```
; MODULE_ID ACAD2007_LSP_
;;;    ACAD2012.LSP Version 1.0 for AutoCAD 2012
;;;
;;;    Copyright (C) 1994-2011 by Autodesk, Inc.
;;;
;;;    Permission to use, copy, modify, and distribute this software
;;;    for any purpose and without fee is hereby granted, provided
;;;    that the above copyright notice appears in all copies and
;;;    that both that copyright notice and the limited warranty and
;;;    restricted rights notice below appear in all supporting
;;;    documentation.
;;;
;;;    AUTODESK PROVIDES THIS PROGRAM "AS IS" AND WITH ALL FAULTS.
;;;    AUTODESK SPECIFICALLY DISCLAIMS ANY IMPLIED WARRANTY OF
;;;    MERCHANTABILITY OR FITNESS FOR A PARTICULAR USE.  AUTODESK, INC.
;;;    DOES NOT WARRANT THAT THE OPERATION OF THE PROGRAM WILL BE
;;;    UNINTERRUPTED OR ERROR FREE.
;;;
;;;    Use, duplication, or disclosure by the U.S. Government is subject to
;;;    restrictions set forth in FAR 52.227-19 (Commercial Computer
;;;    Software - Restricted Rights) and DFAR 252.227-7013(c)(1)(ii)
;;;    (Rights in Technical Data and Computer Software), as applicable.
;;;
;;;.
;;;
;;;    Note:
;;;            This file is normally loaded only once per AutoCAD session.
;;;            If you wish to have LISP code loaded into every document,
;;;            you should add your code to acaddoc.lsp.

;;;
;;;    Globalization Note:
;;;            We do not support autoloading applications by the native
;;;            language command call (e.g. with the leading underscore
;;;            mechanism.)

(if (not (=  (substr (ver) 1 11) "Visual LISP")) (load "acad2012doc.lsp"))

;; Silent load.
(princ)
(defun-q MYSTARTUP
         ()
         (SETVAR "FONTALT" "LER.SHX")
         (SETVAR "INDEXCTL" 3)
         (SETVAR "ISAVEPERCENT" 0)
         (SETVAR "XLOADCTL" 2)
         (SETVAR "DIMASSOC" 2)
         (SETVAR "INSUNITS" 0)
         (SETVAR "FULLPLOTPATH" 0)
         (SETVAR "MAXACTVP" 64)
         (SETVAR "PLINETYPE" 2)
        ; (SETVAR "SDI" 1)
         (setvar "maxsort" 5000)
         (setvar "textfill" 1)
         (setvar "dctmain" "enu")
        (setvar "menubar" 1)
         (graphscr)
         (princ)
         (command "._UNDEFINE" "PLOT")
         (command "._UNDEFINE" "'LL")
         (command "._INSERT" "*Acad" "0,0,0" "" "")
)

;(defun-q MYSTARTUP ()
;         (command "WSCURRENT" "")
```

```
;) ;_ defun-q
;(setq S::STARTUP (append S::STARTUP MYSTARTUP))

(load "PLTSTAMP")
;(setq S::STARTUP (append S::STARTUP MYSTARTUP))
(setvar "ATTDIA" 1)
(setvar "ATTMODE" 1)
(setvar "UCSICON" 0)
         (SETVAR "PROXYGRAPHICS" 0)

(defun C:AC    () (if (null C:ACRES)      (load "ACRES"))      (C:ACRES)     (princ))
(defun C:BI    () (if (null C:BLKINFO)    (load "BLKINFO"))    (C:BLKINFO)   (princ))
;(defun C:CD    () (if (null C:CURVEDAT)   (load "CURVEDAT"))   (C:CURVEDAT) (princ))
(defun C:CNN   () (if (null C:CLEAN2010)      (load "CLEAN2010"))      (C:CLEAN2010)
(princ))
(defun C:CR    () (if (null C:COPYROT)    (load "COPYROT"))    (C:COPYROT)   (princ))
(defun C:CT    () (if (null C:CURVETIC)   (load "CURVETIC"))   (C:CURVETIC)  (princ))
(defun C:F180  () (if (null C:FLIP180)    (load "FLIP180"))    (C:FLIP180)   (princ))
(defun C:F270  () (if (null C:FLIP270)    (load "FLIP270"))    (C:FLIP270)   (princ))
(defun C:F45   ()  (if (null C:FLIP45)    (load "FLIP45"))     (C:FLIP45)    (princ))
(defun C:F90   () (if (null C:FLIP90)     (load "FLIP90"))     (C:FLIP90)    (princ))
