

PlotDwgs

```

;-----
; Program Name: PlotDwgs.lsp [PlotDwgs R3]
; Created By: Terry Miller (Email: terrycadd@yahoo.com)
;             (URL: http://web2.airmail.net/terrycad)
; Date Created: 10-20-03
; Function: PlotDwgs is a plot utility program with several unique options.
;           Among its main features are plotting all open drawings, and
;           plotting a folder of user selected drawings. Drawings may be
;           plotted to a specified size, or by selecting the "Varies"
option,
;           the program determines the correct paper size to plot. When the
;           program is first run on a new layout tab, it defaults to the
;           correct settings per that layout. If you make selection changes
;           while on that layout, it stores and uses the previous
information
;           when you run it again. Also included is the option of plotting
;           all layouts in reverse order, and plotting a folder of user
;           selected drawings in reverse order. The associated files are
;           PlotDwgs.lsp, PlotDwgs.dcl and PlotDwgs.dvb.
;-----

```

```

; Shortcuts: PD - Shortcut for PlotDwgs function.
;            PF - Shortcut for PlotDwgs with Plot Folder of Drawings
defaults.

```

```

;           works best if only one drawing in the folder to plot is
open.
;-----

```

```

; Revision History
; Rev By Date Description
;-----
; 1 TM 10-20-03 Initial version
; 2 TM 5-20-07 Revised program to be easier to customize by creating the
; *PlotterInfo@ and *PlotStyles@ global list variables.
; Included plot folder and plot open drawings. Redesigned
; the dialogs and added additional features.
; 3 TM 1-20-09 Changed the temp dcl filename C:\Temp\PlotDwgs .dcl to
; C:\Temp\PlotDwgsTemp.dcl.
;-----

```

```

; Note: Change the following global lists *PlotterInfo@ and *PlotStyles@ to suit
; the specifications of each AutoCAD department. The departments are classified
; by the members, (strcase (getvar "LOGINNAME")), for each AutoCAD department.
; The Paper Size must be "A-Size", "B-Size", "C-Size", or "D-Size". The Icon
; must be "Printer" or "Plotter". Include plotter lists in the order of the
; preferred defaults from A-Size up to D-Size. Review the note following the
; lists below to determine the correct settings for each paper size. Comment
; or delete the alert message located at the end of the file.
;-----

```

```

(setq *LoginName$ (strcase (getvar "LOGINNAME")));LoginName in all caps
(cond

```

```

((member *LoginName$ (list "MARKO" "DAVOR" "DANKO"));AutoCAD department 1
(setq *PlotterInfo@ (list
;Variable Names ;Usage and

```

```

Specifications
(list "Plotter" ;Icon$ ;Icon, Printer or Plotter
"OKI C830(PCL)" ;Plotter$ ;Plotter Display Name
"\OKI C830(PCL)" ;PlotDevice$ ;Plot Device Name, per available
choices
"A-Size" ;PaperSize$ ;Paper Size, A-Size thru D-Size
"\A3 297x420" ;MediaSize$ ;Media Size, per available
choices
"N" ;Flip$ ;Flip Plot, Y or N
"C" ;Offset$ ;Offset, C or x,y coordinates
"B-Size" ;PaperSize$ ;Paper Size, A-Size thru D-Size
"\A4 210x297" ;MediaSize$ ;Media Size, per available
choices
"N" ;Flip$ ;Flip Plot, Y or N
"C" ;Offset$ ;Offset, C or x,y coordinates
);list
;Add C-Size and D-Size information as required per plotter.
;Add additional plotter lists as required per each AutoCAD

```

## PlotDwgs

```

department.
  ));list;setq
  (setq *PlotStyles@ (list "Draft.ctb" "Final.ctb" "Generic.ctb"
"\monochrome.ctb"));Revise as required.
  );case
);cond
;-----
; Note: To determine the correct settings for each paper size, first plot a
; drawing of each paper size with a border of all the paper sizes available per
; each AutoCAD department. Check the option "Save changes to layout". After
; plotting the drawing, plot the drawing again using the command line "-plot"
; version. You may copy the text screen information into NotePad for reference,
; or just copy the required information directly into the *PlotterInfo@ and
; *PlotStyles@ lists. In the following example, the lines beginning with "*"
; are the ones that you will be using to copy the information into your lists.
;
; Command: -plot
; Detailed plot configuration? [Yes/No] <No>: Y
; Enter a layout name or [?] <Layout1>:
;*Enter an output device name or [?] <\\MAIN\DRAFTING>: <-- PlotDevice$
;*Enter paper size or [?] <Letter>: <-- MediaSize$
; Enter paper units [Inches/Millimeters] <Inches>:
; Enter drawing orientation [Portrait/Landscape] <Landscape>:
;*Plot upside down? [Yes/No] <No>: <-- Flip$ (Y or N)
; Enter plot area [Display/Extents/Limits/View/Window] <Extents>:
; Enter plot scale (Plotted Inches=Drawing Units) or [Fit] <Fit>:
;*Enter plot offset (x,y) or [Center] <Center>: <-- Offset$ (x,y or C)
; Plot with plot styles? [Yes/No] <Yes>:
;*Enter plot style table name or [?] (enter . for none) <Final.ctb>: <-- (add to
*PlotStyles@ list)
; Plot with lineweights? [Yes/No] <Yes>:
;(Layouts) Scale lineweights with plot scale? [Yes/No] <No>:
;(Layouts) Plot paper space first? [Yes/No] <No>:
;(Layouts) Hide paperspace objects? [Yes/No] <Yes>:
;(Model) Enter shade plot setting [As displayed/Wireframe/Hidden/Rendered]
<Wireframe>:
; Write the plot to a file [Yes/No] <N>:
; Save changes to page setup [Yes/No]? <N>
; Proceed with plot [Yes/No] <Y>:
;
; Note: The lines above beginning with (Layouts) are only prompted when plotting
; Layouts. The line beginning (Model) is only prompted when plotting the Model
; tab.
;-----
; c:PlotDwgs - Dialog function for Plot Drawings
;-----
(defun c:PD ()(c:PlotDwgs));Shortcut
(defun c:PlotDwgs (/ Copies$ Copies@ Ctab$ CurrentDrawing Dcl_Id% Drawing$
Drawings@ First Icon$ Layout$ $Layout$ Layouts@ List@ MediaSize$
PaperSize$ $PaperSize$ PaperSizes@ PlotDevice$ PlotStyle$ PlotStyles@ Plotter$
$Plotter$ Plotters@ Return# Reverse$ Set_Vars: Size$ Sizes@)
(princ "\nPlot Drawings\n")(princ)(setvar "CMDECHO" 0)
;-----
; Set_Vars: - Set dialog tiles and variables
;-----
(defun Set_Vars: (ListName$ VarName$ / SaveVar$)
(setq SaveVar$ (eval (read VarName$)))
(if (= ListName$ "Copies@")
(progn
(set_other_intlist "Copies@" "Copies$")
(if (< (atoi Copies$) 1)
(progn
(alert "The number of copies\nmust be at least one.")
(setq Copies$ SaveVar$)
(setq Copies@ (delete_nth (- (length Copies@) 3) Copies@))
);progn
);if
(setq Copies@ (append (mapcar 'itoa (number_sort (mapcar 'atoi (cdr (cdr

```

```

PlotDwgs
(reverse Copies@)))))(list "" "Other"))
  (set_tile_list "Copies" Copies@ Copies$)
);progn
(set_list_value ListName$ VarName$)
);if
(if (and (= ListName$ "Layouts@")(/= Layout$ SaveVar$)(= Layout$ "window"))
  (progn
    (if (= PaperSize$ "Varies")
      (progn
        (setq PaperSize$ $PaperSize$
          Plotter$ $Plotter$
        );setq
        (set_tile_list "Plotters" Plotters@ Plotter$)
        (set_tile_list "PaperSizes" PaperSizes@ PaperSize$)
      );progn
    );if
    (if (/= Drawing$ "Current Drawing")
      (progn
        (setq Drawing$ "Current Drawing")
        (set_tile_list "Drawings" Drawings@ Drawing$)
      );progn
    );if
  );progn
);if
(if (and (= ListName$ "Layouts@")(/= Layout$ SaveVar$)(member Layout$
(GetLayoutList)))
  (progn
    (if (/= Drawing$ "Current Drawing")
      (progn
        (setq Drawing$ "Current Drawing")
        (set_tile_list "Drawings" Drawings@ Drawing$)
      );progn
    );if
  );progn
);if
(if (and (= ListName$ "PaperSizes@")(/= PaperSize$ SaveVar$)(= PaperSize$
"Varies")(= Layout$ "window"))
  (progn
    (setq PaperSize$ SaveVar$ SaveVar$ PaperSize$)
    (set_tile_list "PaperSizes" PaperSizes@ PaperSize$)
  );progn
);if
(if (and (= ListName$ "Plotters@")(/= Plotter$ SaveVar$)(= Plotter$
"Varies")(= Layout$ "window"))
  (progn
    (setq Plotter$ SaveVar$ SaveVar$ Plotter$)
    (set_tile_list "Plotters" Plotters@ Plotter$)
  );progn
);if
(if (and (= ListName$ "Drawings@")(/= Drawing$ "Current Drawing")(= Layout$
"window"))
  (progn
    (setq Drawing$ "Current Drawing")
    (set_tile_list "Drawings" Drawings@ Drawing$)
  );progn
);if
(if (and (= ListName$ "Plotters@")(/= Plotter$ SaveVar$))
  (if (= Plotter$ "Varies")
    (progn
      (setq PaperSize$ "Varies")
      (set_tile_list "PaperSizes" PaperSizes@ PaperSize$)
    );progn
  (progn
    (foreach List@ *PlotterInfo@
      (if (member Plotter$ List@)
        (progn
          (setq PlotDevice$ (nth 2 List@))
          (if (/= Icon$ (nth 0 List@))

