File created: 2-Nov-2023 23:35:15 **{WMEDLEY}<sources>ADISPLAY.;12**

edit by: rmk

changes to: (VARS ADISPLAYCOMS)
(FNS SCREENREGIONP)

previous date: 1-Mar-2023 07:49:03 {WMEDLEY}<sources>ADISPLAY.;11

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
(RPAQQ ADISPLAYCOMS
       [(COMS                                                    ; COMPILE SUPPORT
              (DECLARE%: EVAL@COMPILE DONTCOPY (FILES (LOADCOMP)
                                                      WINDOW)))
        (P (MOVD? 'NILL 'BIGBITMAPP))
        (COMS                                                    ; Interlisp-D dependent stuff.
              (EXPORT (RECORDS REGION BITMAP BITMAPWORD POSITION CURSOR MOUSEEVENT SCREENREGION SCREENPOSITION))
              (FNS SCREENREGIONP)
              (SYSRECORDS PILOTBBT \DISPLAYDATA)
              (CONSTANTS (BITSPERINTEGER 32))
              (FNS \BBTCURVEPT)
              (FNS CREATETEXTUREFROMBITMAP PRINTBITMAP PRINT-BITMAPS-NICELY PRINTCURSOR \WRITEBITMAP)
              (P (DEFPRINT 'BITMAP 'PRINT-BITMAPS-NICELY))
              (FNS \GETINTEGERPART \CONVERTTOFRACTION)
              (CONSTANTS (INTEGERBITS 12)))
        [COMS                                                    ; cursor functions not on LLDISPLAY
              (FNS CURSORP CURSORBITMAP CreateCursorBitMap)
              (EXPORT (MACROS CURSORBITMAP)
                     (CONSTANTS (HARDCURSORHEIGHT 16)
                            (HARDCURSORWIDTH 16))
                     (DECLARE%: EVAL@COMPILE (ADDVARS (GLOBALVARS CursorBitMap]
        (COMS * CARETCOMS)
        (COMS                                                    ; Region functions
              (FNS CREATEREGION REGIONP INTERSECTREGIONS UNIONREGIONS REGIONSINTERSECTP SUBREGIONP EXTENDREGION
                   EXTENDREGIONBOTTOM EXTENDREGIONLEFT EXTENDREGIONRIGHT EXTENDREGIONTOP INSIDEP STRINGREGION))
        (COMS                                                    ; line and spline drawing.
              (COMS                                              ; Brushes and brush initialization
                    (GLOBALRESOURCES \BRUSHBBT)
                    (FNS \BRUSHBITMAP \GETBRUSH \GETBRUSHBBT \InitCurveBrushes \BrushFromWidth)
                    (FNS \MAKEBRUSH.DIAGONAL \MAKEBRUSH.HORIZONTAL \MAKEBRUSH.VERTICAL \MAKEBRUSH.SQUARE
                         \MAKEBRUSH.ROUND)
                    (FNS INSTALLBRUSH)
                    (VARS \BrushNames)
                    (INITVARS (KNOWN.BRUSHES NIL)
                           (\BrushAList NIL))
                    (RECORDS BRUSHITEM)
                    (DECLARE%: DONTEVAL@LOAD DOCOPY (P (\InitCurveBrushes)))
                    (DECLARE%: DONTCOPY (GLOBALVARS \BrushAList KNOWN.BRUSHES)))
                                                               ; Lines
              (FNS \DRAWLINE.DISPLAY RELMOVETO MOVETOUPPERLEFT)
              (FNS \CLIPANDDRAWLINE \CLIPANDDRAWLINE1 \CLIPCODE \LEASTPTAT \GREATESTPTAT \DRAWLINE1
                   \DRAWLINE.UFN)
              (DECLARE%: DONTCOPY (MACROS .DRAWLINEX. .DRAWLINEY.))
                                                               ; Curves
              (FNS \DRAWCIRCLE.DISPLAY \DRAWARC.DISPLAY \DRAWARC.GENERIC \COMPUTE.ARC.POINTS
                   \DRAWELLIPSE.DISPLAY \DRAWCURVE.DISPLAY \DRAWPOINT.DISPLAY \DRAWPOLYGON.DISPLAY
                   \LINEWITHBRUSH)
              (FNS LOADPOLY PARAMETRICSPLINE \CURVE \CURVE2 \CURVEEND \CURVESLOPE \CURVESTART \FDIFS/FROM/DERIVS
                   )
              (DECLARE%: DONTCOPY                              ; Used by drawcurve
                     (EXPORT (RECORDS POLYNOMIAL SPLINE)))
              (DECLARE%: DONTCOPY (EXPORT (MACROS HALF \FILLCIRCLEBLT))
                     (MACROS \CURVEPT .SETUP.FOR.\BBTCURVEPT. \CIRCLEPTS \CURVESMOOTH))
              (FNS \FILLCIRCLE.DISPLAY \LINEBLT))
        [COMS                                                    ; making and copying bitmaps
              (FNS SCREENBITMAP BITMAPP BITMAPHEIGHT BITSPERPIXEL)
              (EXPORT (FILEPKGCOMS BITMAPS CURSORS))
              (DECLARE%: EVAL@COMPILE (EXPORT (ADDVARS (GLOBALVARS SCREENHEIGHT SCREENWIDTH ScreenBitMap]
        [COMS                                                    ; Display stream functions that are not needed in the primitive
                                                                 ; system
              (FNS DSPFILL INVERTW)
              (FNS \DSPCOLOR.DISPLAY \DSPBACKCOLOR.DISPLAY DSPEOLFN)
              (EXPORT (CONSTANTS (BLACKSHADE 65535)
                             (WHITESHADE 0))
                     (VARS (GRAYSHADE 43605))
                     (ADDVARS (GLOBALVARS GRAYSHADE)))
              (MACROS DSPRUBOUTCHAR)
              (FNS DSPCLEOL DSPRUBOUTCHAR \DSPMOVELR)
              (COMS                                              ; for cursor
                    (BITMAPS \DefaultCursor)
                    (FNS \CURSOR.DEFPRINT)
                    [DECLARE%: DONTEVAL@LOAD DOCOPY (INITVARS (DEFAULTCURSOR (CURSORCREATE \DefaultCursor NIL 0
                                                                                           15)))
```

```
                        (P (COND ((NULL \CURRENTCURSOR)
                                  (SETQ \CURRENTCURSOR DEFAULTCURSOR)))
                           (DEFPRINT 'CURSOR '\CURSOR.DEFPRINT]
                  (DECLARE%: DONTCOPY (GLOBALVARS DEFAULTCURSOR]
        [COMS                                                          ; stuff to interpret colors as textures which is needed even in
                                                                       ; system that don't have color.
              (FNS TEXTUREOFCOLOR \PRIMARYTEXTURE \LEVELTEXTURE INSURE.B&W.TEXTURE INSURE.RGB.COLOR
                   \LOOKUPCOLORNAME RGBP HLSP HLSTORGB \HLSVALUEFN)
              (VARS COLORNAMES)
              (GLOBALVARS COLORNAMES)
              (DECLARE%: DONTCOPY (GLOBALVARS BLACKSHADE16 DARKGRAY16 MEDIUMGRAY16 LIGHTGRAY16 WHITESHADE16
                                          REDTEXTURE GREENTEXTURE BLUETEXTURE))
              (UGLYVARS BLACKSHADE16 DARKGRAY16 MEDIUMGRAY16 LIGHTGRAY16 WHITESHADE16 REDTEXTURE GREENTEXTURE
                     BLUETEXTURE)
              (DECLARE%: DONTCOPY                                      ; Used by drawcurve
                     (EXPORT (RECORDS HLS RGB]
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                              (NLAML)
                                                                              (LAMA UNIONREGIONS INTERSECTREGIONS
                                                                                    ])
```

;; COMPILE SUPPORT

```
(DECLARE%: EVAL@COMPILE DONTCOPY

(FILESLOAD (LOADCOMP)
       WINDOW)
)

