
Subject: Positioning PostScript output, IDL Widgets

Posted by [oet](#) on Thu, 24 Mar 1994 08:45:25 GMT

[View Forum Message](#) <> [Reply to Message](#)

As mentioned in the IDL User's Guide 3.5, 3-40 and several news messages in the past, users are often confused by the use of the XOFFSET and YOFFSET keywords to the PostScript DEVICE routine. As a submodul for our idl interfaces I wrote a widget modul which allows interactively positioning postscript output on the paper. It can easy be used in interactive IDL mode or from within other IDL widget applications. It's just a first approach, comments and suggestions are welcome for further work on it.

Thomas

```
;
;+
; NAME:
; PS_PAPER
;
; PURPOSE:
;   Widget to interactively position IDL plots on a PostScript printer.
;   As noted in the IDL Users's Guide 3.5, 3-40, IDL users are often
;   confused by the use of the XOFFSET and YOFFSET keywords to the
;   PostScript DEVICE routine. The ps_paper widget makes it easy to
;   set these parameters correctly. The procedure can be used as a
;   sub modul from within other IDL GUI's, the keyword /MODAL should
;   be set in this case.
;   After setting the needed variables are stored in the common block
;   ps_device_com and can be accessed from the calling program similar
;   to the colors common block from xpalette.pro from the idl distribution.
;
; CATEGORY:
;   Printing
;
; CALLING SEQUENCE:
;   ps_paper [, /MODAL [, COLORS=<intarr(3)>] ]
;
; KEYWORD PARAMETERS:
;
;   WIN_SIZES
;   Two element vector with window sizes
;   If this keyword is set, the relation between
;   width and height is fixed, only the slider
;   which changes the longer side of the plot
;   area takes effect in this case. This can
;   be used to have the same aspect ratio for both
```

```

; the window and the PostScript device for
; WYSIWYG output. This is also useful for usage
; in combination with the 'annotate' draw program from
; the IDL widget library ( IDL 3.5.1 and above ).
;
; TITLE   Window title for popup window
;
; FONT    Widget Font for buttons, annotations
;
; MODAL   If ps_paper is called from within another
;         IDL widget application this keyword should
;         be set to set the calling widgets unsensitive
;         during the settings.
;
; COLORS  3 element vector ( int ) with the color indices
;         for the draw widget background, paper and text colors:
;
;         element 1 ( index 0 ): background color
;         element 2 ( index 1 ): paper color
;         element 3 ( index 2 ): text color
;
; HARDWARE_OFFS
;         Two element vector (float) with the hardware printer offsets
;         for the left/right offset and the top/bottom offset in
;         cm. Some printer can not print on the entire page.
;         This keyword allows to set correct hardware offsets
;         depending on the printer selected in an environment
;         with different printers. Some evaluated values:
;
;         Sparc Printer           [0.0,0.0]
;         Cannon Bubble Jet bj10ex [0.4,2.9]
;
;         If the centered is selected and the output on the
;         paper is not exact on the center of the page adjust
;         this values corresponding to your printer.
;
; OUTPUTS:
; After termination the postscript device keyword values for
; xsize, ysize, xoffset, yoffset, orientation can be accessed via
; the common block ps_device_com. If orient is equal 0 means portrait
; orientation, otherwise landscape orientation is selected. The
; values for xsize, ysize, xoffset, yoffset is set in centimeters.
;
; NOTE: -> if the orientation is not set together on the same <-
;         -> call to the device routine, it *must* be set before <-
;         -> setting the values for XSIZE, YSIZE, XOFFSET and <-
;         -> YOFFSET otherwise scaling the postscript output this <-
;         -> way won't work. See the example below. <-