```

```

PlotDwgs
      (if (= (setq Icon$ (nth 0 List@)) "Plotter")
        (PlotterIcon)
        (PrinterIcon)
      );if
    );if
  );progn
);if
);foreach
(if (= PaperSize$ "Varies")
  (setq PaperSize$ $PaperSize$)
);if
(set_tile_list "PaperSizes" PaperSizes@ PaperSize$)
);progn
);if
);if
(if (and (= ListName$ "PaperSizes@")(/= PaperSize$ SaveVar$))
  (if (= PaperSize$ "Varies")
    (progn
      (setq Plotter$ "Varies")
      (set_tile_list "Plotters" Plotters@ Plotter$)
    );progn
    (progn
      (setq Plotters@ nil)
      (foreach List@ *PlotterInfo@
        (if (member PaperSize$ List@)
          (setq Plotters@ (append Plotters@ (list (nth 1 List@))))
        );if
      );foreach
      (setq Plotters@ (append Plotters@ (list "Varies")))
      (setq Plotter$ $Plotter$)
      (if (not (member Plotter$ Plotters@))
        (setq Plotter$ (nth 0 Plotters@))
      );if
      (foreach List@ *PlotterInfo@
        (if (member Plotter$ List@)
          (progn
            (setq PlotDevice$ (nth 2 List@)
              MediaSize$ (nth 1 (member PaperSize$ List@))
            );setq
            (if (/= Icon$ (nth 0 List@))
              (if (= (setq Icon$ (nth 0 List@)) "Plotter")
                (PlotterIcon)
                (PrinterIcon)
              );if
            );if
          );progn
        );if
      );foreach
      (set_tile_list "Plotters" Plotters@ Plotter$)
    );progn
  );if
);if
(if (and (= ListName$ "Drawings@")(/= Drawing$ "Current Drawing")(member
Layout$ (GetLayoutList)))
  (progn
    (setq Layout$ $Layout$ SaveVar$ Layout$)
    (set_tile_list "Layouts" Layouts@ Layout$)
  );progn
);if
);if
(if (and (= ListName$ "Drawings@")(= Drawing$ "Folder of Drawings")(>
(length (GetDwgsList)) 1))
  (progn
    (alert (strcat "Folder of Drawings can only be run\n"
      "in a Single Document Interface. Close\n"
      "all other open drawings and try again."))
    );alert
    (setq Drawing$ "Current Drawing")
    (set_tile_list "Drawings" Drawings@ Drawing$)
  )
)

```

## PlotDwgs

```

);progn
);if
(if (or (and (/= Layout$ "Model and layouts")(/= Layout$ "All layouts"))(and
(= Drawing$ "Current Drawing")=(length (GetLayoutList)) 1)))
  (progn
    (set_tile "Reverse" "0")
    (mode_tile "Reverse" 1)
  );progn
  (progn
    (set_tile "Reverse" Reverse$)
    (mode_tile "Reverse" 0)
  );progn
);if
);if
(if (and (/= PaperSize$ "Varies")(/= Plotter$ "Varies"))
  (setq $PaperSize$ PaperSize$
    $Plotter$ Plotter$
  );setq
);if
(if (member Layout$ (list "Model" "Model and layouts" "All layouts"))
  (setq $Layout$ Layout$)
);if
);defun Set_Vars:
;-----
; Set Default Variables and List values
;-----
(setq Layouts@ (cons "window" (cons "Model" (cons "Model and layouts" (cons
"All layouts" (GetLayoutList))))))
Copies@ (list "1" "2" "3" "4" "5" "" "Other")
PlotStyles@ *PlotStyles@
Drawings@ (list "Current Drawing" "All open Drawings" "Folder of
Drawings")
Sizes@ (list "A-Size" "B-Size" "C-Size" "D-Size")
ctab$ (getvar "CTAB")
);setq
(foreach Size$ Sizes@
  (foreach List@ *PlotterInfo@
    (if (and (member Size$ List@)(not (member Size$ PaperSizes@)))
      (setq PaperSizes@ (append PaperSizes@ (list Size$)))
    );if
  );foreach
);foreach
(setq PaperSizes@ (append PaperSizes@ (list "Varies")))
(if (not *PlotDwgs@)
  (progn
    (setq Layout$ (getvar "CTAB")
      Copies$ "1"
      PlotStyle$ (nth 0 PlotStyles@)
      Drawing$ "Current Drawing"
      Reverse$ "0"
      PaperSize$ (PaperSize)
    );setq
    (if (and (not (member PaperSize$ PaperSizes@))(setq First t))
      (foreach Size$ (cdr (member PaperSize$ (reverse Sizes@)))
        (if (and (member Size$ PaperSizes@) First)
          (setq PaperSize$ Size$ First nil)
        );if
      );foreach
    );if
    (setq First t)
    (foreach List@ *PlotterInfo@
      (if (member PaperSize$ List@)
        (setq Plotters@ (append Plotters@ (list (nth 1 List@))))
      );if
    );if
    (if (and First (member PaperSize$ List@))
      (setq First nil
        Icon$ (nth 0 List@)
        Plotter$ (nth 1 List@)
        PlotDevice$ (nth 2 List@)

```

```

PlotDwgs
MediaSize$ (nth 1 (member PaperSize$ List@))
);setq
);if
);foreach
(setq Plotters@ (append Plotters@ (list "Varies")))
(setq $Layout$ "All layouts"
$PaperSize$ PaperSize$
$Plotter$ Plotter$
);setq
);progn
(progn
(if (and (= (nth 8 *PlotDwgs@) "Folder of Drawings")(> (length
(GetDwgsList) 1))
(progn
(setq *PlotDwgs@ (change_nth 8 "Current Drawing" *PlotDwgs@))
(setq *PlotDwgs@ (change_nth 9 "0" *PlotDwgs@))
(setq *PlotDwgs@ (change_nth 13 "" *PlotDwgs@))
);progn
);if
(if (or (= (nth 8 *PlotDwgs@) "All open Drawings") (= (nth 8 *PlotDwgs@
"Folder of Drawings") (= (nth 13 *PlotDwgs@) Ctab$))
(progn
(setq Icon$ (nth 0 *PlotDwgs@)
Layout$ (nth 1 *PlotDwgs@)
Copies$ (nth 2 *PlotDwgs@)
PaperSize$ (nth 3 *PlotDwgs@)
MediaSize$ (nth 4 *PlotDwgs@)
Plotter$ (nth 5 *PlotDwgs@)
PlotDevice$ (nth 6 *PlotDwgs@)
PlotStyle$ (nth 7 *PlotDwgs@)
Drawing$ (nth 8 *PlotDwgs@)
Reverse$ (nth 9 *PlotDwgs@)
$Layout$ (nth 10 *PlotDwgs@)
$PaperSize$ (nth 11 *PlotDwgs@)
$Plotter$ (nth 12 *PlotDwgs@)
);setq
(if (not (member Copies$ Copies@))
(setq Copies@ (insert_nth 10 Copies$ Copies@))
);if
(foreach List@ *PlotterInfo@
(if (member $PaperSize$ List@)
(setq Plotters@ (append Plotters@ (list (nth 1 List@))))
);if
);foreach
(setq Plotters@ (append Plotters@ (list "Varies")))
);progn
(progn
(setq Layout$ (getvar "CTAB")
Copies$ "1"
PlotStyle$ (nth 7 *PlotDwgs@)
Drawing$ "Current Drawing"
Reverse$ "0"
PaperSize$ (PaperSize)
);setq
(if (and (not (member PaperSize$ PaperSizes@))(setq First t))
(foreach Size$ (cdr (member PaperSize$ (reverse Sizes@)))
(if (and (member Size$ PaperSizes@) First)
(setq PaperSize$ Size$ First nil)
);if
);foreach
);if
(setq First t)
(foreach List@ *PlotterInfo@
(if (member PaperSize$ List@)
(setq Plotters@ (append Plotters@ (list (nth 1 List@))))
);if
(if (and First (member PaperSize$ List@))
(setq First nil

```

```

                PlotDwgs
                Icon$ (nth 0 List@)
                Plotter$ (nth 1 List@)
                PlotDevice$ (nth 2 List@)
                MediaSize$ (nth 1 (member PaperSize$ List@))
            );setq
        );if
    );foreach
    (setq Plotters@ (append Plotters@ (list "Varies")))
    (if (and (member (nth 5 *PlotDwgs@) Plotters@)(/= (nth 5 *PlotDwgs@)
"Varies")))
        (foreach List@ *PlotterInfo@
            (if (member (nth 5 *PlotDwgs@) List@)
                (setq Plotter$ (nth 5 *PlotDwgs@)
                    PlotDevice$ (nth 2 List@)
                    Icon$ (nth 0 List@)
                );setq
            );if
        );foreach
    );if
    (setq $Layout$ "All layouts"
        $PaperSize$ PaperSize$
        $Plotter$ Plotter$
    );setq
    );progn
    );if
);progn
);if
;-----
; Load Dialog PlotDwgs
;-----
(setq Dcl_Id% (load_dialog "PlotDwgs.dcl"))
(new_dialog "PlotDwgs" Dcl_Id%)
;-----
; Set Dialog Initial Settings
;-----
(set_tile "Text1" "Plot layouts")
(set_tile "Text2" "Number of copies")
(set_tile "Text3" "Paper size")
(set_tile "Text4" "Printer/Plotter")
(set_tile "Text5" "Plot style")
(set_tile "Text6" "Drawings to plot")
(set_tile_list "Layouts" Layouts@ Layout$)
(set_tile_list "Copies" Copies@ Copies$)
(set_tile_list "PaperSizes" PaperSizes@ PaperSize$)
(set_tile_list "Plotters" Plotters@ Plotter$)
(set_tile_list "PlotStyles" PlotStyles@ PlotStyle$)
(set_tile_list "Drawings" Drawings@ Drawing$)
(if (or (and (/= Layout$ "Model and layouts")(/= Layout$ "All layouts"))(and
(= Drawing$ "Current Drawing")=(length (GetLayoutList)) 1)))
    (progn
        (setq Reverse$ "0")
        (mode_tile "Reverse" 1)
    );progn
    (progn
        (set_tile "Reverse" Reverse$)
        (mode_tile "Reverse" 0)
    );progn
);if
(if (= Icon$ "Plotter")
    (PlotterIcon)
    (PrinterIcon)
);if
;-----
; Dialog Actions
;-----
(action_tile "Layouts" "(Set_Vars: \\\"Layouts@\\\" \\\"Layout$\\\" )")
(action_tile "Copies" "(Set_Vars: \\\"Copies@\\\" \\\"Copies$\\\" )")
(action_tile "PaperSizes" "(Set_Vars: \\\"PaperSizes@\\\" \\\"PaperSize$\\\" )")