(MOVD? 'NILL 'BIGBITMAPP)
```

;; Interlisp-D dependent stuff.

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

(RECORD REGION (LEFT BOTTOM WIDTH HEIGHT)
       LEFT _ -16383 BOTTOM _ -16383 WIDTH _ 32767 HEIGHT _ 32767 [ACCESSFNS ((TOP (IPLUS (fetch (REGION BOTTOM)
                                                                                              of DATUM)
                                                                                       (fetch (REGION HEIGHT)
                                                                                          of DATUM)
                                                                                       -1))
                                                                              (PTOP (IPLUS (fetch (REGION BOTTOM)
                                                                                             of DATUM)
                                                                                      (fetch (REGION HEIGHT)
                                                                                         of DATUM)))
                                                                              (RIGHT (IPLUS (fetch (REGION LEFT)
                                                                                             of DATUM)
                                                                                      (fetch (REGION WIDTH)
                                                                                         of DATUM)
                                                                                      -1))
                                                                              (PRIGHT (IPLUS (fetch (REGION LEFT)
                                                                                             of DATUM)
                                                                                      (fetch (REGION WIDTH
                                                                                                    )
                                                                                             of DATUM]
       [TYPE? (AND (EQLENGTH DATUM 4)
                   (EVERY DATUM (FUNCTION NUMBERP]
       (SYSTEM))

(DATATYPE BITMAP ((BITMAPBASE POINTER)
                  (BITMAPRASTERWIDTH WORD)
                  (BITMAPHEIGHT WORD)
                  (BITMAPWIDTH WORD)
                  (BITMAPBITSPERPIXEL WORD))
       BITMAPBITSPERPIXEL _ 1 (BLOCKRECORD BITMAP ((BitMapHiLoc WORD)
                                                   (BitMapLoLoc WORD))
                                                                       ; overlay initial pointer

                                      )
       (SYSTEM))

(BLOCKRECORD BITMAPWORD ((BITS WORD))
       (SYSTEM))

(RECORD POSITION (XCOORD . YCOORD)
       [TYPE? (AND (LISTP DATUM)
                   (NUMBERP (CAR DATUM))
                   (NUMBERP (CDR DATUM]
       (SYSTEM))

(DATATYPE CURSOR (CUIMAGE CUMASK CUHOTSPOTX CUHOTSPOTY CUDATA)
       [ACCESSFNS ((CUBITSPERPIXEL (fetch (BITMAP BITMAPBITSPERPIXEL) of (fetch (CURSOR CUIMAGE) of DATUM]
       (SYSTEM))

(RECORD MOUSEEVENT (MOUSEX MOUSEY MOUSEBUTTONS KEYBOARD MOUSETIME)
```

```
       (SYSTEM))

(RECORD SCREENREGION (SCREEN . REGION)
       (SUBRECORD REGION)
       [TYPE? (AND (LISTP DATUM)
                   (type? SCREEN (CAR DATUM))
                   (type? REGION (CDR DATUM]
       (SYSTEM))

(RECORD SCREENPOSITION (SCREEN . POSITION)
       (SUBRECORD POSITION)
       [TYPE? (AND (LISTP DATUM)
                   (type? SCREEN (CAR DATUM))
                   (type? POSITION (CDR DATUM]
       (SYSTEM))
)

(/DECLAREDATATYPE 'BITMAP '(POINTER WORD WORD WORD WORD)
       ;; ---field descriptor list elided by lister---
       '6)

(/DECLAREDATATYPE 'CURSOR '(POINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '10)

;; END EXPORTED DEFINITIONS

(DEFINEQ

(SCREENREGIONP
  [LAMBDA (X)                                          ; Edited  2-Nov-2023 23:34 by rmk
    (CL:WHEN (type? SCREENREGION X)
           X])

)

(ADDTOVAR SYSTEMRECLST
          (DATATYPE PILOTBBT ((PBTDESTLO WORD)
                              (PBTDESTHI WORD)
                              (PBTDESTBIT WORD)
                              (PBTDESTBPL SIGNEDWORD)
                              (PBTSOURCELO WORD)
                              (PBTSOURCEHI WORD)
                              (PBTSOURCEBIT WORD)
                              (PBTSOURCEBPL SIGNEDWORD)
                              (PBTWIDTH WORD)
                              (PBTHEIGHT WORD)
                              (PBTFLAGS WORD)
                              (NIL 5 WORD)))
          (DATATYPE \DISPLAYDATA
                 (DDXPOSITION DDYPOSITION DDXOFFSET DDYOFFSET DDDestination DDClippingRegion DDFONT
                        DDSlowPrintingCase DDWIDTHSCACHE DDOFFSETSCACHE DDCOLOR DDLINEFEED DDRightMargin
                        DDLeftMargin DDScroll DDOPERATION DDSOURCETYPE (DDClippingLeft WORD)
                        (DDClippingRight WORD)
                        (DDClippingBottom WORD)
                        (DDClippingTop WORD)
                        (NIL WORD)
                        (DDHELDFLG FLAG)
                        (XWINDOWHINT XPOINTER)
                        (DDPILOTBBT POINTER)
                        DDXSCALE DDYSCALE DDCHARIMAGEWIDTHS DDEOLFN DDPAGEFULLFN DDTexture DDMICAXPOS DDMICAYPOS
                        DDMICARIGHTMARGIN DDCHARSET (DDCHARSETASCENT WORD)
                        (DDCHARSETDESCENT WORD)
                        DDCHARHEIGHTDELTA
                        (DDSPACEWIDTH WORD))))

(DECLARE%: EVAL@COMPILE

(RPAQQ BITSPERINTEGER 32)

(CONSTANTS (BITSPERINTEGER 32))
)

(DEFINEQ

(\BBTCURVEPT
  [LAMBDA (X Y BBT LEFT BRUSHWIDTH LEFTMINUSBRUSH RIGHTPLUS1 NBITSRIGHTPLUS1 TOPMINUSBRUSH DestinationBitMap
             BRUSHHEIGHT BOTTOMMINUSBRUSH TOP BRUSHBASE DESTINATIONBASE RASTERWIDTH BRUSHRASTERWIDTH
             COLORBRUSHBASE NBITS DISPLAYDATA)                    (* kbr%: "27-Aug-86 23:17")
    ;; Called by \CURVEPT macro.  Draws a brush point by bitblting BRUSHBM to point X,Y in DestinationBitMap.  BBT is a BitBlt table where
    ;; everything is already set except the source and destination addresses, width and height.  In other words, only the easy stuff
                                                                   ; set the width fields of the bbt
    [PROG (CLIPPEDTOP STY)
          [COND
```

```
            [(ILEQ Y TOPMINUSBRUSH)                          ; the top part of the brush is visible
             (SETQ CLIPPEDTOP (IPLUS Y BRUSHHEIGHT))
             (replace PBTSOURCE of BBT with BRUSHBASE)
             (freplace PBTHEIGHT of BBT with (IMIN BRUSHHEIGHT (IDIFFERENCE Y BOTTOMMINUSBRUSH]
            (T                                               ; only the bottom is visible
               (SETQ CLIPPEDTOP TOP)
               [replace PBTSOURCE of BBT with (\ADDBASE BRUSHBASE (ITIMES BRUSHRASTERWIDTH (SETQ STY
                                                                                             (IDIFFERENCE Y
                                                                                                    TOPMINUSBRUSH]
               (freplace PBTHEIGHT of BBT with (IDIFFERENCE (IMIN BRUSHHEIGHT (IDIFFERENCE Y BOTTOMMINUSBRUSH))
                                                            STY]
         (freplace PBTDEST of BBT with (\ADDBASE DESTINATIONBASE (ITIMES RASTERWIDTH (\SFInvert DestinationBitMap
                                                                                       CLIPPEDTOP]
   [COND
      (COLORBRUSHBASE [COND
                         [(ILESSP X LEFT)                    ; only the right part of the brush is visible
                                                             ; FOR NOW BRUTE FORCE WITH NBITS CHECK
                          [freplace PBTDESTBIT of BBT with (COND
                                                             ((EQ NBITS 4)
                                                              (LLSH LEFT 2))
                                                             (T (LLSH LEFT 3]
                          (freplace PBTSOURCEBIT of BBT with (IDIFFERENCE BRUSHWIDTH
                                                               (freplace PBTWIDTH of BBT
                                                                  with (COND
                                                                         ((EQ NBITS 4)
                                                                          (LLSH (IDIFFERENCE X
                                                                                             LEFTMINUSBRUSH)
                                                                                2))
                                                                         (T (LLSH (IDIFFERENCE X
                                                                                               LEFTMINUSBRUSH)
                                                                                  3]
                         (T                                  ; left edge is visible
                            [freplace PBTDESTBIT of BBT with (SETQ X (COND
                                                                       ((EQ NBITS 4)
                                                                        (LLSH X 2))
                                                                       (T (LLSH X 3]
                            (freplace PBTSOURCEBIT of BBT with 0)  ; set width to the amount that is visible
                            (freplace PBTWIDTH of BBT with (IMIN BRUSHWIDTH (IDIFFERENCE NBITSRIGHTPLUS1 X]
                                                             ; if color brush is used, the ground must be cleared before the
                                                             ; brush is put in.
             (\SETPBTFUNCTION BBT (ffetch DDSOURCETYPE of DISPLAYDATA)
                    'ERASE)
             (\PILOTBITBLT BBT 0)                            ; reset the source to point to the color bitmap.
             [COND
                ((ILEQ Y TOPMINUSBRUSH)                      ; the top part of the brush is visible
                 (freplace PBTSOURCE of BBT with COLORBRUSHBASE))
                (T                                           ; only the bottom is visible
                   (freplace PBTSOURCE of BBT with (\ADDBASE COLORBRUSHBASE (ITIMES BRUSHRASTERWIDTH
                                                                                    (IDIFFERENCE Y TOPMINUSBRUSH]
             (\SETPBTFUNCTION BBT (ffetch DDSOURCETYPE of DISPLAYDATA)
                    'PAINT))
      (T (COND
            [(ILESSP X LEFT)                                 ; only the right part of the brush is visible
             (freplace PBTDESTBIT of BBT with LEFT)
             (freplace PBTSOURCEBIT of BBT with (IDIFFERENCE BRUSHWIDTH (freplace PBTWIDTH of BBT
                                                                          with (IDIFFERENCE X LEFTMINUSBRUSH]
            (T                                               ; left edge is visible
               (freplace PBTDESTBIT of BBT with X)
               (freplace PBTSOURCEBIT of BBT with 0)         ; set width to the amount that is visible
               (freplace PBTWIDTH of BBT with (IMIN BRUSHWIDTH (IDIFFERENCE RIGHTPLUS1 X]
   (\PILOTBITBLT BBT 0])

)

(DEFINEQ

(CREATETEXTUREFROMBITMAP
  [LAMBDA (BITMAP)                                           (* rrb "17-May-84 11:22")
    ;; creates a texture object from the lower left corner of a bitmap
    (OR (BITMAPP BITMAP)
        (\ILLEGAL.ARG BITMAP))
    (PROG ((H (fetch BITMAPHEIGHT of BITMAP))
           (W (fetch BITMAPWIDTH of BITMAP))
           TEXTHEIGHT TEXTURE)
          (COND
             ((AND (OR (EQ W 2)
                       (EQ W 4))
                   (OR (EQ H 2)
                       (EQ H 4)))                            ; small texture will match bitmap exactly so use integer
                                                             ; representation.
              (SETQ TEXTURE 0)
              [for X from 0 to 3 do (for Y from 0 to 3
                                       do (COND
                                             ([NOT (EQ 0 (BITMAPBIT BITMAP (IREMAINDER X W)
                                                                    (IREMAINDER Y H]
                                              (SETQ TEXTURE (LOGOR TEXTURE (\BITMASK (IPLUS (ITIMES (IDIFFERENCE
```

```
                                                                                                    3 Y)
                                                                                                  4)
                                                                                                X]
            (RETURN TEXTURE))
           ((AND (EQ W 16)
                 (ILESSP H 17))                                  ; if it is already 16 by n n<=16, use it.
            (RETURN BITMAP))
           (T                                                    ; make a 16 bit wide one.
              (SETQ TEXTURE (BITMAPCREATE 16 (IMIN H 16)))
              (for X from 0 by W to 16 do (BITBLT BITMAP 0 0 TEXTURE X 0 W H 'INPUT 'REPLACE))
              (RETURN TEXTURE])
```

(PRINTBITMAP

```
  [LAMBDA (BITMAP FILE)                                          ; Edited  1-Dec-86 16:24 by Pavel
```

;;; Writes a bitmap on a file such that READBITMAP will read it back in.

```
    (DECLARE (LOCALVARS . T))
    (PROG ((BM BITMAP))
          (COND
             ((type? BITMAP BITMAP))
             ([AND (LITATOM BITMAP)
                   (type? BITMAP (SETQ BM (EVALV BITMAP]          ; Coerce litatoms for compatibility with original specification

              )
             (T (printout T "******** " BITMAP " is not a BITMAP." T)
                (RETURN NIL)))
          (printout FILE "(" .P2 (BITMAPWIDTH BM)
                 %, .P2 (BITMAPHEIGHT BM))                        ; if the number of bits per pixel is not 1, write it out.
          (COND
             ((NEQ (BITSPERPIXEL BM)
                   1)
              (SPACES 1 FILE)
              (PRIN2 (BITSPERPIXEL BM)
                     FILE)))                                      ; Enclose in list so that compile-copying works.
          (\WRITEBITMAP BM FILE)                                  ; Now write out contents.
          (PRIN1 ")" FILE])
```

(PRINT-BITMAPS-NICELY

```
  [LAMBDA (BITMAP STREAM)                                        ; Edited 20-Mar-87 17:06 by jop
```

;;; The syntax for bitmaps is

;; #*(width height [bits-per-pixel])XXXXXX...

;;; where WIDTH and HEIGHT are the dimensions of the bitmap, BITS-PER-PIXEL can be omitted if it is equal to one, and the X's are single characters ;;; between @ and O (in ASCII), each representing four bits. There will be exactly (* (ceiling (* WIDTH BITS-PER-PIXEL) 16) 4) characters for each row ;;; of the bitmap and exactly HEIGHT rows. Note that there are no spaces allowed between the * and the (, between the) and the first X, or anywhere ;;; inside the string of X's. Also, the character after the last X must not be of type OTHER.

;;; This function "observes" *print-length*: it truncates after printing *print-length* characters in the bitmap's representation.

```
    (if (OR (NULL STREAM)
            (NULL *PRINT-ARRAY*))
        then ;; Let it be printed in the normal way, with an address.
             NIL
      else ;; Print this bitmap in the preferred way.
           (LET* ((WIDTH (BITMAPWIDTH BITMAP))
                  (HEIGHT (BITMAPHEIGHT BITMAP))
                  (BITS-PER-PIXEL (BITSPERPIXEL BITMAP))
                  (BASE (fetch BITMAPBASE of BITMAP))
                  (QUAD-CHARS-PER-ROW (FOLDHI (CL:* WIDTH BITS-PER-PIXEL)
                                              16))
                  (CHARS-SO-FAR *PRINT-LENGTH*))
                 (PRINTOUT STREAM "#*(" .P2 WIDTH " " .P2 HEIGHT)
                 (if (NEQ BITS-PER-PIXEL 1)
                     then (PRINTOUT STREAM " " .P2 BITS-PER-PIXEL))
                 (PRINTOUT STREAM ")")
                 (PROG NIL
                       [CL:MACROLET [(ELIDE? NIL `(IF (AND CHARS-SO-FAR (EQ 0 (CL:DECF CHARS-SO-FAR)))
                                                      THEN (PRINTOUT STREAM "...")
                                                           (GO OUT]
                             (CL:DOTIMES (ROW HEIGHT)
                                 (CL:DOTIMES (QUAD QUAD-CHARS-PER-ROW)
                                     (CL:WRITE-CHAR (CL:INT-CHAR (+ (LRSH (\GETBASEBYTE BASE 0)
                                                                          4)
                                                                    (CL:CHAR-INT #\@)))
                                                    STREAM)
                                     (ELIDE?)
                                     (CL:WRITE-CHAR (CL:INT-CHAR (+ (LOGAND (\GETBASEBYTE BASE 0)
                                                                            15)
                                                                    (CL:CHAR-INT #\@)))
                                                    STREAM)
                                     (ELIDE?)
```

```
                                    (CL:WRITE-CHAR (CL:INT-CHAR (+ (LRSH (\GETBASEBYTE BASE 1)
                                                                          4)
                                                                   (CL:CHAR-INT #\@)))
                                           STREAM)
                                    (ELIDE?)
                                    (CL:WRITE-CHAR (CL:INT-CHAR (+ (LOGAND (\GETBASEBYTE BASE 1)
                                                                           15)
                                                                   (CL:CHAR-INT #\@)))
                                           STREAM)
                                    (ELIDE?)
                                    (SETQ BASE (\ADDBASE BASE 1))))]
              OUT (RETURN T])
```

(**PRINTCURSOR**

```
  [LAMBDA (VAR)                                                    ; Edited  2-Dec-86 14:15 by Pavel
```

;; Writes an expression that will define the cursor value of VAR

```
    (PROG (CUR IMAGE MASK)
          (COND
             ([NOT (type? CURSOR (SETQ CUR (EVALV VAR 'PRINTCURSOR]
              (printout T "******** " VAR " is not a CURSOR." T)
              (RETURN NIL)))                                       ; write out defining form.
          (\CURSORBITSPERPIXEL CUR 1)
          (SETQ IMAGE (fetch (CURSOR CUIMAGE) of CUR))
          (SETQ MASK (fetch (CURSOR CUMASK) of CUR))
          (PRINT `(RPAQ (\, VAR)
                        (CURSORCREATE ',IMAGE ',(AND (NOT (EQ IMAGE MASK))
                                                     MASK)
                               ,(fetch (CURSOR CUHOTSPOTX)
                                       of CUR)
                               ,(fetch (CURSOR CUHOTSPOTY)
                                       of CUR)))])
```

(**\WRITEBITMAP**

```
  [LAMBDA (BITMAP FILE)                                            ; Edited  1-Dec-86 16:24 by Pavel
```