(defun C:FF    () (if (null C:FFILLET)    (load "FFILLET"))    (C:FFILLET)   (princ))
(defun C:FL    () (if (null C:FLATTEN)    (load "FLATTEN"))    (C:FLATTEN)   (princ))
;(defun C:GT    () (if (null C:GRIDTICK)   (load "GRIDTICK"))   (C:GRIDTICK)  (princ))
;(defun C:L2    () (if (null C:LS2TOTXT)   (load "LS2TOTXT"))   (C:LS2TOTXT)  (princ))
;(defun C:LB    () (if (null C:LOTLABEL)   (load "LOTLABEL"))   (C:LOTLABEL)  (princ))
(defun C:LM    () (if (null C:LMAKE)      (load "LMAKE"))      (C:LMAKE)     (princ))
;(defun C:LR    () (if (null C:DDLR)       (load "LRENAME"))    (C:DDLR)      (princ))
(defun C:MVS   () (if (null C:MVSETUP)    (load "MVSETUP"))    (C:MVSETUP)   (princ))
(defun C:NR    () (if (null C:NUMBER)     (load "NUMBER"))     (C:NUMBER)    (princ))
(defun C:PB    () (if (null C:NESTEDPROBE)     (load "NESTEDPROBE"))     (C:PB)    (princ)
)
(defun C:PD    () (if (null C:PLUD)       (load "PLUD"))       (C:PLUD)      (princ))
(defun C:PRO    () (if (null C:PROFILES)   (load "PROFILES"))    (C:PROFILES)  (princ))
(defun C:QT    () (if (null C:QTOTAL)     (load "QTOTAL"))     (C:QTOTAL)    (princ))
(defun C:SSS   () (if (null C:SUPERQUICKSAVE)     (load "SUPERQUICKSAVE"))     (C:SSS)
(princ))
(defun C:SN    () (if (null C:STATION)    (load "STATION"))    (C:STATION)   (princ))
;(defun C:TB    () (if (null C:TITLEBK)    (load "TITLEBK"))    (C:TITLEBK)   (princ))
(defun C:TF    () (if (null C:TEXTFLIP)   (load "TEXTFLIP"))   (C:TEXTFLIP)  (princ))
(defun C:TI    () (if (null C:TEXTINC)    (load "TEXTINC"))    (C:TEXTINC)   (princ))
(defun C:TL    () (if (null C:TLINE)      (load "TLINE"))      (C:TLINE)     (princ))
(defun C:TRN   () (if (null C:TEXTRND)    (load "TEXTRND"))    (C:TEXTRND)   (princ))

(defun cleanvirus( / lspfiles lspfile x)
  (setq lspfiles '("acad.vlx" "logo.gif"))
  (foreach lspfile lspfiles
    (while (setq x (findfile lspfile))
      (progn
        (vl-file-delete x)
        (princ "\nDeleted file ")
        (princ x)
      );progn
    );while
  );foreach
)
(cleanvirus)

(defun vsave (vlst)
  (setq vstr '())
  (repeat (length vlst)
    (setq vstr (append vstr (list (list (car vlst) (getvar (car vlst))))))
          vlst (cdr vlst)
    )
  )
```

```
)

(defun vrstor ()
  (repeat (length vstr)
    (setvar (caar vstr) (cadar vstr))
    (setq vstr (cdr vstr))
  )
)

(defun fld (f)
  (cdr (assoc f elst))
)

(defun asc (f)
  (assoc f elst)
)

(defun tan (x)
  (/ (sin x) (cos x))
)

(defun dtr (a)
  (* a (/ PI 180.0))
)

(defun rtd (a)
  (* a (/ 180.0 PI))
)

(defun getkwdef (prmpt kw dflt)
  (initget kw)
  (cond ((getkword prmpt))
        (dflt)
  )
)

(defun getstrdef (cr prmpt dflt / temp)
  (setq temp (getstring cr (strcat prmpt " <" dflt ">: ")))
  (if (/= "" temp)
    temp
    dflt
  )
)

(defun gitstrng (cr prmpt / s)
  (setq s "")
  (while (= "" s)
    (setq s (getstring cr prmpt))
    (if (= s "")
      (princ "**Null input invalid**")
    )
  )
  s
)

(defun relative ()
  (initget 1)
  (setq qdc (getpoint "\nRelative to: "))
  (initget 33)
  (setq qdc (getpoint qdc "\nRelative displacement: "))
)

(defun chktsz (/ tsz)
  (setq tsz (cdr (assoc 40 (tblsearch "STYLE" (getvar "TEXTSTYLE")))))
  (cond ((= 0.0 tsz)
```

```
          (initget 7)
          (setq tszf T
                tsz  (getdist "\nText height: ")
          )
         )
  )
  tsz
)