```

```

PlotDwgs
(action_tile "Plotters" "(Set_Vars: \"Plotters@\" \"Plotter$\" )")
(action_tile "PlotStyles" "(Set_Vars: \"PlotStyles@\" \"PlotStyle$\" )")
(action_tile "Drawings" "(Set_Vars: \"Drawings@\" \"Drawing$\" )")
(action_tile "Reverse" "(setq Reverse$ $value)")
(setq Return# (start_dialog))
(unload_dialog Dcl_Id%)
(if (or (and (/= Layout$ "Model and layouts")(/= Layout$ "All layouts"))(and
(= Drawing$ "Current Drawing")=(length (GetLayoutList)) 1)))
  (setq Reverse$ "0")
);if
(setq *PlotDwgs@
  (list Icon$ Layout$ Copies$ PaperSize$ MediaSize$ Plotter$ PlotDevice$
  PlotStyle$ Drawing$ Reverse$ $Layout$ $PaperSize$ $Plotter$ Ctab$)
);setq
(if (= Return# 0) (exit))
(if (not (findfile "C:\\Temp\\PlotDwgs.dat"))
  (vl-mkdir "C:\\Temp"))
);if
(writePlotDwgs)
(cond
  ((= Layout$ "window")(Plotwindow))
  ((= Drawing$ "All open Drawings")(PlotOpenDwgs))
  ((= Drawing$ "Folder of Drawings")(PlotFolderDwgs))
  ((setq CurrentDrawing t)(PlotDwgs))
);cond
(princ)
);defun c:PlotDwgs
;-----
; PlotDwgs - Main function for Plot Drawings
;-----
(defun PlotDwgs (/ Copies# Ctab$ FileName% First Layout$ LayoutList@ List@
MediaSize$
PaperSize$ PaperSizes@ PlotDevice$ PlotStyle$ Pt1 Pt2 Reverse$ Size$ Sizes@
Text$)
  (if CurrentDrawing
    (princ "\nCommand:\nPlotting Drawing...\n")
    (princ "\nCommand:\nPlotting Drawings...\n"))
);if
(princ)
(ReadPlotDwgs)
(setq Layout$ (nth 1 *PlotDwgs@)
Copies# (atoi (nth 2 *PlotDwgs@))
PaperSize$ (nth 3 *PlotDwgs@)
MediaSize$ (nth 4 *PlotDwgs@)
PlotDevice$ (nth 6 *PlotDwgs@)
PlotStyle$ (nth 7 *PlotDwgs@)
Reverse$ (nth 9 *PlotDwgs@)
Sizes@ (list "A-Size" "B-Size" "C-Size" "D-Size"))
);setq
(foreach Size$ Sizes@
  (foreach List@ *PlotterInfo@
    (if (and (member Size$ List@)(not (member Size$ PaperSizes@)))
      (setq PaperSizes@ (append PaperSizes@ (list Size$)))
    );if
  );foreach
);foreach
(setq Ctab$ (getvar "CTAB"))
(if (/= (getvar "CTAB") "Model") (command "PSPACE"))
(setq Pt1 (polar (getvar "VIEWCTR") (* pi 0.5)/(getvar "VIEWSIZE") 2.0))
(setq Pt2 (polar Pt1 (* pi 1.5) (getvar "VIEWSIZE")))
(if (= Layout$ "Model and layouts")
  (setq LayoutList@ (cons "Model" (GetLayoutList)))
  (setq LayoutList@ (GetLayoutList)))
);if
(if (= Reverse$ "1")
  (setq LayoutList@ (reverse LayoutList@))
);if
(cond

```

```

PlotDwgs
((and (or (= Layout$ "Model and layouts") (= Layout$ "All layouts")) (=
PaperSize$ "Varies"))
  (repeat Copies#
    (foreach Layout$ LayoutList@
      (command "LAYOUT" "S" Layout$)
      (if (/= Layout$ "Model") (command "PSPACE"))
      (setq PaperSize$ (PaperSize))
      (if (and (not (member PaperSize$ PaperSizes@))(setq First t))
        (foreach Size$ (cdr (member PaperSize$ (reverse Sizes@)))
          (if (and (member Size$ PaperSizes@) First)
            (setq PaperSize$ Size$ First nil)
          );if
        );foreach
      );if
      (setq First t)
      (foreach List@ *PlotterInfo@
        (if (and First (member PaperSize$ List@))
          (setq First nil
            PlotDevice$ (nth 2 List@)
            MediaSize$ (nth 1 (member PaperSize$ List@))
          );setq
        );if
      );foreach
      (if (setq SS& (ssget "x" (list '(-4 . "<AND"))(cons 410 (getvar
"CTAB"))'(-4 . "<NOT"))'(0 . "VIEWPORT"))'(-4 . "NOT>"))'(-4 . "AND>"))))
        (PlotLayout Layout$ PlotDevice$ PaperSize$ MediaSize$ PlotStyle$)
        (princ (strcat "\nNothing to plot on layout " Layout$ "."))
      );if
    );foreach
  );repeat
);case
((or (= Layout$ "Model and layouts") (= Layout$ "All layouts"))
  (repeat Copies#
    (foreach Layout$ LayoutList@
      (command "LAYOUT" "S" Layout$)
      (if (/= Layout$ "Model") (command "PSPACE"))
      (if (setq SS& (ssget "x" (list '(-4 . "<AND"))(cons 410 (getvar
"CTAB"))'(-4 . "<NOT"))'(0 . "VIEWPORT"))'(-4 . "NOT>"))'(-4 . "AND>"))))
        (PlotLayout Layout$ PlotDevice$ PaperSize$ MediaSize$ PlotStyle$)
        (princ (strcat "\nNothing to plot on layout " Layout$ "."))
      );if
    );foreach
  );repeat
);case
((and (/= Layout$ "Model and layouts") (/= Layout$ "All layouts")) (=
PaperSize$ "Varies"))
  (command "LAYOUT" "S" Layout$)
  (setq PaperSize$ (PaperSize))
  (if (and (not (member PaperSize$ PaperSizes@))(setq First t))
    (foreach Size$ (cdr (member PaperSize$ (reverse Sizes@)))
      (if (and (member Size$ PaperSizes@) First)
        (setq PaperSize$ Size$ First nil)
      );if
    );foreach
  );if
  (setq First t)
  (foreach List@ *PlotterInfo@
    (if (and First (member PaperSize$ List@))
      (setq First nil
        PlotDevice$ (nth 2 List@)
        MediaSize$ (nth 1 (member PaperSize$ List@))
      );setq
    );if
  );foreach
  (if (setq SS& (ssget "x" (list '(-4 . "<AND"))(cons 410 (getvar
"CTAB"))'(-4 . "<NOT"))'(0 . "VIEWPORT"))'(-4 . "NOT>"))'(-4 . "AND>"))))
    (repeat Copies#
      (PlotLayout Layout$ PlotDevice$ PaperSize$ MediaSize$ PlotStyle$)

```

## PlotDwgs

```

);repeat
  (princ (strcat "\nNothing to plot on layout " Layout$ "."))
);if
);case
((and (/= Layout$ "Model and layouts")(/= Layout$ "All layouts"))
  (command "LAYOUT" "S" Layout$)
  (if (/= Layout$ "Model") (command "PSPACE"))
  (if (setq SS& (ssget "x" (list '(-4 . "<AND") (cons 410 (getvar
"CTAB"))'(-4 . "<NOT")'(0 . "VIEWPORT")'(-4 . "NOT>")'(-4 . "AND>"))))
    (repeat Copies#
      (PlotLayout Layout$ PlotDevice$ PaperSize$ MediaSize$ PlotStyle$)
    );repeat
  (princ (strcat "\nNothing to plot on layout " Layout$ "."))
);if
);case
);cond
(cond
  (CurrentDrawing
    (setvar "CTAB" Ctab$)
    (command "ZOOM" Pt1 Pt2)
  );case
  ((= Ctab$ "Model")
    (setvar "CTAB" "Model")
    (command "ZOOM" Pt1 Pt2)
  );case
  (t (setvar "CTAB" (nth 0 (GetLayoutList))))
);cond
(princ "\nCommand:\nPlot Drawings complete.")
(princ)
);defun PlotDwgs
;-----
; PlotLayout - Function that Plots Layouts
;-----
(defun PlotLayout (Layout$ PlotDevice$ PaperSize$ MediaSize$ PlotStyle$ / ExtMax
ExtMin
Flip$ List@ LtScale~ Offset$ Orientation$ Plotscale~ Widest~ X-Max~ X-Min~
Y-Max~ Y-Min~)
  (setq LtScale~ (getvar "LTSCALE"))
  (setq Orientation$ (GetOrientation))
  (foreach List@ *PlotterInfo@
    (if (member PlotDevice$ List@)
      (setq Flip$ (nth 2 (member PaperSize$ List@))
        Offset$ (nth 3 (member PaperSize$ List@)))
    );setq
  );if
);foreach
(if (= Layout$ "Model")
  (progn
    (setq ExtMin (getvar "EXTMIN")
      X-Min~ (nth 0 ExtMin)
      Y-Min~ (nth 1 ExtMin)
      ExtMax (getvar "EXTMAX")
      X-Max~ (nth 0 ExtMax)
      Y-Max~ (nth 1 ExtMax)
    );setq
    (if (> (- X-Max~ X-Min~) (- Y-Max~ Y-Min~))
      (setq Widest~ (- X-Max~ X-Min~))
      (setq Widest~ (- Y-Max~ Y-Min~))
    );if
    (cond;Change 0.25 to your preferred ltscale
      ((= PaperSize$ "A-Size")(setq PlotScale~ (* (/ widest~ 11.0) 0.25)))
      ((= PaperSize$ "B-Size")(setq PlotScale~ (* (/ widest~ 17.0) 0.25)))
      ((= PaperSize$ "C-Size")(setq PlotScale~ (* (/ widest~ 21.5) 0.25)))
      ((= PaperSize$ "D-Size")(setq PlotScale~ (* (/ widest~ 33.5) 0.25)))
    );cond
    (if (> PlotScale~ 0)
      (progn
        (setvar "LTSCALE" PlotScale~)