;;; writes the contents of a bitmap onto the currently open output file.

```
    (PROG (LIM (BASE (fetch BITMAPBASE of BITMAP))
               (OFD (GETSTREAM FILE 'OUTPUT))
               (W (fetch BITMAPRASTERWIDTH of BITMAP)))
          (FRPTQ (fetch BITMAPHEIGHT of BITMAP)
                 (TERPRI FILE)
                 (\BOUT OFD (CHARCODE %"))
                 (SETQ LIM (\ADDBASE BASE W))
                 (until (EQ BASE LIM) do (\BOUT OFD (IPLUS (SUB1 (CHARCODE A))
                                                           (LRSH (\GETBASEBYTE BASE 0)
                                                                 4)))
                                         (\BOUT OFD (IPLUS (SUB1 (CHARCODE A))
                                                           (LOGAND (\GETBASEBYTE BASE 0)
                                                                   15)))
                                         (\BOUT OFD (IPLUS (SUB1 (CHARCODE A))
                                                           (LRSH (\GETBASEBYTE BASE 1)
                                                                 4)))
                                         (\BOUT OFD (IPLUS (SUB1 (CHARCODE A))
                                                           (LOGAND (\GETBASEBYTE BASE 1)
                                                                   15)))
                                         (SETQ BASE (\ADDBASE BASE 1)))
                 (\BOUT OFD (CHARCODE %"])

)

(DEFPRINT 'BITMAP 'PRINT-BITMAPS-NICELY)

(DEFINEQ
```

(**\GETINTEGERPART**

```
  [LAMBDA (FRACT)                                                  (* JonL " 7-May-84 02:43")
```

;; gets the integer part of a fixed point number. The integer part has INTEGERBITS worth of significant bits the leftmost of which is sign.

```
    (PROG [HIPART (ROUNDER (COND
                             ([EQ 0 (LOGAND (fetch (FIXP HINUM) of FRACT)
                                            (CONSTANT (LLSH 1 (IDIFFERENCE BITSPERWORD (ADD1 INTEGERBITS]
                              0)
                             (T 1]
```

;; assumes that the number of significant bits --- INTEGERBITS --- is less than can fit in the high order of the two words allocated for the integer.

```
          (RETURN (COND
                     ([IGREATERP [SETQ HIPART (LRSH (fetch (FIXP HINUM) of FRACT)
                                                    (CONSTANT (IDIFFERENCE BITSPERWORD INTEGERBITS]
                                 (CONSTANT (EXPT 2 (SUB1 INTEGERBITS]
                                                                   ; the sign bit is on, make it negative.
                      (IDIFFERENCE (IDIFFERENCE HIPART (CONSTANT (EXPT 2 INTEGERBITS)))
                                   ROUNDER))
                     (T (IPLUS HIPART ROUNDER])
```

```
(\CONVERTTOFRACTION
  [LAMBDA (FLOAT)                                                    (* rmk%: " 3-JUL-82 23:29")
    ;; converts a floating point number into a fixed point number with INTEGERBITS worth of integer part.  Always returns a large integer so that the
    ;; box can be clobbered.
    (PROG (RESULT BOX)
          (RETURN (COND
                     ([SMALLP (SETQ RESULT (FIX (FTIMES FLOAT (CONSTANT (FLOAT (EXPT 2 (IDIFFERENCE
                                                                                        BITSPERINTEGER
                                                                                        INTEGERBITS]
                                                                                    ; clobber a created box.
                      (PutUnboxed (SETQ BOX (CREATECELL \FIXP))
                             RESULT)
                      BOX)
                     (T RESULT])

)

(DECLARE%: EVAL@COMPILE

(RPAQQ INTEGERBITS 12)

(CONSTANTS (INTEGERBITS 12))
)

;; cursor functions not on LLDISPLAY

(DEFINEQ

(CURSORP
  [LAMBDA (X)                                                        (* kbr%: " 5-Jul-85 17:54")
                                                                     ; is X a cursor?
    (type? CURSOR X])

(CURSORBITMAP
  [LAMBDA NIL CursorBitMap])

(CreateCursorBitMap
  [LAMBDA (ARRAY)                                                    (* rmk%: " 1-APR-82 22:20")
                                                                     ; makes a bitmap out of an array of values.
    (PROG ((BM (BITMAPCREATE 16 16))
           BASE)
          (SETQ BASE (ffetch BITMAPBASE of BM))
          (for I from 0 to 15 do (\PUTBASE BASE I (LOGAND (ELT ARRAY (ADD1 I))
                                                          WORDMASK)))
          (RETURN BM])

)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(PUTPROPS CURSORBITMAP MACRO (NIL CursorBitMap))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ HARDCURSORHEIGHT 16)

(RPAQQ HARDCURSORWIDTH 16)

(CONSTANTS (HARDCURSORHEIGHT 16)
       (HARDCURSORWIDTH 16))
)

(DECLARE%: EVAL@COMPILE

(ADDTOVAR GLOBALVARS CursorBitMap)
)

;; END EXPORTED DEFINITIONS

(RPAQQ CARETCOMS
       ((BITMAPS \DefaultCaret)
        (INITVARS (\CARET.UP NIL
                         ;; global.  NIL if no caret showing, otherwise a CARET1 record with CURSOR, stream, x, y, and RATE (= off rate)
                         )
                  (\CARET.DEFAULT NIL                                ; global = default caret to put up.  An instance of CARET1
                                                                     ; datatype
                         )
```

```
               (\CARET.TIMER (SETUPTIMER 0)
                                                                  ; time for next caret action
                      )
               (DEFAULTCARET (CURSORCREATE \DefaultCaret NIL 3 4))
               (DEFAULTCARETRATE 333                              ; default rate for flashing caret
                      )
               (\CARET.ON.RATE DEFAULTCARETRATE)
               (\CARET.OFF.RATE DEFAULTCARETRATE)
               (\CARET.FORCED.OFF.RATE 0))
        (ADDVARS (\SYSTEMTIMERVARS \CARET.TIMER))
        (DECLARE%: DONTCOPY (RECORDS CARET1))
        (INITRECORDS CARET1)
        (FNS CARET \CARET.CREATE \CARET.DOWN \CARET.FLASH? \CARET.SHOW CARETRATE \CARET.FLASH.AGAIN
             \CARET.FLASH.MULTIPLE \CARET.FLASH)
        (FNS \MEDW.CARET.SHOW)
                                                                  ; some declarations are on LLDISPLAY -- macro for
                                                                  ; \CHECKCARET and globalvar declaration for \CARET.UP
        (GLOBALVARS \CARET.DEFAULT \CARET.ON.RATE \CARET.OFF.RATE DEFAULTCARET \CARET.TIMER \CARET.UP
              \CARET.FORCED.OFF.RATE)
        (DECLARE%: DONTEVAL@LOAD DOCOPY (ADDVARS (TTYBACKGROUNDFNS \CARET.FLASH?)))
        (FNS \AREAVISIBLE? \REGIONOVERLAPAREAP \AREAINREGIONP)
        (P (CARET T))))

(RPAQQ \DefaultCaret )

(RPAQ? \CARET.UP NIL ;; global.  NIL if no caret showing, otherwise a CARET1 record with CURSOR, stream, x, y, and RATE (= off rate)
                     )

(RPAQ? \CARET.DEFAULT NIL                                         ; global = default caret to put up.  An instance of CARET1
                                                                  ; datatype
                          )

(RPAQ? \CARET.TIMER (SETUPTIMER 0)
                                                                  ; time for next caret action
                    )

(RPAQ? DEFAULTCARET (CURSORCREATE \DefaultCaret NIL 3 4))

(RPAQ? DEFAULTCARETRATE 333                                       ; default rate for flashing caret
                                )

(RPAQ? \CARET.ON.RATE DEFAULTCARETRATE)

(RPAQ? \CARET.OFF.RATE DEFAULTCARETRATE)

(RPAQ? \CARET.FORCED.OFF.RATE 0)

(ADDTOVAR \SYSTEMTIMERVARS \CARET.TIMER)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RECORD CARET1                                                    ; a record that describes a SHOWING caret
       (STREAM                                                    ; the stream the caret is showing in
              STREAMX                                             ; the X position stream relative that it was shown at
              STREAMY                                             ; the Y position stream relative that it was shown at
              CURSOR                                              ; the cursor bitmap + x and y that this caret represents
              RATE                                                ; the 'down rate' for this caret, in ticks.  After comes down (when
                                                                  ; \CARET.TIMER expires), \CARET.TIMER will be rescheduled
                                                                  ; to put something up.  This is the rate to use
                                                                  ; NEXT for threading carets together
               . NEXT))
)
)

(DEFINEQ

(CARET
  [LAMBDA (NEWCARET)                                              (* kbr%: " 6-Jul-85 16:13")
                                                                  ; changes the 'system default' caret

    (PROG1 (COND
              (\CARET.DEFAULT                                     ; merely stored as a 'cursor' record for simplicity
                     (fetch (CARET1 CURSOR) of \CARET.DEFAULT))
              (T 'OFF))
        [COND
           (NEWCARET (\CHECKCARET)
                  (CARETRATE (CARETRATE))                         ; make sure the caret rate is set
                  (SETQ \CARET.DEFAULT (SELECTQ NEWCARET
                                           (T (COND
                                                 ((EQ DEFAULTCARET 'OFF)
                                                  NIL)
                                                 ((CURSORP DEFAULTCARET)
                                                  (create CARET1
                                                          CURSOR _ DEFAULTCARET))
                                                 (T (ERROR "DEFAULTCARET is not a cursor" DEFAULTCARET))))
                                           (OFF NIL)
```

```
                                        (COND
                                           ((CURSORP NEWCARET)
                                            (create CARET1
                                                    CURSOR _ NEWCARET))
                                           (T (LISPERROR "ILLEGAL ARG" NEWCARET])])
```

(\CARET.CREATE

```
[LAMBDA (CURSOR)                                        (* jds "11-Jul-85 19:38")
   (create CARET1
           CURSOR _ (OR CURSOR DEFAULTCARET])
```

(\CARET.DOWN

```
[LAMBDA (STREAM INTERVAL UNLESSOCCLUDED)                (* lmm " 4-May-84 18:15")
```

;; take caret down if it is up. If you take it down, reschedule to put it back up in INTERVAL (or 0) --- often called thru \CHECKCARET macro

```
(COND
   (\CARET.UP (COND
                 ([OR (NULL STREAM)
                      (fetch (CARET1 NEXT) of \CARET.UP)
                      (EQ (fetch (CARET1 STREAM) of \CARET.UP)
                          (COND
                             ((type? WINDOW STREAM)
                              (fetch (WINDOW DSP) of STREAM))
                             (T STREAM]
                  [while (UNINTERRUPTABLY
                             [COND
                                ((\CARET.SHOW \CARET.UP UNLESSOCCLUDED)
                                                        ; take caret down and set global state
                                 (replace (CARET1 STREAM) of \CARET.UP with NIL)
                                 (SETQ \CARET.UP (fetch (CARET1 NEXT) of \CARET.UP])]
                  (SETUPTIMER (OR INTERVAL \CARET.FORCED.OFF.RATE)
                         \CARET.TIMER])
```

(\CARET.FLASH?

```
[LAMBDA (STREAM CARET ONRATE OFFRATE X Y)               (* AJB "17-Jul-85 12:47")
```

;;; Flashes the CARET at the ONRATE/OFFRATE at the X,Y position in the current TTY window. If CARET is NIL, uses \CARET.DEFAULT as the
;;; caret. Takes either a display stream or a textstream as the destination stream to flash the caret. The caret is not flashed on a shift-selection in a
;;; window

```
   (COND
      (\CARET.UP [COND
                    ((TIMEREXPIRED? \CARET.TIMER)
                     (\CARET.DOWN NIL (fetch (CARET1 RATE) of \CARET.UP)
                            (OR (KEYDOWNP 'LSHIFT)
                                (KEYDOWNP 'RSHIFT)
                                (KEYDOWNP 'COPY]
             NIL)
      ((AND (OR CARET (SETQ CARET \CARET.DEFAULT))
            (TIMEREXPIRED? \CARET.TIMER)
            [OR [DISPLAYSTREAMP (OR STREAM (SETQ STREAM (TTYDISPLAYSTREAM]
                (AND (IMAGESTREAMTYPEP STREAM 'TEXT)
                     (SETQ STREAM (WINDOWPROP (CAR (fetch (TEXTOBJ \WINDOW) of (TEXTOBJ STREAM)))
                                              'DSP]
            (\CARET.FLASH CARET STREAM OFFRATE (OR (KEYDOWNP 'LSHIFT)
                                                   (KEYDOWNP 'RSHIFT)
                                                   (KEYDOWNP 'COPY))
                  X Y))
       ;; \CARET.DEFAULT is NIL if by default the caret is OFF --- the KEYDOWNP clause is a hack to detect whether we are doing a copy-select
       (replace (CARET1 NEXT) of CARET with NIL)        ; Since this function is displaying a new caret, destroy any
                                                        ; chaining of multiple carets
       (SETUPTIMER (OR ONRATE \CARET.ON.RATE)
              \CARET.TIMER)
       T])
```

(\CARET.SHOW

```
[LAMBDA (CARET UNLESSOCCLUDED)                          ; Edited 25-Feb-94 16:53 by sybalsky
```