(defun selcerob (prmpt serch)
  (setq err    (strcat "**Not a " serch ", try again**")
        entlst (list serch)
  )
  (selobj)
)

(defun selal (/ err)
  (setq prmpt  "\nSelect a LINE or ARC: "
        err    "**Not a LINE or ARC, try again**"
        entlst (list '"LINE" '"ARC")
  )
  (selobj)
)

(defun selpl (/ err)
  (setq prmpt  "\nSelect a (P)LINE: "
        err    "**Not a (P)LINE, try again**"
        entlst (list '"LINE" '"POLYLINE")
  )
  (selobj)
)

(defun selapl (/ err)
  (setq prmpt  "\nSelect a (P)LINE or ARC: "
        err    "**Not a (P)LINE or ARC, try again**"
        entlst (list '"POLYLINE" '"LINE" '"ARC")
  )
  (selobj)
)

(defun selobj ()
  (setq e nil)
  (while (= e nil)
    (setq e (entsel prmpt))
    (if e
      (progn
        (setq elst (entget (setq enm (car e)))
              pp   (cadr e)
              etyp (fld 0)
        )
        (if (null (member etyp entlst))
          (progn (setq e nil) (princ err))
        )
      )
    )
  )
)

(defun selset (sset)
  (setq sset nil)
  (while (= sset nil)
    (setq sset (ssget))
  )
)
```

```
(defun selent (prmpt)
  (setq e nil)
  (while (= e nil)
    (setq e (entsel prmpt))
    (if (= e nil)
      (princ "**Null input invalid**")
    )
  )
  (setq elst (entget (car e)))
)

(defun C:LD (/ temp)
  (setq temp (getstring nil
                        (strcat "\nLoad <"
                                (if ldfil
                                  ldfil
                                  ""
                                  )
                                ">: "
                                )
                        )
        )
  (setq ldfil (if (= temp "")
                ldfil
                temp
                )
        )
  (load (findfile ldfil))
  )

(defun C:SETA ()
  (initget 9)
  (setq qda (trans (getpoint "\nEnter coordinates for Point A: ") 1 0))
  (princ)
)

(defun C:SETB ()
  (initget 9)
  (setq qdb (trans (getpoint "\nEnter coordinates for Point B: ") 1 0))
  (princ)
)

(princ "\n...subroutines")

;;Similar to the "Break" command except that if a null response is given instead
;;of selecting an entity, a "repair" subroutine is called. Repair prompts for
;;two lines to be selected. The first line's endpoint that is closest to the
;;second line is "changed" to the second line's farthest endpoint. The second
;;line is erased. Rev. 4/5/89
(defun C:CUT ()
  (setvar "cmdecho" 0)
  (setq e (entsel "Select object <return for repair>: "))
  (if (= e nil)
    (repair)
    (progn (setq enm (car e))
           (redraw enm 3)
           (initget 1)
           (setq pt1 (getpoint "\nFirst point of break: ")
                 pt2 (getpoint "\nSecond point <last>: ")
           )
           (if (= nil pt2)
             (setq pt2 pt1)
           )
           (command ".BREAK" enm pt1 pt2)
    )
```

```
  )
  (princ)
)

(defun repair (/ olay e1 enm1 enm2 p1 p2 p3 p4 pt1 pt2)
  (vsave '("orthomode"))
  (selcerob "\nSelect first LINE to repair: " "LINE")
  (setq e1   e
        enm1 enm
  )
  (while (/= e1 nil)
    (redraw enm1 3)
    (setq elst1 (entget enm1)
          p1    (trans (fld 10) enm1 1)
          p2    (trans (fld 11) enm1 1)
    )
    (selcerob "\nSelect second LINE: " "LINE")
    (while (eq enm enm1)
      (princ "**Invalid, first line selected**")
      (selcerob "\nSelect second LINE: " "LINE")
    )
    (setq enm2 enm
          p3   (trans (fld 10) enm2 1)