```

```

                                PlotDwgs
(command "-PLOT" "Y" Layout$ PlotDevice$ MediaSize$ "I" Orientation$
Flip$ "E" "F" Offset$ "Y" PlotStyle$ "Y" "" "N" "N" "Y")
);progn
(princ "\nNothing to plot on layout Model.")
);if
);progn
(progn
(setvar "LTSCALE" 0.25);Change 0.25 to your preferred ltscale
(command "-PLOT" "Y" Layout$ PlotDevice$ MediaSize$ "I" Orientation$ Flip$
"E" "F" Offset$ "Y" PlotStyle$ "Y" "N" "N" "N" "N" "N" "N" "Y")
);progn
);if
(command "ZOOM" "E")
(setvar "LTSCALE" LtScale~)
);defun PlotLayout
;-----
; PlotOpenDwgs - Function to Plot open drawings
;-----
(defun PlotOpenDwgs (/ Copies#)
(command "vbaload" "PlotDwgs.dvb")
(command "-vbarun" "thisdrawing.Main")
(command "vbaunload" "PlotDwgs.dvb")
(princ)
);defun PlotOpenDwgs
;-----
; ReadPlotDwgs - Reads in *PlotDwgs@ list from file
;-----
(defun ReadPlotDwgs (/ FileName% Text$)
(setq *PlotDwgs@ nil)
(if (findfile "C:\\Temp\\PlotDwgs.dat")
(progn
(setq FileName% (open "C:\\Temp\\PlotDwgs.dat" "r"))
(while (setq Text$ (read-line FileName%))
(setq *PlotDwgs@ (append *PlotDwgs@ (list Text$)))
);while
(close FileName%)
);progn
);if
(princ)
);defun ReadPlotDwgs
;-----
; WritePlotDwgs - Writes the *PlotDwgs@ list to a file
;-----
(defun WritePlotDwgs (/ FileName% Text$)
(setq FileName% (open "C:\\Temp\\PlotDwgs.dat" "w"))
(foreach Text$ *PlotDwgs@
(write-line Text$ FileName%)
);foreach
(close FileName%)
(princ)
);defun WritePlotDwgs
;-----
; RefreashPlotDwgs - Refreshes changes made to *PlotDwgs@ during a script
;-----
(defun RefreashPlotDwgs ()
(ReadPlotDwgs)
(setq *PlotDwgs@ (change_nth 2 (nth 13 *PlotDwgs@) *PlotDwgs@))
(WritePlotDwgs)
(princ)
);defun RefreashPlotDwgs
;-----
; c:PF - Shortcut for PlotDwgs with Plot Folder of Drawings defaults.
; works best if only one drawing in the folder to plot is open.
;-----
(defun c:PF ()
(if (and (not *PlotDwgs@)(findfile "C:\\Temp\\PlotDwgs.dat"))
(ReadPlotDwgs)
);if

```

## PlotDwgs

```

(if *PlotDwgs@
  (progn
    (setq *PlotDwgs@ (change_nth 8 "Folder of Drawings" *PlotDwgs@))
    (if (not (member (nth 1 *PlotDwgs@) (list "Model" "Model and layouts" "All
layouts"))))
      (setq *PlotDwgs@ (change_nth 1 "All layouts" *PlotDwgs@))
    );if
  );progn
);if
(c:PlotDwgs)
);defun c:PF
;-----
; PlotFolderDwgs - Function to Plot a folder of drawings
;-----
(defun PlotFolderDwgs (/ Copies# CurrentDwg$ Dcl_Id% DwgName$ DwgPathName$
FileName% FolderDwgs@ FolderName$ NumberDwgs# PathFilename$ Return# Reverse$
SelectAll$ SelectedDwgs@ Set_Vars: Verify_Info:)
(princ "\nCommand:\nSelect a drawing in a folder to Plot Folder:\n")(princ)
;-----
; Set_Vars: - Set dialog tiles and variables
;-----
(defun Set_Vars: (ListName$ VarName$ / SaveVar$)
  (setq SaveVar$ (eval (read VarName$)))
  (if (= VarName$ "SelectAll$")
    (progn
      (setq SelectAll$ $value)
      (if (= SelectAll$ "1")
        (setq SelectedDwgs@ FolderDwgs@)
        (setq SelectedDwgs@ (list nil)))
      );if
      (set_tile_list "FolderDwgs" FolderDwgs@ SelectedDwgs@)
    );progn
  );if
  (if (= VarName$ "Reverse$")
    (progn
      (setq Reverse$ $value)
      (setq FolderDwgs@ (reverse FolderDwgs@))
      (setq SelectedDwgs@ (reverse SelectedDwgs@))
      (set_tile_list "FolderDwgs" FolderDwgs@ SelectedDwgs@)
    );progn
  );if
  (if (= ListName$ "FolderDwgs@")
    (progn
      (set_multilist_value "FolderDwgs@" "SelectedDwgs@")
      (setq SaveVar$ SelectAll$)
      (if (equal SelectedDwgs@ FolderDwgs@)
        (setq SelectAll$ "1")
        (setq SelectAll$ "0"))
      );if
      (if (/= selectAll$ saveVar$)
        (set_tile "SelectAll" SelectAll$)
      );if
    );progn
  );if
  (set_tile_list "FolderDwgs" FolderDwgs@ SelectedDwgs@)
  (set_tile "Reverse" Reverse$)
  (set_tile_list "FolderDwgs" FolderDwgs@ SelectedDwgs@)
);defun Set_Vars:
;-----
; Verify_Info: - Verifies dialog information
;-----
(defun Verify_Info: ()
  (if (equal SelectedDwgs@ (list nil))
    (alert "select one or more drawings to plot.")
    (done_dialog 1)
  );if
);defun Verify_Info:
;-----

```

```

PlotDwgs
; Select a drawing in a folder to Plot Folder
-----
(if (not *PlotFolder$)
  (setq *PlotFolder$ (getvar "DWGPREFIX")))
);if
(if (setq PathFilename$ (getfiled " Select a drawing in a folder to Plot
Folder" *PlotFolder$ "dwg" 2))
  (setq FolderName$ (vl-filename-directory PathFilename$)
    *PlotFolder$ (strcat FolderName$ "\\"))
  );setq
  (exit)
);if
-----
; Set Default Variables and List Values
-----
(setq FolderDwgs@ (win_sort (vl-directory-files *PlotFolder$ "*.dwg" 1))
  NumberDwgs# (length FolderDwgs@)
  Copies# (atoi Copies$)
  SelectedDwgs@ (list nil)
  SelectAll$ "0"
  Reverse$ "0")
);setq
(if (and *PlotFolderDwgs@ (= (nth 0 *PlotFolderDwgs@) *PlotFolder$)
  (or (equal (nth 1 *PlotFolderDwgs@) FolderDwgs@)
    (equal (nth 1 *PlotFolderDwgs@) (reverse FolderDwgs@))))
  );and
  (setq FolderDwgs@ (nth 1 *PlotFolderDwgs@)
    SelectedDwgs@ (nth 2 *PlotFolderDwgs@)
    SelectAll$ (nth 3 *PlotFolderDwgs@)
    Reverse$ (nth 4 *PlotFolderDwgs@)
  );setq
);if
(if (> (nth 1 (DclTextwidth FolderName$)) 39.26)
  (progn
    (setq FolderName$ (strcat (substr FolderName$ 4) (substr FolderName$ 1 3)
  "..."))
    (while (> (nth 1 (DclTextwidth FolderName$)) 39.26)
      (setq FolderName$ (substr FolderName$ 2))
    );while
    (setq FolderName$ (strcat (substr FolderName$ (- (strlen FolderName$) 5))
  (substr FolderName$ 1 (- (strlen FolderName$) 6))))
  );progn
);if
-----
; Load Dialog PlotFolderDwgs
-----
(PlotFolderDcl NumberDwgs#)
(princ "\nCommand:\nMulti-select Drawings to plot:\n")(princ)
(setq Dcl_Id% (load_dialog "C:\\Temp\\PlotDwgsTemp.dcl"))
(new_dialog "PlotFolderDwgs" Dcl_Id%)
-----
; Set Dialog Initial Settings
-----
(set_tile "FolderName" FolderName$)
(set_tile_list "FolderDwgs" FolderDwgs@ SelectedDwgs@)
(set_tile "SelectAll" SelectAll$)
(set_tile "Reverse" Reverse$)
(if (= (length FolderDwgs@) 1)
  (mode_tile "Reverse" 1)
);if
(PlotterIcon)
-----
; Dialog Actions
-----
(action_tile "FolderDwgs" "(Set_Vars: \\\"FolderDwgs@\\\" \\\"SelectedDwgs@\\\" )")
(action_tile "SelectAll" "(Set_Vars: \\\"\\\" \\\"SelectAll$\\\" )")
(action_tile "Reverse" "(Set_Vars: \\\"\\\" \\\"Reverse$\\\" )")
(action_tile "accept" "(Verify_Info:)")

```

## PlotDwgs

```

(setq Return# (start_dialog))
(unload_dialog Dcl_Id%)
(setq *PlotFolderDwgs@ (list *PlotFolder$ FolderDwgs@ selectedDwgs@ selectAll$
Reverse$))
(if (= Return# 0) (exit))
;-----
; Write script file to plot drawings
;-----
(if (= (getvar "DWGTITLED") 0)
  (if (findfile "C:\\Temp\\PlotDwgs.dwg")
    (command ".SAVEAS" "" "C:\\Temp\\PlotDwgs.dwg" "Y")
    (command ".SAVEAS" "" "C:\\Temp\\PlotDwgs.dwg")
  );if
  (if (/= (getvar "DBMOD") 0)
    (command ".QSAVE")
  );if
);if
(setq CurrentDwg$ (strcat (getvar "DWGPREFIX") (getvar "DWGNAME")))
(setq FileName% (open "C:\\Temp\\PlotDwgs.scr" "w"))
(write-line "FileOpen" FileName%)
(repeat Copies#
  (foreach DwgName$ selectedDwgs@
    (setq DwgPathName$ (strcat *PlotFolder$ DwgName$))
    (write-line (strcat "\\\" DwgPathName$ "\\\"") FileName%)
    (write-line "(load \\\"PlotDwgs\\\") (PlotDwgs)" FileName%)
    (write-line "QSave FileOpen" FileName%)
  );foreach
);repeat
(write-line (strcat "\\\" CurrentDwg$ "\\\"") FileName%)
(write-line "(load \\\"PlotDwgs\\\") (RefreashPlotDwgs)" FileName%)
(write-line "(CmdMsg \\\"Plot Drawings complete.\\\")" FileName%)
(close FileName%)
(setq *PlotDwgs@ (change_nth 2 "1" *PlotDwgs@))
(setq *PlotDwgs@ (change_nth 13 Copies$ *PlotDwgs@))
(writePlotDwgs)
(command "SCRIPT" "C:\\Temp\\PlotDwgs.scr")
);defun PlotFolderDwgs
;-----
; PlotFolderDcl - Creates C:\\Temp\\PlotDwgsTemp.dcl for PlotFolderDwgs dialogs
;-----
(defun PlotFolderDcl (NumberDwgs# / DclList@ FileName% Height$)
  (if (> NumberDwgs# 30)
    (setq NumberDwgs# 30)
  );if
  (setq Height$ (rtos (+ (* (fix (- (1+ NumberDwgs#) (* NumberDwgs# 0.18751))) (/
16 13.0))0.12)2 2))
  (setq DclList@ (list
    "PlotFolderDwgs : dialog {"
    "  key = \\\"Title\\\";"
    "  label = \\\" Plot Folder of Drawings\\\";"
    "  spacer;"
    "  : boxed_column {"
    "    label = \\\"Multi-select Drawings to plot\\\";"
    "    : text {"
    "      key = \\\"FolderName\\\";"
    "      label = \\\"\\\";"
    "    }"
    "    : list_box {"
    "      multiple_select = true;"
    "      key = \\\"FolderDwgs\\\";"
    (strcat "      height = \" Height$ \";") ;Height$ changes
    "      fixed_height = true;"
    "      width = 39.42;"
    "      fixed_width = true;"
    "    }"
    "    : column {"
    "      fixed_width = true;"
    "      alignment = centered;"
  )