;; GENERIC caret flasher.

```
(LET (DS)
     (SETQ DS (fetch (CARET1 STREAM) of CARET))
     (WINDOWOP 'SCCARETFLASH (FETCH (WINDOW SCREEN) OF (FETCH (\DISPLAYDATA XWINDOWHINT)
                                                               OF (FETCH (STREAM IMAGEDATA) OF DS)))
            CARET UNLESSOCCLUDED])
```

(CARETRATE

```
[LAMBDA (ONRATE OFFRATE)                                (* lmm " 3-May-84 11:35")
```

;; sets the default caret rate (s) to be ONRATE/OFFRATE in milliseconds

```
(PROG1 (COND
          ((EQ \CARET.ON.RATE \CARET.OFF.RATE)
```

```
              \CARET.ON.RATE)
             (T (CONS \CARET.ON.RATE \CARET.OFF.RATE)))
        [COND
           ((OR ONRATE OFFRATE)
            (SETUPTIMER 0 \CARET.TIMER)
            (SETQ \CARET.ON.RATE (OR (FIXP ONRATE)
                                     (FIX DEFAULTCARETRATE)))
            (SETQ \CARET.OFF.RATE (OR (FIXP OFFRATE)
                                      \CARET.ON.RATE])])
```

(\CARET.FLASH.AGAIN

```
  [LAMBDA (CARET STREAM X Y)                                    (* AJB "14-Aug-85 17:04")
    (LET ((OCARET \CARET.UP))
         (COND
            ([AND OCARET CARET (DISPLAYSTREAMP (OR STREAM (SETQ STREAM (TTYDISPLAYSTREAM]
             (for (OC _ OCARET) by (fetch (CARET1 NEXT) of OC)
                do (COND
                       [(NULL OC)
                        (RETURN (COND
                                    ((\CARET.FLASH CARET STREAM (fetch (CARET1 RATE) of \CARET.UP)
                                            (OR (KEYDOWNP 'LSHIFT)
                                                (KEYDOWNP 'RSHIFT)
                                                (KEYDOWNP 'COPY))
                                            X Y)                ; OK, showed this one
                                     (OR (EQ \CARET.UP CARET)
                                         (SHOULDNT))
                                     (replace (CARET1 NEXT) of CARET with OCARET]
                       ((EQ OC CARET)                           ; this CARET is already showing
                        (RETURN])
```

(\CARET.FLASH.MULTIPLE

```
  [LAMBDA (STREAMS CARETS ONRATE OFFRATE)                       (* AJB "14-Aug-85 17:10")
                                                                ; this is probably just a template for how to flash multiple carets
    (COND
       ((\CARET.FLASH? (CAR STREAMS)
              (CAR CARETS)
              ONRATE OFFRATE)
        (for STR in (CDR STREAMS) as CARET in (CDR CARETS) do (\CARET.FLASH.AGAIN CARET STR])
```

(\CARET.FLASH

```
  [LAMBDA (CARET STREAM RATE UNLESSOCCLUDED X Y)                (* kbr%: " 5-Jul-85 17:51")
    (PROG (CURSOR ANSWER)
          (SETQ CURSOR (fetch (CARET1 CURSOR) of CARET))
          (replace (CARET1 STREAM) of CARET with STREAM)
          (replace (CARET1 STREAMX) of CARET with (IDIFFERENCE (OR X (DSPXPOSITION NIL STREAM))
                                                               (fetch (CURSOR CUHOTSPOTX) of CURSOR)))
          (replace (CARET1 STREAMY) of CARET with (IDIFFERENCE (OR Y (DSPYPOSITION NIL STREAM))
                                                               (fetch (CURSOR CUHOTSPOTY) of CURSOR)))
          (replace (CARET1 RATE) of CARET with (OR RATE \CARET.OFF.RATE))
          (UNINTERRUPTABLY
              (COND
                 ((\CARET.SHOW CARET UNLESSOCCLUDED)
                  (SETQ \CARET.UP CARET)
                  (SETQ ANSWER T))))
          (RETURN ANSWER])

)

(DEFINEQ
```

(\MEDW.CARET.SHOW

```
  [LAMBDA (SCREEN CARET UNLESSOCCLUDED)                         ; Edited 17-Jan-94 10:28 by sybalsky:mv:envos
    ;; MEDLEY-window-system specific version of \CARET.SHOW (vectored thru the screen).  Flash the caret (by inverting its image).
    ;; UNLESSOCCLUDED controls whether you bring the window to the top if the caret is under some other window.
    (PROG (DS)
          (SETQ DS (fetch (CARET1 STREAM) of CARET))
          (RETURN (PROG (DD CARETWIN CBMX CBMY CURSOR CARETBM CWX CWY CARETBMWIDTH CARETBMHEIGHT CLIPREG CLIPVAR
                            )
                        (SETQ DD (fetch (STREAM IMAGEDATA) of DS))
                        (SETQ CARETWIN (WFROMDS DS))
                        (SETQ CBMX 0)
                        (SETQ CBMY 0)
                        (SETQ CURSOR (fetch (CARET1 CURSOR) of CARET))
                        (\CURSORBITSPERPIXEL CURSOR (BITSPERPIXEL (DSPDESTINATION NIL CARETWIN)))
                        (SETQ CARETBM (fetch (CURSOR CUIMAGE) of CURSOR))
                        (SETQ CWX (fetch (CARET1 STREAMX) of CARET))
                        (SETQ CWY (fetch (CARET1 STREAMY) of CARET))
                        (SETQ CARETBMWIDTH (fetch (BITMAP BITMAPWIDTH) of CARETBM))
                        (SETQ CARETBMHEIGHT (fetch (BITMAP BITMAPHEIGHT) of CARETBM))
                                                                ; calculate how much to reduce the caret region by do to the
                                                                ; clipping region of the window.
                        (SETQ CLIPREG (fetch (\DISPLAYDATA DDClippingRegion) of DD))
                        (COND
```

```
                        ((IGREATERP (SETQ CLIPVAR (fetch (REGION LEFT) of CLIPREG))
                                    CWX)
                         [SETQ CARETBMWIDTH (IDIFFERENCE CARETBMWIDTH (SETQ CBMX (IDIFFERENCE CLIPVAR CWX]
                         (SETQ CWX CLIPVAR)))
                     (COND
                        ((IGREATERP CARETBMWIDTH (SETQ CLIPVAR (IDIFFERENCE (IPLUS CLIPVAR
                                                                                   (fetch (REGION WIDTH)
                                                                                      of CLIPREG))
                                                                            CWX)))
                         (SETQ CARETBMWIDTH CLIPVAR)))
                     (COND
                        ((IGREATERP (SETQ CLIPVAR (fetch (REGION BOTTOM) of CLIPREG))
                                    CWY)
                         [SETQ CARETBMHEIGHT (IDIFFERENCE CARETBMHEIGHT (SETQ CBMY (IDIFFERENCE CLIPVAR CWY]
                         (SETQ CWY CLIPVAR)))
                     (COND
                        ((IGREATERP CARETBMHEIGHT (SETQ CLIPVAR (IDIFFERENCE (IPLUS CLIPVAR
                                                                                   (fetch (REGION HEIGHT)
                                                                                      of CLIPREG))
                                                                             CWY)))
                         (SETQ CARETBMHEIGHT CLIPVAR)))
```

(* note the time of the next change. This must be done without creating boxes because happens during keyboard wait.)

```
                     (COND
                        ((OR (ILESSP CARETBMWIDTH 1)
                             (ILESSP CARETBMHEIGHT 1))          (* caret isn't within clipping region.)
                         (RETURN T)))                           (* convert the base of the caret location to screen coordinates.)
                     (SETQ CWX (\DSPTRANSFORMX CWX DD))
                     (SETQ CWY (\DSPTRANSFORMY CWY DD))
```

(* having only this section uninterruptable leaves open the possibility that the window moves or the timer is wrong but these will only mess up the display and are low frequency events.)

```
                     (COND
                        [(AND (OPENWP CARETWIN)
                              (\AREAVISIBLE? CARETWIN CWX CWY (IPLUS CWX (SUB1 CARETBMWIDTH))
                                    (IPLUS CWY (SUB1 CARETBMHEIGHT]
                        (UNLESSOCCLUDED (RETURN))
                        (T (TOTOPW CARETWIN)))
                     (BITBLT CARETBM CBMX CBMY (DSPDESTINATION NIL CARETWIN)
                           CWX CWY CARETBMWIDTH CARETBMHEIGHT 'INPUT 'INVERT)
                     (RETURN T])

)

;; some declarations are on LLDISPLAY -- macro for \CHECKCARET and globalvar declaration for \CARET.UP

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \CARET.DEFAULT \CARET.ON.RATE \CARET.OFF.RATE DEFAULTCARET \CARET.TIMER \CARET.UP
       \CARET.FORCED.OFF.RATE)
)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(ADDTOVAR TTYBACKGROUNDFNS \CARET.FLASH?)
)

(DEFINEQ
```

(\AREAVISIBLE?

```
  [LAMBDA (WIN LFT BTM RGHT TOP)                                (* kbr%: "18-Feb-86 18:05")
    ;; is the area whose screen limits are LFT BTM RGHT and TOP eniretly visible within WIN,
    (PROG (WPTR)
          (SETQ WPTR (fetch (SCREEN SCTOPW) of (fetch (WINDOW SCREEN) of WIN)))
          (COND
             ((NOT (\AREAINREGIONP (fetch (WINDOW REG) of WIN)
                          LFT BTM RGHT TOP))                    ; if the caret region isn't completely within the window, forget it.
              (RETURN)))
      LP  (COND
             ((EQ WPTR WIN)
              (RETURN T))
             ((\REGIONOVERLAPAREAP (fetch (WINDOW REG) of WPTR)
                     LFT BTM RGHT TOP)
              (RETURN NIL))
             ((SETQ WPTR (fetch (WINDOW NEXTW) of WPTR))
              (GO LP])
```

(\REGIONOVERLAPAREAP

```
  [LAMBDA (REG LFT BTM RGHT TOP)                                (* rrb "17-Feb-86 18:50")
    ;; is there any overlap between the region REG and the area defined by left bottom right and top?
    (NOT (OR (IGREATERP (fetch (REGION LEFT) of REG)
                    RGHT)
```

```
            (IGREATERP LFT (fetch (REGION RIGHT) of REG))
            (IGREATERP (fetch (REGION BOTTOM) of REG)
                   TOP)
            (IGREATERP BTM (fetch (REGION TOP) of REG])
```

(\AREAINREGIONP

```
  [LAMBDA (REGION LFT BTM RGHT TOP)                    (* rrb "14-OCT-83 15:32")
    (AND (IGEQ LFT (fetch LEFT of REGION))
         (IGEQ BTM (fetch BOTTOM of REGION))
         (IGEQ (fetch PRIGHT of REGION)
               RGHT)
         (IGEQ (fetch PTOP of REGION)
               TOP])
```

)

(**CARET** T)

;; Region functions

(DEFINEQ

(CREATEREGION

```
  [LAMBDA (LEFT BOTTOM WIDTH HEIGHT)                   (* rrb "17-JUN-83 08:56")
                                                       ; creates a region structure.
    (create REGION
            LEFT _ LEFT
            BOTTOM _ BOTTOM
            WIDTH _ WIDTH
            HEIGHT _ HEIGHT])
```

(REGIONP

```
  [LAMBDA (X)                                          (* rrb "29-Jun-84 18:00")
    (AND (type? REGION X)
         X])
```

(INTERSECTREGIONS

```
  [LAMBDA REGIONS                                      (* kbr%: "24-Jan-86 18:30")
    ;; returns the largest region that is contained in all of REGIONS
    (COND
       ((EQ REGIONS 0)
        ;; this is documented as returning a very large region.  This one covers the entire FIXP range so should work for many purposes.  rrb
        (create REGION
                LEFT _ (SUB1 MIN.FIXP)
                BOTTOM _ (SUB1 MIN.FIXP)
                WIDTH _ (PLUS (TIMES 2 MAX.FIXP)
                              4)
                HEIGHT _ (PLUS (TIMES 2 MAX.FIXP)
                               4)))
       (T (PROG (REG LFT RGHT BTTM TP)
                (SETQ REG (ARG REGIONS 1))
                (SETQ LFT (fetch (REGION LEFT) of REG))
                [SETQ RGHT (SUB1 (IPLUS LFT (fetch (REGION WIDTH) of REG]
                (SETQ BTTM (fetch (REGION BOTTOM) of REG))
                [SETQ TP (SUB1 (IPLUS BTTM (fetch (REGION HEIGHT) of REG]
                [for I from 2 thru REGIONS do (SETQ REG (ARG REGIONS I))
                                              [COND
                                                 ((IGREATERP (fetch (REGION LEFT) of REG)
                                                             LFT)
                                                  (SETQ LFT (fetch (REGION LEFT) of REG]
                                              [COND
                                                 ((IGREATERP (fetch (REGION BOTTOM) of REG)
                                                             BTTM)
                                                  (SETQ BTTM (fetch (REGION BOTTOM) of REG]
                                              [COND
                                                 ((ILESSP (fetch (REGION RIGHT) of REG)
                                                          RGHT)
                                                  (SETQ RGHT (fetch (REGION RIGHT) of REG]
                                              (COND
                                                 ((ILESSP (fetch (REGION TOP) of REG)
                                                          TP)
                                                  (SETQ TP (fetch (REGION TOP) of REG]
                (RETURN (COND
                           ((AND (IGEQ RGHT LFT)
                                 (IGEQ TP BTTM))
                            (create REGION
                                    LEFT _ LFT
                                    BOTTOM _ BTTM
                                    WIDTH _ (ADD1 (IDIFFERENCE RGHT LFT))
                                    HEIGHT _ (ADD1 (IDIFFERENCE TP BTTM])
```