```

```

;
;
; COMMON BLOCKS:
;   COMMON ps_paper_com, s, r_sizes, hw_offs, v_sizes, clrs
;   COMMON ps_device_com, xsize, ysize, xoffset, yoffset, orient
;
;
; SIDE EFFECTS:
;   Variables in common block ps_device_com are over written.
;
;
; RESTRICTIONS:
;   Currently the arrays r_sizes and v_sizes are hard coded only for
;   paper sizes a4 and a3, but it should be easy to change or add other
;   paper formats, simply editing the r_sizes and v_sizes structures.
;   IDL 3.5.1 or above required ( uses widget_control /NO_COPY ).
;   Only tested on SunOS 4.1.3, Solaris 2.3 with Sparc Printer,
;   PC-486-66 with ghostscript and a Canon bubble jet bj10ex printer.
;
;
; EXAMPLE:
;   Extract all inside the '*' border and save to 't_pspr.pro'. Start then
;   't_pspr' from the IDL prompt.
;
; *****
; * PRO t_pspr                                ;*
; * ;Example call procedure for ps_paper      ;*
; * COMMON ps_device_com, xsize, ysize, xoffset, yoffset, orient ;*
; *                                           ;*
; * ps_paper                                ;*
; *                                           ;*
; * old_dev = !d.name                      ;*
; * set_plot, 'PS'                         ;*
; *                                           ;*
; * ; Important: if the orientation is not set together on the same ;*
; * ; call to the device routine, it *must* be set before ;*
; * ; setting the values for XSIZE, YSIZE, XOFFSET and ;*
; * ; YOFFSET otherwise scaling the postscript output won't ;*
; * ; work.                                ;*
; * if ( orient EQ 0 ) then $              ;*
; *   device, /PORTRAIT $                  ;*
; * else device, /LANDSCAPE                ;*
; *                                           ;*
; * device, XSIZE=xsize, YSIZE=ysize, XOFFSET=xoffset, $ ;*
; *   YOFFSET=yoffset, FILENAME='ps_paper.ps' ;*
; * map_set, /CONTINENTS, /GRID           ;*
; * device, /close                         ;*
; *                                           ;*
; * print, "The file ps_paper.ps is now ready for printing" ;*
;

```



```
; Draw output box
; Reclaculate virtual paper sizes
```

```
v_sizes = $
{a4:[r_sizes.a4(0)- hw_offs(0)*2,r_sizes.a4(1)- hw_offs(1)*2], $
a3:[r_sizes.a3(0)- hw_offs(0)*2,r_sizes.a3(1)- hw_offs(1)*2] }
```

```
x0 = ( hw_offs(xp) ) * s.w_fact      ; printer hardware offsets
y0 = ( hw_offs(yp) ) * s.w_fact
tmp_v_sizes=v_sizes.(0)
xn = ( tmp_v_sizes(xp) * s.w_fact ) * s.width_fact
yn = ( tmp_v_sizes(yp) * s.w_fact ) * s.height_fact
```

```
if ( keyword_set(CENTER) ) then begin
s.centered=1
tmp_v_sizes=v_sizes.(0)
curr_xsize = ( tmp_v_sizes(xp) * s.w_fact )
curr_diffx = curr_xsize - ( curr_xsize * s.width_fact )
s.loffset = curr_diffx / 2.
```

```
tmp_v_sizes=v_sizes.(0)
curr_ysize = ( tmp_v_sizes(yp) * s.w_fact )
curr_diffy = curr_ysize - ( curr_ysize * s.height_fact )
s.boffset = curr_diffy / 2.
```

```
x0 = x0 + s.loffset
y0 = y0 + s.boffset
```

```
endif else begin
```

```
s.centered=0
```

```
endelse
```

```
wait, 0.1      ; a short wait was required on the IDL MS-Windows 3.5.1
                ; version on a 486-66 with VLB accelrated Cirrus VGA-Card;
                ; without this two waits before and after, the plot output
                ; below was erased after plotting and was never seen ...
                ; tooks some time to find out why ...
```

```
tmp_r_sizes=r_sizes.(0)
x0=x0 + ( !d.x_size / 2 - ( tmp_r_sizes(xp) * s.w_fact ) / 2 )
y0=y0 + ( !d.y_size / 2 - ( tmp_r_sizes(yp) * s.w_fact ) / 2 )
```

```
x2 = x0 + xn - 1      ; Draw new box.
y2 = y0 + yn - 1
```

```
plots,[x0,x2,x2,x0,x0],[y0,y0,y2,y2,y0],/DEVICE,COLOR=clrs.t ext, THICK=2.0
polyfill,[x0,x2,x2,x0,x0],[y0,y0,y2,y2,y0],/DEVICE,COLOR=clr s.text,$
/LINE_FILL, SPACING=0.1
```

```
wait, 0.1
```

```
empty
```

```
END
```

```
PRO ps_paper_event, ev
COMMON ps_paper_com, s, r_sizes,hw_offs, v_sizes, clrs
COMMON ps_device_com, xsize, ysize, xoffset, yoffset, orient
```

```
stash = widget_info(ev.handler, /CHILD)
widget_control, stash, GET_UVALUE=state, /NO_COPY
```

```
case ev.id of
```

```
state.orient_w: begin
    s.orient_p=ev.value
    goto, finish
end
```

```
state.size_w: begin
    s.size_p=ev.value
    goto, finish
end
```

```
state.width_w: begin
    s.width_fact=ev.value
    goto, finish
end
```

```
state.height_w: begin
    s.height_fact=ev.value
    goto, finish
end
```

```
state.hw_lr_offs_w: goto, finish
```

```
state.hw_tb_offs_w: goto, finish
```

```
state.center_w: begin
```

```

if ( s.centered EQ 1 ) then begin
  s.centered=0
  widget_control, state.center_w, SET_VALUE='Not centered'
endif else begin
  s.centered=1
  widget_control, state.center_w, SET_VALUE=' Centered '
endelse
goto, finish

end