```

PlotDwgs

```

"      height = 1.97;"
"      fixed_height = true;"
"      spacer_0;"
"      : row {"
"        fixed_width = true;"
"        alignment = centered;"
"        : spacer {"
"          width = 3.09;"
"        }"
"        : toggle {"
"          key = \"SelectAll\";"
"          label = \"Select all\";"
"        }"
"        : toggle {"
"          key = \"Reverse\";"
"          label = \"Reverse order\";"
"        }"
"      }"
"      spacer_0;"
"    }"
"  : row {/<"
"    width = 31.26;"
"    fixed_width = true;"
"    alignment = centered;"
"    : ok_button {"
"      width = 11;"
"    }"
"    : image {"
"      key = \"iconimage\";"
"      width = 5.42;"
"      height = 2.51;"
"      fixed_width = true;"
"      fixed_height = true;"
"      aspect_ratio = 1;"
"      color = -15;"
"    }"
"    : cancel_button {"
"      width = 11;"
"    }"
"  }/>"
"}// PlotFolderDwgs"
));list;setq
(setq FileName% (open "C:\\Temp\\PlotDwgsTemp.dcl" "w"))
(foreach Text$ DclList@
  (write-line Text$ FileName%)
);foreach
(close FileName%)
);defun PlotFolderDcl
;-----
; PlotWindow - Function to Plot a window
;-----
(defun PlotWindow (/ Copies# Flip$ Layout$ List@ LtScale~ Offset$ Orientation$
Pt1 Pt2 Widest~ X-Max~ X-Min~ Y-Max~ Y-Min~)
(princ "\nSpecify window for printing")
(if (setq Pt1 (getpoint "\nSpecify first corner: "))
  (setq Pt2 (getcorner Pt1 "Specify opposite corner: "))
);if
(if (and Pt1 Pt2)
  (progn
    (command "ZOOM" Pt1 Pt2)
    (if (< (nth 0 Pt1)(nth 0 Pt2))
      (setq X-Min~ (nth 0 Pt1) X-Max~ (nth 0 Pt2))
      (setq X-Min~ (nth 0 Pt2) X-Max~ (nth 0 Pt1))
    );if
    (if (< (nth 1 Pt1)(nth 1 Pt2))
      (setq Y-Min~ (nth 1 Pt1) Y-Max~ (nth 1 Pt2))
      (setq Y-Min~ (nth 1 Pt2) Y-Max~ (nth 1 Pt1))
    );if
  )
)

```

## PlotDwgs

```

);if
(if (> (- X-Max~ X-Min~) (- Y-Max~ Y-Min~))
  (setq Widest~ (- X-Max~ X-Min~))
  (setq Widest~ (- Y-Max~ Y-Min~))
);if
(setq Copies# (atoi Copies$)
      Layout$ (getvar "CTAB")
      LtScale~ (getvar "LTSCALE"))
);setq
(cond;Change 0.25 to your preferred ltyscale
  ((= PaperSize$ "A-Size")(setvar "LTSCALE" (* (/ widest~ 11.0) 0.25)))
  ((= PaperSize$ "B-Size")(setvar "LTSCALE" (* (/ widest~ 17.0) 0.25)))
  ((= PaperSize$ "C-Size")(setvar "LTSCALE" (* (/ widest~ 21.5) 0.25)))
  ((= PaperSize$ "D-Size")(setvar "LTSCALE" (* (/ widest~ 33.5) 0.25)))
);cond
(setq Orientation$ (if (> (- X-Max~ X-Min~) (- Y-Max~ Y-Min~)) "L" "P"))
(foreach List@ *PlotterInfo@
  (if (member PlotDevice$ List@)
    (setq Flip$ (nth 2 (member PaperSize$ List@))
          Offset$ (nth 3 (member PaperSize$ List@)))
    );setq
  );if
);foreach
(if (= Layout$ "Model")
  (repeat Copies#
    (command "-PLOT" "Y" Layout$ PlotDevice$ MediaSize$ "I" Orientation$
Flip$ "w" Pt1 Pt2 "F" Offset$ "Y" PlotStyle$ "Y" "" "N" "N" "Y")
    );repeat
    (repeat Copies#
      (command "-PLOT" "Y" Layout$ PlotDevice$ MediaSize$ "I" Orientation$
Flip$ "w" Pt1 Pt2 "F" Offset$ "Y" PlotStyle$ "Y" "N" "N" "N" "N" "N" "N" "Y")
      );repeat
    );if
    (command "ZOOM" "P")
    (setvar "LTSCALE" LtScale~)
  );progn
);if
(princ)
);defun PlotWindow
;-----
; PaperSize - Determines the Paper Size of a Layout tab
;-----
(defun PaperSize (/ ExtMax ExtMin X-Max~ X-Min~ Y-Max~ Y-Min~ widest~)
  (if (/= (getvar "CTAB") "Model") (command "PSPACE"))
  (setq ExtMin (getvar "EXTMIN")
        X-Min~ (nth 0 ExtMin)
        Y-Min~ (nth 1 ExtMin)
        ExtMax (getvar "EXTMAX")
        X-Max~ (nth 0 ExtMax)
        Y-Max~ (nth 1 ExtMax))
  );setq
  (if (> (- X-Max~ X-Min~) (- Y-Max~ Y-Min~))
    (setq Widest~ (- X-Max~ X-Min~))
    (setq Widest~ (- Y-Max~ Y-Min~))
  );if
  (cond;Make adjustments as required
    (> widest~ 21.5) "D-Size"
    (> widest~ 17) "C-Size"
    (> widest~ 11) "B-Size"
    (t "A-Size")
  );cond
);defun PaperSize
;-----
; GetOrientation - Determines the Orientation of a Layout tab
;-----
(defun GetOrientation (/ ExtMax ExtMin X-Max~ X-Min~ Y-Max~ Y-Min~)
  (setq ExtMin (getvar "EXTMIN")
        X-Min~ (nth 0 ExtMin)

```

## PlotDwgs

```

Y-Min~ (nth 1 ExtMin)
ExtMax (getvar "EXTMAX")
X-Max~ (nth 0 ExtMax)
Y-Max~ (nth 1 ExtMax)
);setq
(if (> (- X-Max~ X-Min~) (- Y-Max~ Y-Min~)) "L" "P")
);defun GetOrientation
-----
; GetDwgsList - Returns a list of open drawings
; Use (length (GetDwgsList)) for the number of open drawings.
-----
(defun GetDwgsList (/ AcadObj DocsObj DwgsList@)
  (if (>= (atoi (getvar "ACADVER")) 15)
    (progn
      (setq AcadObj (vlax-get-acad-object)
            DocsObj (vlax-get-property AcadObj "Documents"))
      );setq
      (vlax-for ForItem DocsObj
        (setq DwgsList@ (cons (strcat (vlax-get-property ForItem "Path") "\\")
                              (vlax-get-property ForItem "Name")) DwgsList@))
      );vlax-for
      (setq DwgsList@ (reverse DwgsList@))
    );progn
    (setq DwgsList@ (list (strcat (getvar "DWGPREFIX") (getvar "DWGNAME"))))
  );if
  DwgsList@
);defun GetDwgsList
-----
; GetLayoutList - Returns a list of layouts in the drawing in tab order
-----
(defun GetLayoutList (/ Layouts@)
  (vlax-map-collection (vla-get-layouts (vla-get-activatedocument
(vlax-get-acad-object)))
    '(lambda (x) (setq Layouts@ (cons x Layouts@))))
  );vlax-map-collection
  (setq Layouts@ (vl-sort Layouts@ '(lambda (x y) (< (vla-get-taborder x)
(vla-get-taborder y)))))
  (vl-remove "Model" (mapcar '(lambda (x) (vla-get-name x)) Layouts@))
);defun GetLayoutList
-----
; PlotterIcon - Plotter Icon image
-----
(defun PlotterIcon ()
  (start_image "iconimage")
  (fill_image 0 0 (dimx_tile "iconimage")(dimy_tile "iconimage") -15)
  (mapcar 'vector_image; Color 151
    (list 16 16 0 2 1 2 3 3 1 2 1 1 1 1 1 4 5
4 5 12 13 14 11 8 8 8 9 7 6 8 9 10 14 15 12 13 14
15 26 21 17 18 20 19 22 24 26 26 28 27 27 28)
    (list 18 3 27 26 26 10 10 18 15 17 16 13 14 11 12 18 18
9 9 28 28 28 6 15 14 13 15 8 8 7 7 6 19 18 5 5 4
4 20 1 3 2 1 2 0 0 1 0 20 20 1 2)
    (list 16 16 0 2 1 2 3 3 1 2 1 1 1 1 1 4 5
4 5 12 13 14 11 8 8 8 9 7 6 8 9 10 14 15 12 13 14
15 26 21 17 18 20 19 22 24 26 26 28 27 27 28)
    (list 18 3 27 26 26 10 10 18 15 17 16 13 14 11 12 18 18
9 9 28 28 28 6 15 14 13 15 8 8 7 7 6 19 18 5 5 4
4 20 1 3 2 1 2 0 0 1 0 20 20 1 2)
    (list 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151
151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151
151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151 151)
  );mapcar
  (mapcar 'vector_image; Color 153
    (list 16 16 16 5 4 3 3 3 2 2 5 4 4 4 5 5 4
4 5 5 5 7 6 8 9 15 11 6 6 7 7 7 6 6 6 8
8 8 7 7 6 6 6 9 9 7 6 8 8 9 10 10 12 12 13 14
14 15 22 20 23 21 21 17 18 18 20 20 19 22 22 23 25 25 24 26
29 30 27)

