; Linear X
  idx = fix((xmax-xmin)*(pdx-1)/(xcma-xcmi))+1
endif else begin ; Log X
  idx = fix(alog(xmax/xmin)/alog(xcma/xcmi)*(pdx-1))+1
  ; /data does not work in LOG coordinates so fix xout
  xout = alog(xout/xcmi)/alog(xcma/xcmi)*(xcma-xcmi)+xcmi
endelse

; Is the Y-axis linear or logarithmic
if((ytype and 1) eq 0) then begin ; Linear Y
  idy = fix((ymax-ymin)*(pdy-1)/(ycma-ycmi))+1
endif else begin ; Log Y
  idy = fix(alog(ymax/ymin)/alog(ycma/ycmi)*(pdy-1))+1
  ; /data does not work in LOG coordinates so fix yout
  yout = alog(yout/ycmi)/alog(ycma/ycmi)*(ycma-ycmi)+ycmi

```

```

endelse

; Expand the input array to the proper image dimensions for screen
expand,anew(x0:x1,y0:y1),idx/pixf+1,idy/pixf+1,work,maxv=max val,fillv=fillval

; Check for nodata
m = where(work eq fillval,num)

; Scale it to the color scale
if(btype eq 0) then begin
  work = byte(topcolor/float(brange(1)-brange(0))* $
    (((work>brange(0))<brange(1))-brange(0)))
endif else begin
  logb = alog10(brange)
  work = byte(topcolor/(logb(1)-logb(0))* $
    (((alog10(work)>logb(0))<logb(1))-logb(0)))
endelse

; Fill nodata with proper color
if(num ne 0.) then work(m) = bkgnd

; Replicate Pixel to Achieve Final Size
bigp = replicate(0B,pixf,pixf)

; These loops are ugly but faster than alternatives
image = bytarr(idx+pixf,idy+pixf)
for ky=0,idy/pixf do begin
  iiy=ky*pixf
; Fill in the image
  for kx=0,idx/pixf do image(kx*pixf,iiy)=work(kx,ky)+bigp
endfor

endelse

; output image to ps file or screen
if keyword_set(ps) then begin
  image=image(0:idx-1,0:idy-1)
  tv,image,normcoor(0,0),normcoor(1,0),/normal,xsize=nx,ysize= ny
endif else begin
; Trim to fit and output to window at proper position
  tv,image(0:idx-1,0:idy-1),xout,yout,/data
endelse

; Save memory
work = 0 & image = 0

; Draw all the axes and overlay a contour if requested
if not keyword_set(lev) then lev=[0,0]

```

```
contour,anew,x,y,levels=lev,/follow,nodata=overlay,XRANGE=xrange,XSTYLE= $
xstyle,XTICKS=xticks,XTITLE=xtitle,xtype=xtype, YRANGE=yrange, $
YSTYLE=ystyle,YTICKS=yticks,YTITLE=ytitle,ytype=ytype,TITLE= title, $
SUBTITLE=subtitle,TICKLEN=ticklen,CHARSIZE=charsize,/spline, $
XTICKNAME=xtickname,YTICKNAME=ytickname,XMINOR=xminor,YMINOR =yminor
```

```
; Skip this if postscript
If not keyword_set(ps) then begin
  ; Place Date and Time on Plot if available
  if(time_flag) then begin
    tstr = systime(0)
    timestr = strmid(tstr,4,3) + ' ' + strmid(tstr,8,2) + $
    ' ' + strmid(tstr,20,4)
    xyouts,.98,.02,timestr,size=1.0,/normal,align=1.
  endif

  if not keyword_set(show) then wshow,!d.window,1
endif

if keyword_set(reset) then begin
  ; Restore Saved Plotting Variables
  !x = save_x & !y = save_y & !p = save_p
  !QUIET = quiet
endif
!p.noerase = save_noerase

return
end
```