(UNIONREGIONS

```
[LAMBDA REGIONS                                                    (* rrb "30-Dec-85 17:07")
```

;; returns the smallest region that encloses all of REGIONS

```
(COND
   ((EQ 0 REGIONS)
    NIL)
   (T (PROG (REG LFT RGHT BTTM TP)
            (SETQ REG (ARG REGIONS 1))
            (SETQ LFT (fetch (REGION LEFT) of REG))
            (SETQ RGHT (fetch (REGION PRIGHT) of REG))
            (SETQ BTTM (fetch (REGION BOTTOM) of REG))
            (SETQ TP (fetch (REGION PTOP) of REG))
            [for I from 2 thru REGIONS do (SETQ REG (ARG REGIONS I))
                                          [COND
                                             ((LESSP (fetch (REGION LEFT) of REG)
                                                     LFT)
                                              (SETQ LFT (fetch (REGION LEFT) of REG]
                                          [COND
                                             ((LESSP (fetch (REGION BOTTOM) of REG)
                                                     BTTM)
                                              (SETQ BTTM (fetch (REGION BOTTOM) of REG]
                                          [COND
                                             ((GREATERP (fetch (REGION PRIGHT) of REG)
                                                        RGHT)
                                              (SETQ RGHT (fetch (REGION PRIGHT) of REG]
                                          (COND
                                             ((GREATERP (fetch (REGION PTOP) of REG)
                                                        TP)
                                              (SETQ TP (fetch (REGION PTOP) of REG]
            (RETURN (create REGION
                            LEFT _ LFT
                            BOTTOM _ BTTM
                            WIDTH _ (DIFFERENCE RGHT LFT)
                            HEIGHT _ (DIFFERENCE TP BTTM])
```

(REGIONSINTERSECTP

```
[LAMBDA (REGION1 REGION2)                                          (* rrb "16-AUG-81 08:29")
```

;; determines if two regions intersect

```
(NOT (OR (IGREATERP (fetch LEFT of REGION1)
                    (fetch RIGHT of REGION2))
         (IGREATERP (fetch LEFT of REGION2)
                    (fetch RIGHT of REGION1))
         (IGREATERP (fetch BOTTOM of REGION1)
                    (fetch TOP of REGION2))
         (IGREATERP (fetch BOTTOM of REGION2)
                    (fetch TOP of REGION1])
```

(SUBREGIONP

```
[LAMBDA (LARGEREGION SMALLREGION)                                  (* rrb "25-JUN-82 15:09")
```

;; determines if small region is a subset of large region. (SUBREGIONP '(9 0 100 100) '(0 10 100 80))

```
(AND (IGEQ (fetch LEFT of SMALLREGION)
           (fetch LEFT of LARGEREGION))
     (IGEQ (fetch BOTTOM of SMALLREGION)
           (fetch BOTTOM of LARGEREGION))
     (IGEQ (fetch PRIGHT of LARGEREGION)
           (fetch PRIGHT of SMALLREGION))
     (IGEQ (fetch PTOP of LARGEREGION)
           (fetch PTOP of SMALLREGION])
```

(EXTENDREGION

```
[LAMBDA (REGION INCLUDEREGION)                                     (* rrb " 5-FEB-82 09:25")
```

;; destructively extends REGION to include INCLUDEREGION

```
[COND
   ((IGREATERP (fetch (REGION LEFT) of REGION)
               (fetch (REGION LEFT) of INCLUDEREGION))
    (replace (REGION WIDTH) of REGION with (IDIFFERENCE (fetch (REGION PRIGHT) of REGION)
                                                        (fetch (REGION LEFT) of INCLUDEREGION)))
    (replace (REGION LEFT) of REGION with (fetch (REGION LEFT) of INCLUDEREGION]
[COND
   ((IGREATERP (fetch (REGION BOTTOM) of REGION)
               (fetch (REGION BOTTOM) of INCLUDEREGION))
    (replace (REGION HEIGHT) of REGION with (IDIFFERENCE (fetch (REGION PTOP) of REGION)
                                                         (fetch (REGION BOTTOM) of INCLUDEREGION)))
    (replace (REGION BOTTOM) of REGION with (fetch (REGION BOTTOM) of INCLUDEREGION]
[COND
   ((IGREATERP (fetch (REGION RIGHT) of INCLUDEREGION)
               (fetch (REGION RIGHT) of REGION))
    (replace (REGION WIDTH) of REGION with (ADD1 (IDIFFERENCE (fetch (REGION RIGHT) of INCLUDEREGION)
                                                              (fetch (REGION LEFT) of REGION]
[COND
```

```
       ((IGREATERP (fetch (REGION TOP) of INCLUDEREGION)
                   (fetch (REGION TOP) of REGION))
        (replace (REGION HEIGHT) of REGION with (ADD1 (IDIFFERENCE (fetch (REGION TOP) of INCLUDEREGION)
                                                                   (fetch (REGION BOTTOM) of REGION]
    REGION])
```

(EXTENDREGIONBOTTOM

```
  [LAMBDA (REG NEWBOTTOM)                                          (* rrb "29-DEC-81 10:02")
                                                                   ; extends a region to the bottom
    (PROG ((OLDBOTTOM (fetch (REGION BOTTOM) of REG)))
          [COND
             ((IGREATERP OLDBOTTOM NEWBOTTOM)
              (replace (REGION BOTTOM) of REG with NEWBOTTOM)
              (replace (REGION HEIGHT) of REG with (IPLUS (fetch (REGION HEIGHT) of REG)
                                                          (IDIFFERENCE OLDBOTTOM NEWBOTTOM]
          (RETURN REG])
```

(EXTENDREGIONLEFT

```
  [LAMBDA (REG NEWLEFT)                                            (* rrb "29-DEC-81 09:37")
                                                                   ; extends a region to the left
    (PROG ((OLDLEFT (fetch (REGION LEFT) of REG)))
          [COND
             ((IGREATERP OLDLEFT NEWLEFT)
              (replace (REGION LEFT) of REG with NEWLEFT)
              (replace (REGION WIDTH) of REG with (IPLUS (fetch (REGION WIDTH) of REG)
                                                         (IDIFFERENCE OLDLEFT NEWLEFT]
          (RETURN REG])
```

(EXTENDREGIONRIGHT

```
  [LAMBDA (REG NEWRIGHT)                                           (* rrb "29-DEC-81 10:06")
                                                                   ; extends a region to the left
    (PROG ((OLDRIGHT (fetch (REGION RIGHT) of REG)))
          [COND
             ((ILESSP OLDRIGHT NEWRIGHT)
              (replace (REGION WIDTH) of REG with (IPLUS (fetch (REGION WIDTH) of REG)
                                                         (IDIFFERENCE NEWRIGHT OLDRIGHT]
          (RETURN REG])
```

(EXTENDREGIONTOP

```
  [LAMBDA (REG NEWTOP)                                             (* rrb "29-DEC-81 10:07")
                                                                   ; extends a region to the top
    (PROG ((OLDTOP (fetch (REGION TOP) of REG)))
          [COND
             ((ILESSP OLDTOP NEWTOP)
              (replace (REGION HEIGHT) of REG with (IPLUS (fetch (REGION HEIGHT) of REG)
                                                          (IDIFFERENCE NEWTOP OLDTOP]
          (RETURN REG])
```

(INSIDEP

```
  [LAMBDA (REGION POSORX Y)                                        (* rrb "18-May-84 21:04")
```

;; returns T if the position X Y is inside the region REGION. If POSORX is a position, returns T if that position is inside of REGION

```
    (COND
       ((WINDOWP REGION)
        (INSIDEP (DSPCLIPPINGREGION NIL REGION)
                 POSORX Y))
       (T (COND
             ((AND (NUMBERP POSORX)
                   (NUMBERP Y))
              (INSIDE? REGION POSORX Y))
             ((POSITIONP POSORX)
              (INSIDE? REGION (fetch (POSITION XCOORD) of POSORX)
                              (fetch (POSITION YCOORD) of POSORX)))
             ((NUMBERP POSORX)
              (\ILLEGAL.ARG Y))
             (T (\ILLEGAL.ARG POSORX])
```

(STRINGREGION

```
  [LAMBDA (STR STREAM PRIN2FLG RDTBL)                              (* rmk%: "25-AUG-83 18:06")
```

;; returns the region taken up by STR if it were printed at the current position of STREAM

```
    (create REGION
            LEFT _ (DSPXPOSITION NIL STREAM)
            BOTTOM _ (IDIFFERENCE (DSPYPOSITION NIL STREAM)
                                  (FONTPROP STREAM 'DESCENT))
            WIDTH _ (STRINGWIDTH STR STREAM PRIN2FLG RDTBL)
            HEIGHT _ (FONTPROP STREAM 'HEIGHT])

)
```

;; line and spline drawing.

```
;; Brushes and brush initialization

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

[PUTDEF '\BRUSHBBT 'RESOURCES '(NEW (create PILOTBBT]
)
)

(/SETTOPVAL '\\BRUSHBBT.GLOBALRESOURCE NIL)

(DEFINEQ
```

(\BRUSHBITMAP

```
  [LAMBDA (BRUSHSHAPE BRUSHWIDTH)                                    (* rrb " 9-Sep-86 16:30")

;;; returns the bitmap for the brush of the shape and size.  See comments on \InitCurveBrushes.

    (DECLARE (GLOBALVARS \BrushAList))
    (LET [(BRUSHES&METHOD (CDR (OR (FASSOC BRUSHSHAPE \BrushAList)
                                   (\ILLEGAL.ARG BRUSHSHAPE]
         (COND
            ((NOT (GREATERP BRUSHWIDTH 0))
             ;; if brush is 0 or negative, return an empty brush.  Might want to error but this would require users to handle it.
             (BITMAPCREATE 0 0))
            [(ILESSP BRUSHWIDTH 17)                                  ; lowest 16 brushes are stored.  FIX them so ELT works.
             (ELT (fetch (BRUSHITEM BRUSHARRAY) of BRUSHES&METHOD)
                  (COND
                     ((FIXP BRUSHWIDTH))
                     ((GREATERP BRUSHWIDTH 1)
                      (FIXR BRUSHWIDTH))
                     (T 1]
            [(CDR (FASSOC BRUSHWIDTH (fetch (BRUSHITEM BRUSHCACHE) of BRUSHES&METHOD]
            (T ;; cache the brush bitmap.  This is done so that the brush creation methods don't have to be efficient.
               (LET ((NEWBRUSHBM (APPLY* (fetch (BRUSHITEM CREATEMETHOD) of BRUSHES&METHOD)
                                         BRUSHWIDTH)))
                    (replace (BRUSHITEM BRUSHCACHE) of BRUSHES&METHOD with (CONS (CONS BRUSHWIDTH NEWBRUSHBM)
                                                                                 (fetch (BRUSHITEM BRUSHCACHE)
                                                                                    of BRUSHES&METHOD)))

                    NEWBRUSHBM])
```

(\GETBRUSH

```
  [LAMBDA (BRUSH)                                                    (* rrb " 9-Sep-86 16:30")
    (COND
       ((type? BITMAP BRUSH)
        BRUSH)
       [(LISTP BRUSH)
        (\BRUSHBITMAP (CAR BRUSH)
               (CAR (LISTP (CDR BRUSH]
       (T (\BRUSHBITMAP 'ROUND (OR BRUSH 1])
```

(\GETBRUSHBBT

```
  [LAMBDA (BRUSHBM DISPLAYDATA BBT)                                  (* kbr%: "18-Aug-85 12:46")
    ;; Initializes BBT for the BRUSHBM and DS and returns BBT, unless the BRUSHBM is a 1-point brush, in which case it returns NIL.
    (COND
       ((AND (EQ (fetch (BITMAP BITMAPHEIGHT) of BRUSHBM)
                 1)
              (EQ (ffetch (BITMAP BITMAPWIDTH) of BRUSHBM)
                 1)
              (EQ (BITMAPBIT BRUSHBM 0 0)
                 1))                                                 ; special case of single point brush shape.
        NIL)
       (T                                                            ; update as many fields in the brush bitblt table as possible from
                                                                     ; DS.
          (replace (PILOTBBT PBTDESTBPL) of BBT with (UNFOLD (fetch (BITMAP BITMAPRASTERWIDTH)
                                                                    of (fetch (\DISPLAYDATA DDDestination) of
                                                                                                          DISPLAYDATA
                                                                                                          ))
                                                             BITSPERWORD))
          (freplace (PILOTBBT PBTSOURCEBPL) of BBT with (UNFOLD (ffetch (BITMAP BITMAPRASTERWIDTH) of BRUSHBM)
                                                                BITSPERWORD))
          (freplace (PILOTBBT PBTFLAGS) of BBT with 0)
          (freplace (PILOTBBT PBTDISJOINT) of BBT with T)
          (\SETPBTFUNCTION BBT (ffetch (\DISPLAYDATA DDSOURCETYPE) of DISPLAYDATA)
                 (SELECTQ (ffetch (\DISPLAYDATA DDOPERATION) of DISPLAYDATA)
                      ((PAINT REPLACE)
                          'PAINT)
                      ((INVERT ERASE)
                          'ERASE)
                      (SHOULDNT)))
          BBT])
```