```

```

PlotDwgs
12 (list 29 4 5 10 10 11 16 17 16 11 15 15 16 17 16 17 11
13 14 12 27 28 28 29 29 7 10 15 15 16 18 17 18 17 16 18
17 16 14 13 14 13 12 16 17 9 9 9 8 8 7 8 6 7 6 6
5 5 20 20 21 2 19 4 3 4 3 2 3 2 1 1 1 2 1 2
21 21 2)
4 (list 16 16 16 5 4 3 3 3 2 2 5 4 4 4 5 5 4
5 5 5 7 6 8 9 15 11 6 6 7 7 7 6 6 6 8
8 8 7 7 6 6 6 9 9 7 6 8 8 9 10 10 12 12 13 14
14 15 22 20 23 21 21 17 18 18 20 20 19 22 22 23 25 25 24 26
29 30 27)
12 (list 29 4 5 10 10 11 16 17 16 11 15 15 16 17 16 17 11
13 14 12 27 28 28 29 29 7 10 15 15 16 18 17 18 17 16 18
17 16 14 13 14 13 12 16 17 9 9 9 8 8 7 8 6 7 6 6
5 5 20 20 21 2 19 4 3 4 3 2 3 2 1 1 1 2 1 2
21 21 2)
153 (list 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153
153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153
153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153 153
153 153 153)
);mapcar
(mapcar 'vector_image; Color 155
4 (list 16 16 16 16 16 16 3 3 2 4 4 5 3 3 2 2 4
7 9 8 7 11 11 11 11 11 11 11 11 11 7 6 8 8 7 7 7 7
11 8 8 8 8 8 9 9 9 9 9 9 9 13 14 15 11 11 11
12 11 11 7 8 10 9 9 10 10 9 10 10 15 14 13 12 12 13 12
17 12 13 13 13 15 14 14 14 14 15 15 15 23 19 18 17 21 21 21
21 21 21 21 21 21 21 19 20 17 18 17 17 17 17 18 18 18 20 20
20 19 19 19 17 18 18 20 20 19 24 25 23 22 23 22 22 22 22
23 23 23 23 22 22 22 23 23 23 25 25 25 25 25 25 25 25 22 22
22 23 23 23 25 25 25 25 24 24 24 24 26 26 26 26 26 28 27 27
27 27 28 28 28 28 27 27 28 28)
13 (list 12 19 14 13 10 9 29 27 27 28 27 28 15 12 15 12 14
20 20 20 20 21 23 22 24 26 25 27 26 27 27 26 21 23 24 25
22 24 25 23 22 21 28 26 27 24 23 25 21 22 29 29 30 14 16 15
19 17 12 19 19 15 19 18 16 17 13 13 12 10 10 15 15 16 16 11
14 12 14 13 11 15 15 14 13 11 14 13 12 20 20 20 21 7 10 11
9 17 18 16 15 14 12 10 10 19 19 14 13 11 12 13 11 12 19 11
12 13 11 12 9 9 8 7 8 8 10 10 10 10 15 15 18 19 16 17
19 18 16 17 13 14 11 14 13 11 15 16 19 18 17 14 13 12 7 9
6 9 8 6 5 7 9 8 9 8 6 5 7 9 8 5 4 21 7 8
9 6 7 8 6 5 3 4 3 4)
4 (list 16 16 16 16 16 16 3 3 2 4 4 5 3 3 2 2 4
7 9 8 7 11 11 11 11 11 11 11 11 11 7 6 8 8 7 7 7 7
11 8 8 8 8 8 9 9 9 9 9 9 9 13 14 15 11 11 11
12 11 11 7 8 10 9 9 10 10 9 10 10 15 14 13 12 12 13 12
17 12 13 13 13 15 14 14 14 14 15 15 15 23 19 18 17 21 21 21
21 21 21 21 21 21 21 19 20 17 18 17 17 17 17 18 18 18 20 20
20 19 19 19 17 18 18 20 20 19 24 25 23 22 23 22 22 22 22
23 23 23 23 22 22 22 23 23 23 25 25 25 25 25 25 25 25 22 22
22 23 23 23 25 25 25 25 24 24 24 24 26 26 26 26 26 28 27 27
27 27 28 28 28 28 27 27 28 28)
13 (list 12 19 14 13 10 9 29 27 27 28 27 28 15 12 15 12 14
20 20 20 20 21 23 22 24 26 25 27 26 27 27 26 21 23 24 25
22 24 25 23 22 21 28 26 27 24 23 25 21 22 29 29 30 14 16 15
19 17 12 19 19 15 19 18 16 17 13 13 12 10 10 15 15 16 16 11
14 12 14 13 11 15 15 14 13 11 14 13 12 20 20 20 21 7 10 11
9 17 18 16 15 14 12 10 10 19 19 14 13 11 12 13 11 12 19 11
12 13 11 12 9 9 8 7 8 8 10 10 10 10 15 15 18 19 16 17
19 18 16 17 13 14 11 14 13 11 15 16 19 18 17 14 13 12 7 9
6 9 8 6 5 7 9 8 9 8 6 5 7 9 8 5 4 21 7 8
9 6 7 8 6 5 3 4 3 4)
155 (list 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155
155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155
155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155
155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155
155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155

```

PlotDwgs

```

155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155
155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155
155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155 155
155 155 155 155 155 155 155 155 155 155)
);mapcar
(mapcar 'vector_image; color 250
  (list 16 16 3 2 2 1 3 3 2 2 15 13 12 13 14 17 29
27 28 28)
  (list 20 21 28 28 29 28 14 13 13 14 20 31 30 30 30 20 22
22 23 22)
  (list 16 16 3 2 2 1 3 3 2 2 15 13 12 13 14 17 29
27 28 28)
  (list 20 21 28 28 29 28 14 13 13 14 20 31 30 30 30 20 22
22 23 22)
  (list 250 250 250 250 250 250 250 250 250 250 250 250 250 250 250 250 250 250 250
250 250 250)
);mapcar
(mapcar 'vector_image; color 251
  (list 16 1 10 11 10 10 10 10 10 10 10 10 10 10 10 12 12 15 11
10 10 12 13 14 15 24 25 24 26 26 21 17 18 20 19 26 22 23 24
24 24 24 24 25 24 24 24 24 27 27 28)
  (list 15 29 20 29 28 29 27 26 24 23 25 21 22 31 29 21 18
19 18 17 17 16 16 20 21 21 22 21 13 15 14 13 14 10 12 12 15
19 18 17 16 11 14 13 11 12 23 10 9)
  (list 16 1 10 11 10 10 10 10 10 10 10 10 10 10 10 12 12 15 11
10 10 12 13 14 15 24 25 24 26 26 21 17 18 20 19 26 22 23 24
24 24 24 24 25 24 24 24 24 27 27 28)
  (list 15 29 20 29 28 29 27 26 24 23 25 21 22 31 29 21 18
19 18 17 17 16 16 20 21 21 22 21 13 15 14 13 14 10 12 12 15
19 18 17 16 11 14 13 11 12 23 10 9)
  (list 251 251 251 251 251 251 251 251 251 251 251 251 251 251 251 251 251 251 251
251 251 251 251 251 251 251 251 251 251 251 251 251 251 251 251 251 251 251 251
251 251 251 251 251 251 251 251 251 251 251 251)
);mapcar
(mapcar 'vector_image; color 252
  (list 16 0 1 4 5 5 14 11 11 7 7 6 6 8 10 14 15
11 11 9 10 13 12 13 15 14 15 15 25 21 21 20 22 25 24 21 21
18 17 18 20 19 19 17 20 19 19 22 23 23 25 24 26 25 29 27 27)
  (list 11 28 27 26 26 27 20 28 30 28 29 29 26 29 30 31 31
11 13 12 14 10 13 12 19 12 11 9 20 20 21 21 22 22 8 6
10 10 18 18 19 18 8 9 7 9 8 7 5 6 7 6 4 23 21 5)
  (list 16 0 1 4 5 5 14 11 11 7 7 6 6 8 10 14 15
11 11 9 10 13 12 13 15 14 15 15 25 21 21 20 22 25 24 21 21
18 17 18 20 19 19 17 20 19 19 22 23 23 25 24 26 25 29 27 27)
  (list 11 28 27 26 26 27 20 28 30 28 29 29 26 29 30 31 31
11 13 12 14 10 13 12 19 12 11 9 20 20 21 21 22 22 8 6
10 10 18 18 19 18 8 9 7 9 8 7 5 6 7 6 4 23 21 5)
  (list 252 252 252 252 252 252 252 252 252 252 252 252 252 252 252 252 252 252 252
252 252 252 252 252 252 252 252 252 252 252 252 252 252 252 252 252 252 252
252 252 252 252 252 252 252 252 252 252 252 252)
);mapcar
(mapcar 'vector_image; color 254
  (list 16 16 16 16 16 3 4 5 5 15 11 11 11 10 9 8 7
7 7 6 8 8 9 9 10 9 10 12 12 12 13 13 13 14 14 14 15
15 15 17 18 19 21 21 21 17 17 17 17 18 18 18 20 20 19 19 20
19 22 22 22 23 23 23 23 25 25 24 24 24 26 30 29 30 31)
  (list 30 28 7 8 6 26 29 29 11 28 9 8 10 10 10 10 10
11 12 11 11 12 14 11 11 9 9 10 8 9 7 9 8 7 8 9 7
8 6 29 21 21 4 5 3 18 7 6 5 7 5 6 5 6 5 6 4
4 5 3 4 2 3 4 0 0 3 2 4 3 3 20 20 22 21)
  (list 16 16 16 16 16 3 4 5 5 15 11 11 11 10 9 8 7
7 7 6 8 8 9 9 10 9 10 12 12 12 13 13 13 14 14 14 15
15 15 17 18 19 21 21 21 17 17 17 17 18 18 18 20 20 19 19 20
19 22 22 22 23 23 23 23 25 25 24 24 24 26 30 29 30 31)
  (list 30 28 7 8 6 26 29 29 11 28 9 8 10 10 10 10 10
11 12 11 11 12 14 11 11 9 9 10 8 9 7 9 8 7 8 9 7
8 6 29 21 21 4 5 3 18 7 6 5 7 5 6 5 6 5 6 4
4 5 3 4 2 3 4 0 0 3 2 4 3 3 20 20 22 21)