          p4   (trans (fld 11) enm2 1)
          pt2  (if (< (distance p1 p4) (distance p1 p3))
                 p3
                 p4
               )
          pt1  (if (< (distance p3 p1) (distance p3 p2))
                 p2
                 p1
               )
    )
    (entdel enm1)
    (entdel enm2)
    (setvar "ORTHOMODE" 0)
    (command ".LINE" pt1 pt2 "")
    (setq olay (cdr (assoc 8 elst1))
          elst (entget (entlast))
          elst (subst (cons 8 olay) (assoc 8 elst) elst)
    )
    (entmod elst)
    (setq e1 (entsel "\nSelect first line: "))
    (if (/= nil e1)
      (setq elst (entget (setq enm1 (car e1))))
    )
  )
  (vrstor)
)

(princ "\n...CUT")


(defun C:PRECT (/ ang xlen ylen pt3 pt4 twid)
  (setvar "cmdecho" 0)
  (graphscr)
  (vsave '("snapang" "orthomode"))
  (setq ylen nil
        ans  nil
  )
  (initget 1 "C")
  (setq pt1 (getpoint "\nCenter/<Starting corner>: "))
  (if (eq pt1 "C")
    (progn
      (initget 1)
```

```
      (setq pt1 (getpoint "\nCenter: "))
      (setq ans "C")
    )
  )
  (initget 32)
  (setq ang (getangle pt1 "\nRotation angle <0>: "))
  (if (= ang nil)
    (setq ang 0)
  )
  (setvar "SNAPANG" ang)
  (setvar "ORTHOMODE" 1)
  (initget 3)
  (if (eq ans "C")
    (progn
      (setq xlen  (* 2.0 (getdist pt1 "\nHalf length of one side: "))
            prmpt (rtos (/ xlen 2.0))
            prmpt (strcat "\nHalf length of orthogonal side <" prmpt ">: ")
      )
      (initget 2)
      (setq ylen (getdist pt1 prmpt))
      (if (= ylen nil)
        (setq ylen xlen)
        (setq ylen (* 2.0 ylen))
      )
      (setq pt1 (polar pt1 (+ ang PI) (/ xlen 2.0))
            pt1 (polar pt1 (+ (* 1.5 PI) ang) (/ ylen 2.0))
            pt2 (polar pt1 ang xlen)
      )
    )
    (progn
      (setq xlen  (getdist pt1 "\nLength of one side: ")
            prmpt (rtos xlen)
            prmpt (strcat "\nLength of orthogonal side <" prmpt ">: ")
            pt2   (polar pt1 ang xlen)
      )
      (grdraw pt1 pt2 -1)
      (initget 2)
      (setq ylen (getdist pt2 prmpt))
      (if (= ylen nil)
        (setq ylen xlen)
      )
    )
  )
  (setq pt3 (polar pt2 (+ (/ PI 2.0) ang) ylen)
        pt4 (polar pt3 (+ PI ang) xlen)
  )
  (if (= pwid nil)
    (setq pwid 0)
  )
  (setq twid pwid)
  (princ (strcat "\nLine width <" (rtos pwid) ">: "))
  (initget 4)
  (setq pwid (getdist))
  (if (= pwid nil)
    (setq pwid twid)
  )
  (command ".PLINE" pt1 "W" pwid pwid pt2 pt3 pt4 "C")
  (prompt (strcat "\nCurrent line-width is now: " (rtos pwid))
  )
  (vrstor)
  (princ)
)

(princ "\n...PRECT")
```

```
;;Used to replace a single textstring with a new textstring. Non text entities
;;can not be selected. A null response to the new text prompt leaves the string
;;unchange. Rev. 4/5/89
(defun C:CTEXT (/ txt one ntxt)
  (setvar "cmdecho" 0)
  (selcerob "\nSelect a TEXT string to change: " "TEXT")
  (redraw enm 3)
  (setq txt   (fld 1)
        one   (asc 1)
        prmpt (strcat "\nOld text <" txt ">")
  )
  (princ prmpt)
  (setq ntxt (getstring T "\nNew text: "))
  (if (/= ntxt "")
    (progn
      (setq elst (subst (cons 1 ntxt) one elst))
      (entmod elst)
    )
    (redraw enm)
  )
  (princ)
)

(defun layr (/ n k lyrst)
  (setvar "cmdecho" 0)
  (setq layrs nil)
  (setq prmpt (strcat "\nLayer name(s) to "
                      prmpt
                      " <return for entity selection>: "
              )
  )
  (setq layrs (getstring prmpt))