(\InitCurveBrushes

```
  [LAMBDA NIL                                                    ; Edited 13-Oct-87 14:31 by jds
    ;; Set up the initial set of brush specs for curve drawing.  \BrushAList is an association list from brush-shape-names to a spec which is an instance
    ;; of the record BRUSHITEM.
    (DECLARE (GLOBALVARS \BrushNames \BrushAList \SingleBitBitmap))
    (PROG (BARRAY CREATIONMETHOD)
          (SETQ \SingleBitBitmap (BITMAPCREATE 1 1))
          (BITMAPBIT \SingleBitBitmap 0 0 1)
          (for BRUSHNAME in \BrushNames do (SETQ BARRAY (ARRAY 16 'POINTER NIL 1))
                                           (SETQ CREATIONMETHOD (PACK* '\MAKEBRUSH. BRUSHNAME))
                                           (SETA BARRAY 1 \SingleBitBitmap)
                                           (for SIZE from 2 to 16 do (SETA BARRAY SIZE (APPLY* CREATIONMETHOD SIZE)
                                                                                              ))
                                           (INSTALLBRUSH BRUSHNAME CREATIONMETHOD BARRAY])
```

(\BrushFromWidth

```
  [LAMBDA (W)                                                    (* hdj " 5-Nov-84 16:47")
    (LIST 'ROUND W])

)

(DEFINEQ
```

(\MAKEBRUSH.DIAGONAL

```
  [LAMBDA (SIZE)                                                 (* kbr%: "18-Aug-85 12:51")
    (PROG (BM)
          (SETQ BM (BITMAPCREATE SIZE SIZE))
          (for X from 0 to (SUB1 SIZE) do (BITMAPBIT BM X X 1))
          (RETURN BM])
```

(\MAKEBRUSH.HORIZONTAL

```
  [LAMBDA (SIZE)                                                 (* kbr%: "18-Aug-85 12:52")
```

;;; create a brush that has a horizontal line across it halfway down

```
    (PROG (BM)
          (SETQ BM (BITMAPCREATE SIZE SIZE))
          (BITBLT NIL NIL NIL BM 0 (SUB1 (FOLDHI SIZE 2))
                 NIL 1 'TEXTURE 'REPLACE BLACKSHADE)
          (RETURN BM])
```

(\MAKEBRUSH.VERTICAL

```
  [LAMBDA (SIZE)                                                 (* kbr%: "18-Aug-85 12:53")
    (PROG (BM)
          (SETQ BM (BITMAPCREATE SIZE SIZE))
          (BITBLT NIL NIL NIL BM (SUB1 (FOLDHI SIZE 2))
                 0 1 SIZE 'TEXTURE 'REPLACE BLACKSHADE)
          (RETURN BM])
```

(\MAKEBRUSH.SQUARE

```
  [LAMBDA (SIZE)                                                 (* kbr%: "18-Aug-85 13:07")
    (PROG (BM)
          (SETQ BM (BITMAPCREATE SIZE SIZE))
          (BITBLT NIL NIL NIL BM NIL NIL NIL NIL 'TEXTURE 'REPLACE BLACKSHADE)
          (RETURN BM])
```

(\MAKEBRUSH.ROUND

```
  [LAMBDA (SIZE)                                                 (* rrb "15-Sep-86 14:32")
                                                                 ; special cased 8 so that it wouldn't have a width of 7. rrb
    (PROG (RADIUS BITMAP BASE)
          (SETQ RADIUS (SUB1 (HALF SIZE)))
          (SETQ BITMAP (BITMAPCREATE SIZE SIZE))
          (SETQ BASE (fetch (BITMAP BITMAPBASE) of BITMAP))
          (SELECTQ SIZE
              (1 (\PUTBASE BASE 0 (MASK.1'S 15 1)))
              (2 (\PUTBASE BASE 0 (MASK.1'S 14 2))
                 (\PUTBASE BASE 1 (MASK.1'S 14 2)))
              (3 (\PUTBASE BASE 0 (MASK.1'S 14 1))
                 (\PUTBASE BASE 1 (MASK.1'S 13 3))
                 (\PUTBASE BASE 2 (MASK.1'S 14 1)))
              (4 (\PUTBASE BASE 0 (MASK.1'S 13 2))
                 (\PUTBASE BASE 1 (MASK.1'S 12 4))
                 (\PUTBASE BASE 2 (MASK.1'S 12 4))
                 (\PUTBASE BASE 3 (MASK.1'S 13 2)))
              (5 (\PUTBASE BASE 0 (MASK.1'S 13 1))
                 (\PUTBASE BASE 1 (MASK.1'S 12 3))
                 (\PUTBASE BASE 2 (MASK.1'S 11 5))
                 (\PUTBASE BASE 3 (MASK.1'S 12 3))
                 (\PUTBASE BASE 4 (MASK.1'S 13 1)))
```

```
              (8 (\PUTBASE BASE 0 (MASK.1'S 10 4))
                 (\PUTBASE BASE 1 (MASK.1'S 9 6))
                 (\PUTBASE BASE 2 (MASK.1'S 8 8))
                 (\PUTBASE BASE 3 (MASK.1'S 8 8))
                 (\PUTBASE BASE 4 (MASK.1'S 8 8))
                 (\PUTBASE BASE 5 (MASK.1'S 8 8))
                 (\PUTBASE BASE 6 (MASK.1'S 9 6))
                 (\PUTBASE BASE 7 (MASK.1'S 10 4)))
              (FILLCIRCLE RADIUS RADIUS RADIUS BLACKSHADE (DSPCREATE BITMAP)))
          (RETURN BITMAP])

)

(DEFINEQ

(INSTALLBRUSH
  [LAMBDA (BRUSHNAME BRUSHFN BRUSHARRAY)                    (* kbr%: "18-Jan-86 15:27")
    (DECLARE (GLOBALVARS \BrushAList))
    (PROG (OLDENTRY)
          (SETQ OLDENTRY (FASSOC BRUSHNAME \BrushAList))
          (COND
             (OLDENTRY (AND BRUSHARRAY (replace (BRUSHITEM BRUSHARRAY) of (CDR OLDENTRY) with BRUSHARRAY))
                    (AND BRUSHFN (replace (BRUSHITEM CREATEMETHOD) of (CDR OLDENTRY) with BRUSHFN)))
             (T [COND
                   ((AND BRUSHFN (NOT (ARRAYP BRUSHARRAY)))
                    (SETQ BRUSHARRAY (ARRAY 16 'POINTER NIL 1))
                    (for X from 1 to 16 do (SETA BRUSHARRAY X (APPLY* BRUSHFN X]
                (push \BrushAList (CONS BRUSHNAME (create BRUSHITEM
                                                          BRUSHARRAY _ BRUSHARRAY
                                                          CREATEMETHOD _ BRUSHFN)))
                (push KNOWN.BRUSHES BRUSHNAME])

)

(RPAQQ \BrushNames (ROUND SQUARE DIAGONAL HORIZONTAL VERTICAL))

(RPAQ? KNOWN.BRUSHES NIL)

(RPAQ? \BrushAList NIL)

(DECLARE%: EVAL@COMPILE

(RECORD BRUSHITEM (BRUSHARRAY CREATEMETHOD . BRUSHCACHE))
)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(\InitCurveBrushes)
)

(DECLARE%: DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \BrushAList KNOWN.BRUSHES)
)
)

;; Lines

(DEFINEQ

(\DRAWLINE.DISPLAY
  [LAMBDA (DISPLAYSTREAM X1 Y1 X2 Y2 WIDTH OPERATION COLOR DASHING)
                                                            ; Edited  1-Mar-2023 07:42 by lmm
                                                            ; Edited 29-Jan-91 14:59 by matsuda
    ;; DISPLAYSTREAM is guaranteed to be a display-stream.  Draws a line from x1,y1 to x2,y2 leaving the position at x2,y2
    ;; Added handling of brushes (I think, this is actually pretty tricky).
    (DECLARE (LOCALVARS . T)
           (GLOBALVARS \SCREENBITMAPS))
    [COND
       [(OR DASHING (BRUSHP WIDTH))
        (GLOBALRESOURCE
         \BRUSHBBT
         (LET ((BBT \BRUSHBBT)
               (BRUSH (INSURE.BRUSH WIDTH)))
              (if COLOR
                  then (replace (BRUSH BRUSHCOLOR) of BRUSH with COLOR))
              (IF [NOT (type? BIGBM (ffetch DDDestination of (fetch IMAGEDATA of DISPLAYSTREAM]
                  THEN (\LINEWITHBRUSH X1 Y1 X2 Y2 BRUSH (\GOOD.DASHLST DASHING BRUSH)
                              DISPLAYSTREAM BBT (SELECTQ OPERATION
                                                    (NIL (ffetch DDOPERATION of (fetch IMAGEDATA of DISPLAYSTREAM)
                                                                ))
                                                    ((REPLACE PAINT INVERT ERASE)
                                                        OPERATION)
```

```
                                                (\ILLEGAL.ARG OPERATION)))
            ELSE (PROG ((DD (fetch IMAGEDATA of DISPLAYSTREAM))
                        BITMAP BIGBMLIST HEIGHT BOTTOM BM YY1 YY2 ClippingTop ClippingBottom CTop CBottom)
                       (SETQ BITMAP (ffetch DDDestination of DD))
                       (SETQ BIGBMLIST (fetch (BIGBM BIGBMLIST) of BITMAP))
                       (SETQ HEIGHT (BITMAPHEIGHT BITMAP))
                       (SETQ ClippingTop (ffetch DDClippingTop of DD))
                       (SETQ ClippingBottom (ffetch DDClippingBottom of DD))
                       (SETQ BM (GetNewFragment BIGBMLIST))
                       (while (AND BM (IGREATERP HEIGHT ClippingBottom))
                          do (SETQ BOTTOM (IDIFFERENCE HEIGHT (BITMAPHEIGHT BM)))
                             [SETQ CTop (COND
                                           ((IGREATERP ClippingTop HEIGHT)
                                            (IDIFFERENCE HEIGHT BOTTOM))
                                           (T (IDIFFERENCE ClippingTop BOTTOM]
                             (if (IGEQ CTop 0)
                                 then [SETQ CBottom (COND
                                                       ((ILESSP ClippingBottom BOTTOM)
                                                        0)
                                                       (T (IDIFFERENCE ClippingBottom BOTTOM]
                                      (replace DDDestination of DD with BM)
                                      (replace DDClippingTop of DD with CTop)
                                      (replace DDClippingBottom of DD with CBottom)
                                      (\LINEWITHBRUSH X1 (IDIFFERENCE Y1 BOTTOM)
                                             X2
                                             (IDIFFERENCE Y2 BOTTOM)
                                             BRUSH
                                             (\GOOD.DASHLST DASHING BRUSH)
                                             DISPLAYSTREAM BBT (SELECTQ OPERATION
                                                                       (NIL (ffetch DDOPERATION
                                                                               of (fetch IMAGEDATA of
                                                                                                   DISPLAYSTREAM
                                                                                                   )))
                                                                       ((REPLACE PAINT INVERT ERASE)
                                                                           OPERATION)
                                                                       (\ILLEGAL.ARG OPERATION)))
                                      (SETQ BM (GetNewFragment BIGBMLIST))
                                      (SETQ HEIGHT BOTTOM)))
                       (freplace DDDestination of DD with BITMAP)
                       (freplace DDClippingTop of DD with ClippingTop)
                       (freplace DDClippingBottom of DD with ClippingBottom]
      (T (PROG ((DD (fetch IMAGEDATA of DISPLAYSTREAM))
                BITMAP)
               (\INSURETOPWDS DISPLAYSTREAM)                    ; bring the window to the top
               (SETQ BITMAP (ffetch DDDestination of DD))
               (COND
                  ((NOT (type? BIGBM BITMAP))
                   (\CLIPANDDRAWLINE (\DSPTRANSFORMX (OR (FIXP X1)
                                                         (FIXR X1))
                                                     DD)
                                     (\DSPTRANSFORMY (OR (FIXP Y1)
                                                         (FIXR Y1))
                                                     DD)
                                     (\DSPTRANSFORMX (OR (FIXP X2)
                                                         (FIXR X2))
                                                     DD)
                                     (\DSPTRANSFORMY (OR (FIXP Y2)
                                                         (FIXR Y2))
                                                     DD)
                                     [COND
                                        ((NULL WIDTH)
                                         1)
                                        ((OR (FIXP WIDTH)
                                             (FIXR WIDTH]
                                     (SELECTQ OPERATION
                                          (NIL (ffetch DDOPERATION of DD))
                                          ((REPLACE PAINT INVERT ERASE)
                                              OPERATION)
                                          (\ILLEGAL.ARG OPERATION))
                                     BITMAP
                                     (ffetch DDClippingLeft of DD)
                                     (SUB1 (ffetch DDClippingRight of DD))
                                     (ffetch DDClippingBottom of DD)
                                     (SUB1 (ffetch DDClippingTop of DD))
                                     DISPLAYSTREAM COLOR))
                  (T (PROG ((BIGBMLIST (fetch (BIGBM BIGBMLIST) of BITMAP))
                            (HEIGHT (BITMAPHEIGHT BITMAP))
                            BOTTOM BM CTop CBottom (ClippingTop (ffetch DDClippingTop of DD))
                            (ClippingBottom (ffetch DDClippingBottom of DD))
                            (YY1 (\DSPTRANSFORMY (OR (FIXP Y1)
                                                     (FIXR Y1))
                                                 DD))
                            (YY2 (\DSPTRANSFORMY (OR (FIXP Y2)
                                                     (FIXR Y2))
                                                 DD)))
                           (SETQ BM (GetNewFragment BIGBMLIST))
                           (while (AND BM (IGREATERP HEIGHT ClippingBottom))
```

```
                              do (SETQ BOTTOM (IDIFFERENCE HEIGHT (BITMAPHEIGHT BM)))
                                 [SETQ CTop (COND
                                              ((IGREATERP ClippingTop HEIGHT)
                                               (IDIFFERENCE HEIGHT BOTTOM))
                                              (T (IDIFFERENCE ClippingTop BOTTOM]
                                 (COND
                                    ((IGEQ CTop 0)
                                     [SETQ CBottom (COND
                                                     ((ILESSP ClippingBottom BOTTOM)
                                                      0)
                                                     (T (IDIFFERENCE ClippingBottom BOTTOM]
                                     (\CLIPANDDRAWLINE (\DSPTRANSFORMX (OR (FIXP X1)
                                                                           (FIXR X1))
                                                                       DD)
                                                       (IDIFFERENCE YY1 BOTTOM)
                                                       (\DSPTRANSFORMX (OR (FIXP X2)
                                                                           (FIXR X2))
                                                              DD)
                                                       (IDIFFERENCE YY2 BOTTOM)
                                                       [COND
                                                          ((NULL WIDTH)
                                                           1)
                                                          ((OR (FIXP WIDTH)
                                                               (FIXR WIDTH]
                                                       (SELECTQ OPERATION
                                                            (NIL (ffetch DDOPERATION of DD))
                                                            ((REPLACE PAINT INVERT ERASE)
                                                                OPERATION)
                                                            (\ILLEGAL.ARG OPERATION))
                                                       BM
                                                       (ffetch DDClippingLeft of DD)
                                                       (SUB1 (ffetch DDClippingRight of DD))
                                                       CBottom
                                                       (SUB1 CTop)
                                                       DISPLAYSTREAM COLOR)))
                                 (SETQ BM (GetNewFragment BIGBMLIST))
                                 (SETQ HEIGHT BOTTOM]          ; the generic case of MOVETO is used so that the hardcopy
                                                               ; streams get handled as well.
    (MOVETO X2 Y2 DISPLAYSTREAM])
```