```



```

PlotDwgs
  (set (read SentVar$) SaveVar$)
  (set_tile_list $key SubList@ SaveVar$)
);progn
);if
(princ)
);defun set_list_value
;-----
; set_other_intlist - Function to include other integer numbers to a list
; Arguments: 2
;   SentList$ = String of the list variable name
;   SentVar$ = String of the variable name
; Syntax: (set_other_intlist "ListName" "Variable")
;-----
(defun set_other_intlist (SentList$ SentVar$ / AddOther Cnt# Item Mid$ Other$
Passed
SubList@ SubVar$)
  (setq SubList@ (eval (read SentList$))
    SubVar$ (eval (read SentVar$)))
);setq
(if (= (nth (atoi $value) SubList@) "")
  (setq $value (itoa (- (length SubList@)(length (member subVar$ SubList@)))))
);if
(if (= (nth (atoi $value) SubList@) "Other")
  (progn
    (if (setq Other$ (edit_value "Enter an Integer" SubVar$))
      (setq Other$ (vl-string-trim " " Other$))
      (setq Other$ ""))
    );if
    (if (= (strcase Other$) "OTHER") (setq Other$ ""))
    (if (/= Other$ "")
      (progn
        (setq Cnt# 1 Passed t)
        (repeat (strlen Other$)
          (setq Mid$ (substr Other$ Cnt# 1))
          (if (not (member Mid$ (list "0" "1" "2" "3" "4" "5" "6" "7" "8"
"9"))))
            (setq Passed nil)
          );if
          (setq Cnt# (1+ Cnt#))
        );repeat
        (if (not Passed)
          (progn
            (alert "value must be an integer!")
            (setq other$ "")
          );progn
          );if
        );progn
      );if
    (if (/= Other$ "")
      (progn
        (setq Other$ (itoa (atoi Other$)))
        (setq AddOther t)
        (foreach Item SubList@
          (if (= Other$ Item)
            (setq $value (itoa (- (length SubList@)(length (member Item
SubList@))))
            AddOther nil)
          );if
        );foreach
        (if AddOther
          (setq SubList@ (insert_nth (- (length SubList@) 2) Other$ SubList@)
            $value (itoa (- (length SubList@)(length (member Other$
SubList@))))))
          );if
        );progn
        (setq $value (itoa (- (length SubList@)(length (member SubVar$
SubList@))))))
      );if
    );progn
  );if

```

## PlotDwgs

```

);progn
);if
(setq SubVar$ (nth (atoi $value) SubList@))
(start_list $key) (mapcar 'add_list SubList@(end_list)
(set_tile $key $value)
(set (read SentList$) SubList@)
(set (read SentVar$) SubVar$)
(princ)
);defun set_other_intlist
;-----
; set_multilist_value - Sets SentVar$ to list of the items selected in SentList$
; Arguments: 2
;   SentList$ = String of the list variable name
;   SentVar$ = String of the variable name
; Syntax: (set_multilist_value "ListName" "Variable")
;-----
(defun set_multilist_value (SentList$ SentVar$ / SubList@)
  (setq SubList@ (eval (read SentList$)))
  (set (read SentVar$) (list (nth (atoi $value) SubList@)))
  (setq $value (substr $value (+ (strlen (itoa (atoi $value))) 2)))
  (while (/= $value "")
    (set (read SentVar$) (append (eval (read SentVar$))
      (list (nth (atoi $value) SubList@)))
    );set
    (setq $value (substr $value (+ (strlen (itoa (atoi $value))) 2)))
  );while
);defun set_multilist_value
;-----
; set_tile_list - Sets a dialog popup_list or list_box tile to a list
; Arguments: 3
;   KeyName$ = Key name of tile
;   ListName@ = The list to set in tile
;   Selected = An item in the ListNames@ or a list of items selected
; Syntax: (set_tile_list "TileName" ("A" "B" "C") "B")
;         (set_tile_list "TileName" ("A" "B" "C") ("A" "C"))
; Returns: Sets Selected items in dialog popup_list or list_box tiles.
;-----
(defun set_tile_list (KeyName$ ListName@ Selected / Item)
  (start_list KeyName$ 3)
  (mapcar 'add_list ListName@)
  (end_list)
  (foreach Item (if (listp Selected) Selected (list Selected))
    (if (member Item ListName@)
      (set_tile KeyName$ (itoa (- (length ListName@) (length (member Item
ListName@))))))
  );if
);foreach
);defun set_tile_list
;-----
; edit_value - Dialog to edit a value
; Arguments: 2
;   Title$ = Dialog Title
;   Edit1$ = Edit line
; Syntax: (edit_value "Enter Other Value" "")
;-----
(defun edit_value (Title$ Edit1$ / Dcl_Id% NewText$ Return#)
  ; Set Default Variables
  (setq NewText$ Edit1$)
  ; Load Dialog
  (setq Dcl_Id% (load_dialog "PlotDwgs.dcl"))
  (new_dialog "edit_value" Dcl_Id%)
  ; Set Dialog Initial Settings
  (set_tile "Title" Title$)
  (set_tile "Value" "Value:")
  (set_tile "Edit1" Edit1$)
  ; Dialog Actions
  (action_tile "accept" "(setq NewText$ (get_tile \"Edit1\"))(done_dialog 1)")
  (action_tile "cancel" "(done_dialog 0)")

```

## PlotDwgs

```

(setq Return# (start_dialog))
; Unload Dialog
(unload_dialog Dcl_Id%)
(if (= Return# 0) (setq NewText$ nil))
NewText$
);defun edit_value
;-----
; CmdMsg - Command line message for menus and scripts
; Arguments: 1
;   Msg$ = Message to display on the command line
; Returns: Displays the message without repeating the prompt to print it.
;-----
(defun CmdMsg (Msg$)
  (princ (strcat "\nCommand:\n" Msg$))
  (princ))
);defun CmdMsg
;-----
; Change_nth - Changes the nth item in a list with a new item value.
; Arguments: 3
;   Num# = Nth number in list to change
;   Value = New item value to change to
;   OldList@ = List to change item value
; Returns: A list with the nth item value changed.
;-----
(defun Change_nth (Num# Value OldList@)
  (if (<= 0 Num# (1- (length OldList@)))
    (if (> Num# 0)
      (cons (car OldList@) (Change_nth (1- Num#) Value (cdr OldList@)))
      (cons Value (cdr OldList@)))
    );if
  OldList@
);if
);defun Change_nth
;-----
; Delete_nth - Deletes the nth item from a list.
; Arguments: 2
;   Num# = Nth number in list to delete
;   OldList@ = List to delete the nth item
; Returns: A list with the nth item deleted.
;-----
(defun Delete_nth (Num# OldList@)
  (setq Num# (1+ Num#))
  (vl-remove-if '(lambda (x) (zerop (setq Num# (1- Num#)))) OldList@))
);defun Delete_nth
;-----
; Insert_nth - Inserts a new item value into the nth number in list.
; Arguments: 3
;   Num# = Nth number in list to insert item value
;   Value = Item value to insert
;   OldList@ = List to insert item value
; Returns: A list with the new item value inserted.
;-----
(defun Insert_nth (Num# Value OldList@ / Temp@)
  (if (< -1 Num# (1+ (length OldList@)))
    (progn
      (repeat Num#
        (setq Temp@ (cons (car OldList@) Temp@)
              OldList@ (cdr OldList@))
      );setq
    );repeat
    (append (reverse Temp@) (list Value) OldList@)
  );progn
  OldList@
);if
);defun Insert_nth
;-----
; number_sort - Sorts list of numbers
; Arguments: 1

```

## PlotDwgs

```

; List@ = List of numbers
; Returns: List of sorted numbers
-----
(defun number_sort (List@ / High~ Item~ List1@ List2@ Low~ NewList@ Passed
Swap~)
  (setq Passed t)
  (if (= (type List@) 'LIST)
    (foreach Item~ List@ (if (not (numberp Item~)) (setq Passed nil))))
    (setq Passed nil)
  );if
  (if (not Passed)
    (progn (princ "\nUsage: (number_sort <list of numbers>)" (exit))
    );if
  (repeat (/ (length List@) 2)
    (setq Low~ (car List@) High~ nil NewList@ nil)
    (foreach Item~ (cdr List@)
      (and (< Item~ Low~) (setq Swap~ Low~ Low~ Item~ Item~ Swap~))
      (and (> Item~ High~) (setq Swap~ High~ High~ Item~ Item~ Swap~))
      (setq NewList@ (cons Item~ NewList@))
    );foreach
    (setq List1@ (cons Low~ List1@) List2@ (cons High~ List2@) List@ (cdr
(reverse NewList@)))
  );repeat
  (append (reverse List1@) List@ List2@)
);defun number_sort
-----
; win_sort - Windows type of sort function
; Arguments: 1
; List@ = List of strings or filenames
; Returns: List of strings sorted similar to how windows sorts files
-----
(defun win_sort (Original@ / AlphaSort@ Cnt# Compare$ First Item List@ Loop
Next$ Num# NumSort@ NumStrings@ Passed Prefix$ Prefixes@ PrefixList@
PrefixSort@ Previous$ SortLengths@ SortList@ Str$)
  (setq Passed t)
  (if (= (type Original@) 'LIST)
    (foreach Item Original@ (if (/= (type Item) 'STR) (setq Passed nil))))
    (setq Passed nil)
  );if
  (if (not Passed)
    (progn (princ "\nUsage: (win_sort <list of strings>)" (exit))
    );if
  (setq Original@ (acad_strlsort Original@))
  (setq AlphaSort@ (mapcar 'strcase Original@))
  (setq Num# 0 Next$ (chr 160)); a unique character
  (repeat (length AlphaSort@)
    (setq Previous$ Next$
      Next$ (nth Num# AlphaSort@)
      Prefix$ nil
      Cnt# 1
    );setq
    (if (not (wcmatch (substr Next$ 1 1) "#"))
      (repeat (strlen Next$)
        (setq Str$ (substr Next$ 1 Cnt#)
          Compare$ (strcat Str$ "*"))
        );setq
        (if (and (wcmatch Previous$ Compare$)(not (wcmatch (substr Str$ (strlen
Str$)) "#"))))
          (setq Prefix$ Str$)
        );if
        (setq Cnt# (1+ Cnt#))
      );repeat
    );if
    (if Prefix$
      (progn
        (setq Compare$ (strcat Prefix$ "#*"))
        (if (and (wcmatch Previous$ Compare$)(wcmatch Next$ Compare$))
          (setq Passed t)