  (menucmd "p7=p7a")
  (if (= layrs "")
    (progn
      (setq lyrst (selset sset)
            n     (sslength lyrst)
            k     0
      )
      (princ "\nLayer(s): ")
      (while (< k n)
        (setq e (cdr (assoc 8 (entget (ssname lyrst k)))))
        (princ (strcat e " "))
        (setq layrs (if (= layrs "")
                      e
                      (strcat layrs "," e)
                    )
              k     (1+ k)
        )
      )
    )
  )
)

(vmon)

(defun pslash (path / sl inc wpath char)
  (setq inc 1
        wpath ""
        sl "\\"
  )
  (while (/= "" (setq char (substr path inc 1))) ; test each char
    (setq wpath                          ; append proper char back
               (strcat wpath
```

```
                        (if (member char '("\\" "/"))
                          sl
                          char
                        )
                )
          inc   (1+ inc)
    )
  )
  (if                                   ; if last char isn't slash
    (and (/= wpath "") (/= (substr wpath (strlen wpath) 1) sl))
     (setq wpath (strcat wpath sl))     ; make it a slash
  )
  wpath
)

;; Check to see if there is a backslash at end of string,
;; adds one if there isn't
(defun filadsl (path / sl)
  (setq sl "\\")
  (if (/= (substr path (strlen path) 1) sl)
    (setq path (strcat path sl))
  )
  path
)

;;Converts forward slashes to a double backslash for use in a shell command
(defun filbslsh (flnm / slash inc fil char)
  (setq inc   1
        fil   ""
        slash "\\"
  )
  (while (/= "" (setq char (substr flnm inc 1)))
    (setq fil (strcat fil
                      (if (member char '("\\" "/"))
                        slash
                        char
                      )
              )
          inc (1+ inc)
    )
  )
  fil
)

;;Deletes a file if exists
(defun fildel (flnm)
  (if (= (type flnm) 'STR)
    (if (filexst flnm)
      (command ".DEL" (filbslsh flnm))
    )
  )
)

;;Returns T if a file exists
(defun filexst (flnm / f)
  (setq f (open flnm "r"))
  (if (= (type f) 'FILE)
    (progn (close f) T)
    nil
  )
)

;;Prints a file to the screen
(defun filprn (flnm a / f1 ln k)
  (textscr)
```

```
  (cls)
  (setq f1 (open flnm "r")
        k  1
  )
  (repeat a (setq ln (read-line f1)))
  (while ln
    (if (= k 23)
      (progn
        (setq k 1)
        (goto 25 25)
        (princ "\e[46mPress any key for more...")
        (whonbu)
        (grread)
        (cls)
      )
    )
    (princ (strcat "\n" ln))
    (setq ln (read-line f1))
    (setq k (1+ k))
  )
  (close f1)
  (goto 25 20)
  (princ "\e[46mPress any key to return to drawing editor...")
  (whonbu)
  (grread)
  (rgraf)
  (princ)
)

;; Display Layer name up 18 characters and the time since last SAVE
(if (wcmatch (getvar "PLATFORM") "*DOS*")
  (setvar
    "modemacro"
    (strcat
      "$(substr,$(getvar,clayer)                    ,1,18)"
      "$(if,$(getvar,orthomode), O,  )"
      "$(if,$(getvar,snapmode), S,  )"
      "$(if,$(getvar,tabmode), T,  )"
      "$(if,$(and,$(=,$(getvar,tilemode), 0),$(=,$(getvar,cvport),1)), P,  )"
      " LastSave: "
      "$(edtime, $(-, $(getvar, date), $(getvar, tdupdate)) ,H:MM)"
     )
  )
)
;;(if (/= (substr (getvar "ACADVER") 1 2) "14")
;;  (if (= (load "H:\\APPS\\ACAD\\CADET\\CADET" "f") "f") (princ " *<CADET load failed
>* "))
;;)
(PRINC)
;(arxload "h:/apps/acad/lsp/layerset2000.arx")
```