(RELMOVETO

```
  [LAMBDA (DX DY STREAM)                                       (* rmk%: "25-AUG-83 18:13")
                                                               ; moves the position by a vector
    (DSPXPOSITION [IPLUS DX (DSPXPOSITION NIL (SETQ STREAM (\OUTSTREAMARG STREAM]
          STREAM)
    (DSPYPOSITION (IPLUS DY (DSPYPOSITION NIL STREAM))
          STREAM])
```

(MOVETOUPPERLEFT

```
  [LAMBDA (STREAM REGION)                                      (* hdj " 5-Jul-85 12:19")
    ;; moves the current position to the upper left corner so that the first line of text will all appear.
    (PROG [(ASCENT (FONTPROP (DSPFONT NIL STREAM)
                             'ASCENT]
          (COND
             ((AND REGION (OR (type? REGION REGION)
                              (\ILLEGAL.ARG REGION)))
              (MOVETO (fetch (REGION LEFT) of REGION)
                      (IDIFFERENCE (fetch (REGION PTOP) of REGION)
                             ASCENT)
                      STREAM))
             (T (MOVETO (DSPLEFTMARGIN NIL STREAM)
                        (IDIFFERENCE (fetch (REGION PTOP) of (DSPCLIPPINGREGION NIL STREAM))
                               ASCENT)
                        STREAM)))
          (RETURN STREAM])
```

)

```
(DEFINEQ
```

(\CLIPANDDRAWLINE

```
  [LAMBDA (X1 Y1 X2 Y2 WIDTH OPERATION BITMAP LEFT RIGHT BOTTOM TOP DS COLOR)
                                                               ; Edited 21-Aug-91 12:15 by jds
    ;; draws a line from {X1,Y1} to {X2,Y2} clipped to region specified by LEFT RIGHT BOTTOM and TOP.  This code is a transliterated version of the
    ;; BCPL routine that was in chat.

    ;; assumes that the width is at least 1

    ;; DS is passed so that window can be uninterruptably brought to top.

    (COND
       ((NOT (EQ (fetch (BITMAP BITMAPBITSPERPIXEL) of BITMAP)
                 1))                                           ; make adjustments in case of color.
        (SETQ COLOR (COLORNUMBERP (OR COLOR (DSPCOLOR NIL DS))
                                  (fetch (BITMAP BITMAPBITSPERPIXEL) of BITMAP)))
```

```
                                                                    ; (COND ((EQ OPERATION 'ERASE) ; treat erase as AND of
                                                                    ; background (SETQ COLOR (OPPOSITECOLOR COLOR
                                                                    ; (fetch (BITMAP BITMAPBITSPERPIXEL) of BITMAP)))))

      )
     (T (SETQ COLOR BLACKSHADE)))
  (PROG NIL
        (COND
           [(EQ X1 X2)                                              ; special case of vertical line.
            [COND
               ((IGREATERP WIDTH 2)
                (COND
                   [(EQ Y1 Y2)
                    ;; special case.  Since we don't know whether the guy is headed horizontally or vertically, put out a round brush This is a
                    ;; fairly infrequent case because I didn't get any bug reports on it in three years so efficiency is not a consideration.
                    (RETURN (.WHILE.TOP.DS. DS (\DRAWPOINT.DISPLAY (DSPDESTINATION NIL DS)
                                                                   X1 Y1 (LIST 'ROUND WIDTH COLOR)
                                                                   OPERATION]
                   (T (SETQ X1 (SETQ X2 (IDIFFERENCE X1 (LRSH (SUB1 WIDTH)
                                                              1]
            (PROG (MIN MAX)
                  (RETURN (COND
                             ([OR (IGREATERP X1 RIGHT)
                                  (IGEQ LEFT (SETQ X2 (IPLUS X1 WIDTH)))
                                  (IGREATERP (SETQ MIN (IMIN Y1 Y2))
                                         TOP)
                                  (IGREATERP BOTTOM (SETQ MAX (IMAX Y1 Y2]
                                                                    ; outside clippingregion.
                              NIL)
                             (T (.WHILE.TOP.DS. DS (BITBLT NIL 0 0 BITMAP (SETQ X1 (IMAX X1 LEFT))
                                                                  (SETQ MIN (IMAX MIN BOTTOM))
                                                                  (IDIFFERENCE (IMIN X2 (ADD1 RIGHT))
                                                                         X1)
                                                                  (ADD1 (IDIFFERENCE (IMIN MAX TOP)
                                                                                MIN))
                                                                  'TEXTURE OPERATION COLOR]
           [(EQ Y1 Y2)                                              ; special case of horizontal line.
            [COND
               ((IGREATERP WIDTH 2)
                (SETQ Y1 (SETQ Y2 (IDIFFERENCE Y1 (LRSH (SUB1 WIDTH)
                                                        1]
            (PROG (MIN MAX)
                  (RETURN (COND
                             ([OR (IGREATERP Y1 TOP)
                                  (IGEQ BOTTOM (SETQ Y2 (IPLUS Y1 WIDTH)))
                                  (IGREATERP (SETQ MIN (IMIN X1 X2))
                                         RIGHT)
                                  (IGREATERP LEFT (SETQ MAX (IMAX X1 X2]
                                                                    ; outside clippingregion.
                              NIL)
                             (T (.WHILE.TOP.DS. DS (BITBLT NIL 0 0 BITMAP (SETQ MIN (IMAX MIN LEFT))
                                                                  (SETQ Y1 (IMAX Y1 BOTTOM))
                                                                  (ADD1 (IDIFFERENCE (IMIN MAX RIGHT)
                                                                                MIN))
                                                                  (IDIFFERENCE (IMIN Y2 (ADD1 TOP))
                                                                         Y1)
                                                                  'TEXTURE OPERATION COLOR]
           ((EQ WIDTH 1)                                            ; special case of width 1
            (\CLIPANDDRAWLINE1 X1 Y1 X2 Y2 OPERATION BITMAP LEFT RIGHT BOTTOM TOP DS COLOR))
           ((IGREATERP (IABS (IDIFFERENCE X1 X2))
                   (IABS (IDIFFERENCE Y1 Y2)))                      ; slope is more horizontal, so make line grow in the positive y
                                                                    ; direction.
            [COND
               ((IGREATERP WIDTH 2)
                (PROG (HALFWIDTH)
                      (SETQ HALFWIDTH (LRSH (SUB1 WIDTH)
                                            1))
                      (SETQ Y1 (IDIFFERENCE Y1 HALFWIDTH))
                      (SETQ Y2 (IDIFFERENCE Y2 HALFWIDTH]
            (for I from Y1 to (SUB1 (IPLUS Y1 WIDTH)) as J from Y2
               do (\CLIPANDDRAWLINE1 X1 I X2 J OPERATION BITMAP LEFT RIGHT BOTTOM TOP DS COLOR)))
           (T                                                       ; slope is more vertical, so make line grow in the positive x
                                                                    ; direction.
              [COND
                 ((IGREATERP WIDTH 2)
                  (PROG (HALFWIDTH)
                        (SETQ HALFWIDTH (LRSH (SUB1 WIDTH)
                                              1))
                        (SETQ X1 (IDIFFERENCE X1 HALFWIDTH))
                        (SETQ X2 (IDIFFERENCE X2 HALFWIDTH]
              (for I from X1 to (SUB1 (IPLUS X1 WIDTH)) as J from X2
                 do (\CLIPANDDRAWLINE1 I Y1 J Y2 OPERATION BITMAP LEFT RIGHT BOTTOM TOP DS COLOR])
```