```

PlotDwgs

```

        (setq Passed nil)
    );if
);progn
);if
(if (and Passed Prefix$ (not (member Prefix$ Prefixes@)))
    (setq Prefixes@ (append Prefixes@ (list Prefix$)))
);if
(setq Num# (1+ Num#))
);repeat
(if Prefixes@
    (progn
        (if (> (length Prefixes@) 1)
            (progn
                (setq Num# 1 List@ (cons (nth 0 Prefixes@) (append Prefixes@ (list
(last Prefixes@))))))
                (repeat (length Prefixes@)
                    (setq Compare$ (strcat (nth Num# List@) "*"))
                    (if (and (wcmatch (nth (1- Num#) List@) Compare$)(wcmatch (nth (1+
Num#) List@) Compare$))
                        (setq Prefixes@ (vl-remove (nth Num# List@) Prefixes@))
                    );if
                    (setq Num# (1+ Num#))
                );repeat
            );progn
        );if
        (setq SortLengths@ (reverse (number_sort (mapcar 'strlen Prefixes@))))
        (setq List@ Prefixes@)
        (foreach Num# SortLengths@
            (setq First t)
            (foreach Str$ List@
                (if (and (= (strlen Str$) Num#) First)
                    (setq First nil
                        List@ (vl-remove Str$ List@)
                        PrefixSort@ (append PrefixSort@ (list Str$))
                    );setq
                );if
            );foreach
        );foreach
        (setq Prefixes@ (mapcar 'list PrefixSort@))
        (setq List@ AlphaSort@ Num# 0)
        (foreach Prefix$ PrefixSort@
            (setq Compare$ (strcat Prefix$ "#*")
                PrefixList@ (nth Num# Prefixes@)
                First t
            );setq
            (foreach Str$ List@
                (if (wcmatch Str$ Compare$)
                    (progn
                        (if First
                            (setq PrefixList@ (append PrefixList@ (list (vl-position Str$
AlphaSort@)))
                                First nil
                            );setq
                        );if
                        (setq List@ (vl-remove Str$ List@)
                            Str$ (substr Str$ (1+ (strlen Prefix$)))
                            PrefixList@ (append PrefixList@ (list Str$))
                        );setq
                    );if
                );foreach
            );foreach
            (setq Prefixes@ (change_nth Num# PrefixList@ Prefixes@))
            (setq Num# (1+ Num#))
        );foreach
        (foreach PrefixList@ Prefixes@
            (setq NumStrings@ (cddr PrefixList@)
                NumSort@ (number_sort (mapcar 'atoi NumStrings@))
                List@ nil
            )

```

## PlotDwgs

```

);setq
(foreach Num# NumSort@
  (setq Loop t Cnt# 0)
  (while Loop
    (setq Str$ (nth Cnt# NumStrings@))
    (if (= (atoi Str$) Num#)
      (setq NumStrings@ (delete_nth Cnt# NumStrings@)
            Str$ (strcat (nth 0 PrefixList@) Str$)
            List@ (append List@ (list Str$))
            Loop nil)
      );setq
    );if
    (setq Cnt# (1+ Cnt#))
  );while
);foreach
(setq Num# (nth 1 PrefixList@) SortList@ Original@)
(foreach Str$ List@
  (setq Str$ (nth (vl-position Str$ AlphaSort@) Original@))
  (setq SortList@ (change_nth Num# Str$ SortList@))
  (setq Num# (1+ Num#))
);foreach
(setq Original@ SortList@)
);foreach
);progn
);if
(foreach Str$ AlphaSort@
  (if (wcmatch (substr Str$ 1 1) "#")
    (setq NumStrings@ (append NumStrings@ (list Str$)))
  );if
);foreach
(if NumStrings@
  (progn
    (setq NumSort@ (number_sort (mapcar 'atoi NumStrings@))
          List@ nil)
    );setq
    (foreach Num# NumSort@
      (setq Loop t Cnt# 0)
      (while Loop
        (setq Str$ (nth Cnt# NumStrings@))
        (if (= (atoi Str$) Num#)
          (setq NumStrings@ (delete_nth Cnt# NumStrings@)
                List@ (append List@ (list Str$))
                Loop nil)
          );setq
        );if
        (setq Cnt# (1+ Cnt#))
      );while
    );foreach
    (setq Num# 0 SortList@ Original@)
    (foreach Str$ List@
      (setq Str$ (nth (vl-position Str$ AlphaSort@) Original@))
      (setq SortList@ (change_nth Num# Str$ SortList@))
      (setq Num# (1+ Num#))
    );foreach
    (setq Original@ SortList@)
  );progn
);if
Original@
);defun win_sort
;-----
; DclTextwidth - List of the width in pixels and the dcl width of a string
; Arguments: 1
; Str$ = String
; Returns: List of the width in pixels and the dcl width of a string.
;-----
(defun DclTextwidth (Str$ / Cnt# Mid$ Pixels# Pixelwidth~)
  (setq Cnt# 1 Pixels# 0 Pixelwidth~ 0)
  (if (= (type Str$) 'STR)

```

PlotDwgs

```

(repeat (strlen Str$)
  (setq Mid$ (substr Str$ Cnt# 1))
  (cond
    ((member Mid$ (list "@" "w"))
      (setq Pixels# (+ Pixels# 11));11 Pixels
    );case
    ((= Mid$ "M")
      (setq Pixels# (+ Pixels# 9));9 Pixels
    );case
    ((member Mid$ (list "%" "D" "G" "H" "N" "O" "Q" "R" "U" "m" "w"))
      (setq Pixels# (+ Pixels# 8));8 Pixels
    );case
    ((member Mid$ (list "#" "A" "B" "C" "E" "K" "P" "S" "T" "V" "X" "Y" "Z"
"~"))
      (setq Pixels# (+ Pixels# 7));7 Pixels
    );case
    ((member Mid$ (list "$" "&" "+" "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
"<" "="
">" "?" "F" "L" "^\\" "a" "b" "c" "d" "e" "g" "h" "k" "n" "o" "p"
"q" "u" "v"))
      (setq Pixels# (+ Pixels# 6));6 Pixels
    );case
    ((member Mid$ (list (chr 34) "/" "j" (chr 92) "s" "x" "y" "z"))
      (setq Pixels# (+ Pixels# 5));5 Pixels
    );case
    ((member Mid$ (list "*" "{" "}"))
      (setq Pixels# (+ Pixels# 4));4 Pixels
    );case
    ((member Mid$ (list " " "!" "(" ")" ", " "-" "." ":" ";" "I" "[" "]" "`"
"f" "r" "t"))
      (setq Pixels# (+ Pixels# 3));3 Pixels
    );case
    ((member Mid$ (list "" "i" "j" "l" "|"))
      (setq Pixels# (+ Pixels# 2));2 Pixels
    );case
    ((member (ascii Mid$) (list 198 230))
      (setq Pixels# (+ Pixels# 10));10 Pixels
    );case
    ((= (ascii Mid$) 169)
      (setq Pixels# (+ Pixels# 9));9 Pixels
    );case
    ((member (ascii Mid$) (list 174 188 189 190 208 209 210 211 212 213 214
216 217 218 219 220))
      (setq Pixels# (+ Pixels# 8));8 Pixels
    );case
    ((member (ascii Mid$) (list 192 193 194 195 196 197 199 200 201 202 203
221 222))
      (setq Pixels# (+ Pixels# 7));7 Pixels
    );case
    ((member (ascii Mid$) (list 128 162 163 164 165 167 171 172 175 177 181
182 187 191 215 223 224 225 226 227 228 229 231 232 233 234 235 240 241 242 243
244 245 246 247 248 249 250 251 252 254 255))
      (setq Pixels# (+ Pixels# 6));6 Pixels
    );case
    ((= (ascii Mid$) 253)
      (setq Pixels# (+ Pixels# 5));5 Pixels
    );case
    ((member (ascii Mid$) (list 170 176 186 237 238 239))
      (setq Pixels# (+ Pixels# 4));4 Pixels
    );case
    ((member (ascii Mid$) (list 127 129 130 131 132 133 134 135 136 137 138
139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158
159 160 161 168 173 178 179 180 183 184 185 204 205 206 207))
      (setq Pixels# (+ Pixels# 3));3 Pixels
    );case
    ((member (ascii Mid$) (list 166 236))
      (setq Pixels# (+ Pixels# 2));2 Pixels
    );case
  )
)

```

```

                                PlotDwgs
      (t
        (setq Pixels# (+ Pixels# 7));7 Pixels default
      );case
    );cond
      (setq Cnt# (1+ Cnt#))
    );repeat
  );if
  (if (> Pixels# 0)
    (setq Pixelwidth~ (atof (rtos (+ (* (1- Pixels#) (/ 1 6.0)) 0.09) 2 2)))
  );if
  (list Pixels# Pixelwidth~)
);defun DclTextwidth
;-----
; Comment or delete the alert lines below after reviewing the documentation.
; You can test the dialogs in the program without customizing the global lists
; *PlotterInfo@ and *PlotStyles@. To run the program type "PD" or "PlotDwgs".
;-----
;(alert (strcat "PlotDwgs must first be customized for the\n"
;              "AutoCAD users and the printers and plotters\n"
;              "that they are using. Review the note and\n"
;              "documentation on changing the global lists\n"
;              "*PlotterInfo@ and *PlotStyles@. To run the\n"
;              "program type \"PD\" or \"PlotDwgs\".")
);alert
;-----
(vl-load-com)(princ);End of PlotDwgs.lsp

```