```

```

state.ok_w: begin

```

```

  if ( s.orient_p EQ 0 ) then begin
    xp=0
    yp=1
  endif else begin
    xp = 1
    yp = 0
  endelse

```

```

; Calculation of postscript device keywords XSIZE, YSIZE,
; XOFFSET, YOFFSET to get correct position of output
; on the paper

```

```

; Calculate the values for the keywords XSIZE and YSIZE
tmp_v_sizes=v_sizes.(s.size_p)
xsize = tmp_v_sizes(xp) * s.width_fact
ysize = tmp_v_sizes(yp) * s.height_fact

```

```

; Calculate XOFFSET and YOFFSET

```

```

if ( s.orient_p EQ 0 ) then begin
  if ( s.centered EQ 1 ) then begin
    xoffset=s.loffset / s.w_fact
    yoffset=s.boffset / s.w_fact
  endif else begin
    xoffset=0
    yoffset=0
  endelse
endif else begin
  if ( s.centered EQ 1 ) then begin
    xoffset=s.boffset / s.w_fact

```

```

tmp_r_sizes=r_sizes.(s.size_p)
tmp_v_sizes=v_sizes.(s.size_p)
diffy = tmp_r_sizes(1) - ( tmp_v_sizes(1) * s.width_fact )
yoffset = tmp_r_sizes(1) - ( diffy / 2. )
yoffset = yoffset - hw_offs(1)

```

```

endif else begin
  xoffset=0
  yoffset=tmp_r_sizes(1) - hw_offs(1)
endelse
endelse

; Set orient to 0 for Portrait and 1 for Landscape
orient=s.orient_p

; The variables are set now and can be accessed from other moduls
; - similiar to the color setting widget "xpalette"
; with its common block "colors" - via the common
; block "ps_device_com" defined above
; COMMON ps_device_com, xsize, ysize, xoffset, yoffset, orient
;
widget_control, /DESTROY, ev.top
goto, quit
end

```

endcase

return

finish:

```

if ( total(s.fixed_facts) NE 2.0 ) then begin    ; win_sizes keyword
if ( s.fixed_facts(0) EQ 1.0 ) then begin      ; was set
  s.height_fact = s.fixed_facts(1) * s.width_fact
  widget_control, state.height_w, SET_VALUE=s.height_fact
endif else begin
  s.width_fact = s.fixed_facts(0) * s.height_fact
  widget_control, state.width_w, SET_VALUE=s.width_fact
endelse
endif

```

```

; Update Hardware offsets
widget_control, state.hw_lr_offs_w, GET_VALUE=tmp_val
hw_offs(0)=float(tmp_val) / 10.
widget_control, state.hw_tb_offs_w, GET_VALUE=tmp_val
hw_offs(1)=float(tmp_val) / 10.
preview, CENTER=s.centered

```

```

widget_control, stash, SET_UVALUE=state, /NO_COPY
return

```

quit:

END


```
PRO ps_paper, GROUP = GROUP, COLORS=COLORS, MODAL=MODAL, FONT=FONT, $
    TITLE = TITLE, HARDWARE_OFFS=HARDWARE_OFFS, WIN_SIZES=WIN_SIZES
```

```
COMMON ps_paper_com, s, r_sizes, hw_offs, v_sizes, clrs
```

```
clrs = { back_gr:fix(!d.n_colors / 2 -1), $
        paper:fix(!d.n_colors -1),      $
        text:0 }
```

```
if ( keyword_set(COLORS) ) then begin
    clrs.back_gr = colors(0)
    clrs.paper   = colors(1)
    clrs.text    = colors(2)
endif
```

```
if ( not keyword_set(FONT) ) then begin
CASE !d.name OF
```

```
    'WIN': font = 'COURIER*BOLD*8'
```

```
    'X': font = '8x13bold'
```

```
ELSE: message, 'Set keyword font for correct font name for '+!d.name+'!
```

```
ENDCASE
```

```
endif
```

```
s = { fixed_facts:[1.0,1.0], $
      w_fact:7,           $
      w_font:font,        $
      orient_p:0,         $
      size_p:0,           $
      width_fact:0.0,      $
      height_fact:0.0,     $
      loffset:0.0,        $
      boffset:0.0,        $
      centered:1,         $
      color:0,            $
      encapsulated:0 }
```

```
r_sizes = { a4:[21.0,29.7], $      ; Real paper sizes
            a3:[42.0,59.4] }
```

```
if not ( keyword_set(TITLE) ) then title = 'Postscript Settings...'
```

```

if not ( keyword_set(HARDWARE_OFFS) ) then $
    hw_offs=[0.0,0.0] $
else hw_offs=hardware_offs

; Initial virtual sizes
v_sizes = {a4:[r_sizes.a4(0)- hw_offs(0)*2,r_sizes.a4(1)- hw_offs(1)*2], $
           a3:[r_sizes.a3(0)- hw_offs(0)*2,r_sizes.a3(1)- hw_offs(1)*2] }

if keyword_set(WIN_SIZES) then begin
    win_sizes=float(win_sizes)
    if ( win_sizes(0) GE win_sizes(1) ) then begin
        s.fixed_facts(1) = float(win_sizes(1) / win_sizes(0))
    endif else begin
        s.fixed_facts(0) = float(win_sizes(0) / win_sizes(1))
    endelse
endif

; Inititally width_fact and height_fact are equal to the fixed_facts
; which are both 1.0 if not related window sizes are passed through
; the WIN_SIZES keyword
s.width_fact = s.fixed_facts(0)
s.height_fact = s.fixed_facts(1)

if (XREGISTERED('PS_PAPER')) then return ; Only one copy at a time

ps_paper_base = widget_base(/column, title=title )
row=lonarr(9)
for i=0, 8 do $
    row(i) = widget_base(ps_paper_base, /ROW, YPAD=0, XPAD=0 )

tmp_r_sizes=r_sizes.(s.size_p)
show_w = widget_draw(row(0), XSIZE=tmp_r_sizes(1)*s.w_fact,$
                     YSIZE=tmp_r_sizes(1)*s.w_fact )

junk = widget_label(row(1), VALUE='Printer Hardware Offsets', $
                    FONT=s.w_font )

; IDL-Windows 3.5.1:
; As it wasn't possible to entry the decimal point into
; the cw_field ( /FLOATING) integer / 10. is taken using mm instead of cm
hw_lr_offs_w = cw_field(row(2), TITLE='L/R Offset (mm): ', $
                       /INTEGER, FONT=s.w_font,XSIZE=4, $
                       FIELDFONT=s.w_font, $
                       /RETURN_EVENTS, $
                       VALUE=fix(hw_offs(0) * 10.) )

```

```

hw_tb_offs_w = cw_field(row(3), TITLE='T/B Offset (mm): ', $
                        /INTEGER, FONT=s.w_font, XSIZE=4, $
                        FIELDFONT=s.w_font, $
                        /RETURN_EVENTS, $
                        VALUE=fix(hw_offs(1) * 10.) )

orient_w = CW_BGROUPE(row(4), ['Portrait','Landscape'] , /ROW, /EXCLUSIVE, $
                        LABEL_TOP='Orientation', FONT=s.w_font, /NO_RELEASE, $
                        SET_VALUE=s.orient_p )

size_w = CW_BGROUPE(row(5), tag_names(r_sizes), /ROW, /EXCLUSIVE, $
                        LABEL_TOP='Paper Size', FONT=s.w_font, /NO_RELEASE, $
                        SET_VALUE=s.size_p )

if ( s.fixed_facts(0) NE 1.0 ) then $
    title='Width, fixed relation!' $
else title='Width'
width_w = cw_slider(row(6),TITLE=title, $
                    /EDIT, /DRAG, MINIMUM=0.1, MAXIMUM=1.0, $
                    FORMAT='(F4.2)', VALUE=s.fixed_facts(0) )

if ( s.fixed_facts(1) NE 1.0 ) then $
    title='Height, fixed relation!' $
else title='Height'
height_w = cw_slider(row(7),TITLE=title, $
                    /EDIT, /DRAG, MINIMUM=0.1, MAXIMUM=1.0, $
                    FORMAT='(F4.2)', VALUE=s.fixed_facts(1) )

center_w = widget_button(row(8), VALUE=' Centered ', FONT=s.w_font )

ok_w = widget_button(row(8),value='Ok', $
                    FONT=s.w_font )

state = { ps_paper_base:ps_paper_base, $
          hw_lr_offs_w:hw_lr_offs_w, $
          hw_tb_offs_w:hw_tb_offs_w, $
          orient_w:orient_w, $
          size_w:size_w, $
          width_w:width_w, $
          height_w:height_w, $
          center_w:center_w, $
          show_w:show_w, $
          show_w_id:OL, $
          ok_w:ok_w }

```

```
widget_control, ps_paper_base, /REALIZE
```

```
widget_control, state.show_w, GET_VALUE=temp  
state.show_w_id=temp  
wset, state.show_w_id
```

```
widget_control, widget_info(ps_paper_base, /CHILD), SET_UVALUE=state, $  
/NO_COPY
```

```
preview, CENTER=s.centered
```

```
XMANAGER, 'PS_PAPER', ps_paper_base, MODAL=MODAL, GROUP_LEADER = GROUP
```

```
end
```