(\CLIPANDDRAWLINE1

```
  [LAMBDA (X1 Y1 X2 Y2 OPERATION BITMAP LEFT RIGHT BOTTOM TOP DS COLOR)
```

```
                                                                    (* JonL " 7-May-84 02:57")
;; LEFT, RIGHT, BOTTOM, TOP are set to the boundaries of the clipping region
;; DS is passed so that window can be uninterruptably brought to top.
(PROG (DX DY YMOVEUP HALFDX HALFDY (BMRASTERWIDTH (fetch BITMAPRASTERWIDTH of BITMAP)))
      (COND
         ((IGREATERP X1 X2)                                    ; switch points so DX is always positive.
          (SETQ HALFDX X1)
          (SETQ X1 X2)
          (SETQ X2 HALFDX)
          (SETQ HALFDX Y1)
          (SETQ Y1 Y2)
          (SETQ Y2 HALFDX)))                                   ; calculate differences and sign of Y movement.
      (SETQ HALFDX (LRSH (SETQ DX (IDIFFERENCE X2 X1))
                         1))
      (SETQ HALFDY (LRSH [SETQ DY (COND
                                     ((IGREATERP Y2 Y1)
                                      (SETQ YMOVEUP T)
                                      (IDIFFERENCE Y2 Y1))
                                     (T (IDIFFERENCE Y1 Y2]
                         1))
      (COND
         ((AND (IGEQ X1 LEFT)
               (IGEQ RIGHT X2)
               [COND
                  (YMOVEUP (AND (IGEQ Y1 BOTTOM)
                                (IGEQ TOP Y2)))
                  (T (AND (IGEQ Y2 BOTTOM)
                          (IGEQ TOP Y1]
               (EQ (fetch (BITMAP BITMAPBITSPERPIXEL) of BITMAP)
                   1))                                         ; line is completely visible, fast case.
          (.WHILE.TOP.DS. DS (\DRAWLINE1 X1 (SUB1 (\SFInvert BITMAP Y1))
                                         DX DY DX DY (COND
                                                        ((IGREATERP DX DY)
                                                                    ; X is the fastest mover.
                                                         HALFDX)
                                                        (T          ; y is the fastest mover.
                                                           HALFDY))
                                         (COND
                                            (YMOVEUP                ; y is moving in positive direction but bits are stored inversely
                                                     (IMINUS BMRASTERWIDTH))
                                            (T BMRASTERWIDTH))
                                         OPERATION
                                         (fetch BITMAPBASE of BITMAP)
                                         BMRASTERWIDTH)))
         (T
          (PROG ((CX1 X1)
                 (CY1 Y1)
                 (CX2 X2)
                 (CY2 Y2)
                 (CA1 (\CLIPCODE X1 Y1 LEFT RIGHT TOP BOTTOM))
                 (CA2 (\CLIPCODE X2 Y2 LEFT RIGHT TOP BOTTOM)))
                                                                    ; save the original points for the clipping computation.
                                                                    ; determine the sectors in which the points fall.
            CLIPLP
                [COND
                   ((NOT (EQ 0 (LOGAND CA1 CA2)))                   ; line is entirely out of clipping region
                    (RETURN NIL))
                   ((EQ 0 (IPLUS CA1 CA2))                          ; line is completely visible
                    ;; \SFInvert has an off by one bug that everybody else in LLDISPLAY uses to save computation so SUB1 from what you
                    ;; would expect.                                ; reuse the variable CA1
                    (RETURN (.WHILE.TOP.DS.
                             DS
                             (SELECTQ (fetch (BITMAP BITMAPBITSPERPIXEL) of BITMAP)
                                 (1 (\DRAWLINE1 CX1 (SUB1 (\SFInvert BITMAP CY1))
                                                (IDIFFERENCE CX2 CX1)
                                                (COND
                                                   (YMOVEUP (IDIFFERENCE CY2 CY1))
                                                   (T (IDIFFERENCE CY1 CY2)))
                                                DX DY (COND
                                                         ((IGREATERP DX DY)
                                                                    ; X is the fastest mover.
                                                          (IREMAINDER (IPLUS (ITIMES DY (IDIFFERENCE CX1 X1))
                                                                             HALFDX)
                                                                      DX))
                                                         (T         ; y is the fastest mover.
                                                            (IREMAINDER (IPLUS [ITIMES DX
                                                                                       (COND
                                                                                          (YMOVEUP (IDIFFERENCE
                                                                                                     CY1 Y1))
                                                                                          (T (IDIFFERENCE Y1 CY1]
                                                                               HALFDY)
                                                                        DY)))
                                                (COND
                                                   (YMOVEUP       ; y is moving in positive direction but bits are stored inversely
                                                          (IMINUS BMRASTERWIDTH))
```

```
                                        (T BMRASTERWIDTH))
                                    OPERATION
                                    (fetch BITMAPBASE of BITMAP)
                                    BMRASTERWIDTH))
                            ((4 8)
                                 (\DRAWCOLORLINE1 CX1 (SUB1 (\SFInvert BITMAP CY1))
                                        (IDIFFERENCE CX2 CX1)
                                        (COND
                                           (YMOVEUP (IDIFFERENCE CY2 CY1))
                                           (T (IDIFFERENCE CY1 CY2)))
                                        DX DY (COND
                                                 ((IGREATERP DX DY)
                                                        ; X is the fastest mover.
                                                  (IREMAINDER (IPLUS (ITIMES DY (IDIFFERENCE CX1 X1))
                                                                     HALFDX)
                                                         DX))
                                                 (T     ; y is the fastest mover.
                                                    (IREMAINDER (IPLUS [ITIMES DX
                                                                               (COND
                                                                                  (YMOVEUP (IDIFFERENCE
                                                                                            CY1 Y1))
                                                                                  (T (IDIFFERENCE Y1 CY1
                                                                                            ]
                                                                       HALFDY)
                                                          DY)))
                                        (COND
                                           (YMOVEUP     ; y is moving in positive direction but bits are stored inversely
                                                  (IMINUS BMRASTERWIDTH))
                                           (T BMRASTERWIDTH))
                                        OPERATION
                                        (fetch BITMAPBASE of BITMAP)
                                        BMRASTERWIDTH
                                        (fetch (BITMAP BITMAPBITSPERPIXEL) of BITMAP)
                                        COLOR))
                            (SHOULDNT]
                    [COND
                       ((NEQ CA1 0)
                        ;; now move point CX1 CY1 so that one of the coordinates is on one of the boundaries.  Which boundary is done first
                        ;; was copied from BCPL.
                        (COND
                           ((IGREATERP CA1 7)                    ; y1 less than bottom
                                                                 ; calculate the least X for which Y will be at bottom.
                            [SETQ CX1 (IPLUS X1 (\LEASTPTAT DX DY (IDIFFERENCE BOTTOM Y1]
                            (SETQ CY1 BOTTOM))
                           ((IGREATERP CA1 3)                    ; y1 is greater than top
                            [SETQ CX1 (IPLUS X1 (\LEASTPTAT DX DY (IDIFFERENCE Y1 TOP]
                            (SETQ CY1 TOP))
                           (T                                    ; x1 is less than left
                              [SETQ CY1 (COND
                                           [YMOVEUP (IPLUS Y1 (\LEASTPTAT DY DX (IDIFFERENCE LEFT X1]
                                           (T (IDIFFERENCE Y1 (\LEASTPTAT DY DX (IDIFFERENCE LEFT X1]
                               (SETQ CX1 LEFT)))
                        (SETQ CA1 (\CLIPCODE CX1 CY1 LEFT RIGHT TOP BOTTOM)))
                       (T                                        ; now move point CX2 CY2 so that one of the coordinates is on
                                                                 ; one of the boundaries
                          (COND
                             ((IGREATERP CA2 7)                  ; y2 less than bottom
                              [SETQ CX2 (IPLUS X1 (\GREATESTPTAT DX DY (IDIFFERENCE Y1 BOTTOM]
                              (SETQ CY2 BOTTOM))
                             ((IGREATERP CA2 3)                  ; y2 is greater than top
                              [SETQ CX2 (IPLUS X1 (\GREATESTPTAT DX DY (IDIFFERENCE TOP Y1]
                              (SETQ CY2 TOP))
                             (T                                  ; x2 is greater than right
                                [SETQ CY2 (COND
                                             [YMOVEUP (IPLUS Y1 (\GREATESTPTAT DY DX (IDIFFERENCE RIGHT X1]
                                             (T (IDIFFERENCE Y1 (\GREATESTPTAT DY DX (IDIFFERENCE RIGHT X1]
                                (SETQ CX2 RIGHT)))
                          (SETQ CA2 (\CLIPCODE CX2 CY2 LEFT RIGHT TOP BOTTOM]
                    (GO CLIPLP])
```

(**CLIPCODE**

```
  [LAMBDA (X Y LEFT RIGHT TOP BOTTOM)                            (* rrb " 4-DEC-80 10:34")
    ;; determines the sector code for a point wrt a region.  Used to clip things quickly.
    ;; RIGHT and TOP are one past the region.
    (COND
       ((LESSP X LEFT)                                           ; falls to left of region
        (COND
           ((GREATERP Y TOP)                                     ; left above
            5)
           ((LESSP Y BOTTOM)                                     ; left below
            9)
           (T                                                    ; left inside
              1)))
       ((GREATERP X RIGHT)                                       ; right
```

```
        (COND
           ((GREATERP Y TOP)                                    ; right above
            6)
           ((LESSP Y BOTTOM)                                    ; right below
            10)
           (T                                                   ; right inside
              2)))
       ((GREATERP Y TOP)                                        ; inside top
        4)
       ((LESSP Y BOTTOM)                                        ; inside below
        8)
       (T                                                       ; inside 0
          0])
```

(**LEASTPTAT**

```
[LAMBDA (DA DB THISB)                                           (* rrb " 7-JAN-82 11:56")
```

;; determines the smallest value in the dimension A that would give a B coordinate of THISB if a line were drawn from the point (0,0) with a slope of ;; DA/DB.

```
(COND
   ((IGREATERP DA DB)
    (ADD1 (IQUOTIENT (IPLUS (IDIFFERENCE (ITIMES THISB DA)
                                         (HALF DA))
                            -1)
                     DB)))
   (T (IQUOTIENT (IPLUS (ITIMES THISB DA)
                        (HALF DB))
                 DB])
```

(**GREATESTPTAT**

```
[LAMBDA (DA DB THISB)                                           (* rrb " 7-JAN-82 14:24")
```

;; determines the largest value in the dimension A that would give a B coordinate of THISB if a line were drawn from the point (0,0) with a slope of ;; DA/DB.

```
(COND
   ((IGREATERP DA DB)
    (IQUOTIENT (IPLUS (IDIFFERENCE (ITIMES (ADD1 THISB)
                                           DA)
                                   (HALF DA))
                      -1)
               DB))
   (T (IQUOTIENT (IPLUS (ITIMES THISB DA)
                        (HALF DB))
                 DB])
```

(**DRAWLINE1**

```
[LAMBDA (X0 Y0 XLIMIT YLIMIT DX DY CDL YINC MODE BITMAPBASE RASTERWIDTH)
                                                                (* mpl " 2-Jan-84 18:00")
```

;; this was changed to interface with the opcode for line drawing. It probably be incorporated into the places it is called.

;; draws a line starting at X0,Y0 at a slope of DX/DY until reaching either XLIMIT or YLIMIT with an initial overflow bucket size of CDL in MODE. ;; Arranged so that the clipping routines can determine what the exact location of the end point of the clipped line is wrt line drawing coordinates eg. ;; amount in overflow bucket. XLIMIT and YLIMIT are the number of points to be moved in that direction.

```
(\DRAWLINE.UFN (\ADDBASE BITMAPBASE (IPLUS (ITIMES Y0 RASTERWIDTH)
                                           (FOLDLO X0 BITSPERWORD)))
       (LOGAND X0 15)
       DX YINC DY (SELECTQ MODE
                           (INVERT 2)
                           (ERASE 1)
                           0)
       CDL
       (ADD1 XLIMIT)
       (ADD1 YLIMIT])
```

(**DRAWLINE.UFN**

```
[LAMBDA (FIRSTADDR FIRSTBIT XDELTA YINCR YDELTA OPERATIONCODE INITIALBUCKET PIXELSINX PIXELSINY)
                                                                (* jds " 6-Jan-86 11:27")
```

;; FIRSTADDR is the address of the word which contains the first point. FIRSTBIT is the address of the first bit in FIRSTADDR. XDELTA and ;; YDELTA are how far the complete line has to move in X and Y respectively; both are positive quantities. YINCR is the amount the address ;; should be incremented if the Y coordinate changes and can be either positive or negative. OPERATIONCODE is 0 for REPLACE, 1 for ERASE ;; and 2 for INVERT. INITIALBUCKET is between 0 and the maximum of DX and DY and gives the starting amount of the bucket used to ;; determine when to increment in the slower moving direction. PIXELSINX and PIXELSINY indicates how many pixels should be drawn in the X ;; and Y direction.

```
(DECLARE (LOCALVARS . T))
(PROG ((MASK (\BITMASK FIRSTBIT)))
      (COND
         [(IGEQ XDELTA YDELTA)                                  ; X is the fastest mover.
          (SELECTQ OPERATIONCODE
               (0 (.DRAWLINEX. 'REPLACE/PAINT))
               (1 (.DRAWLINEX. 'ERASE))
               (.DRAWLINEX. 'INVERT]
         (T                                                     ; Y is the fastest mover.
```

```
               (SELECTQ OPERATIONCODE
                    (0 (.DRAWLINEY. 'REPLACE/PAINT))
                    (1 (.DRAWLINEY. 'ERASE))
                    (.DRAWLINEY. 'INVERT])

)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS .DRAWLINEX. MACRO [(MODE)
                            (bind (NY _ 0) for PT from 1 to PIXELSINX
                               do                                  ; main loop
                                  [replace (BITMAPWORD BITS) of FIRSTADDR
                                     with (SELECTQ MODE
                                               (INVERT (LOGXOR MASK (fetch (BITMAPWORD BITS) of FIRSTADDR)))
                                               (ERASE (LOGAND (LOGXOR MASK WORDMASK)
                                                              (fetch (BITMAPWORD BITS) of FIRSTADDR)))
                                               (PROGN                ; case is PAINT or REPLACE.  Legality of OPERATION has
                                                                     ; been checked by \CLIPANDDRAWLINE1
                                                      (LOGOR MASK (fetch (BITMAPWORD BITS) of FIRSTADDR]
                                  [COND
                                     ([NOT (IGREATERP XDELTA (SETQ INITIALBUCKET (IPLUS INITIALBUCKET YDELTA]
                                                                     ; increment in the Y direction
                                      (COND
                                         ((EQ (SETQ NY (ADD1 NY))
                                              PIXELSINY)
                                          (RETURN)))
                                      (SETQ INITIALBUCKET (IDIFFERENCE INITIALBUCKET XDELTA))
                                      (SETQ FIRSTADDR (\ADDBASE FIRSTADDR YINCR]
                                  (SETQ MASK (LRSH MASK 1))
                                  (COND
                                     ((EQ 0 MASK)                    ; crossed word boundary
                                      (SETQ FIRSTADDR (\ADDBASE FIRSTADDR 1))
                                      (SETQ MASK 32768])

(PUTPROPS .DRAWLINEY. MACRO [(MODE)
                            (bind (NX _ 0) for PT from 1 to PIXELSINY
                               do                                  ; main loop
                                  [replace (BITMAPWORD BITS) of FIRSTADDR
                                     with (SELECTQ MODE
                                               (INVERT (LOGXOR MASK (fetch (BITMAPWORD BITS) of FIRSTADDR)))
                                               (ERASE (LOGAND (LOGXOR MASK WORDMASK)
                                                              (fetch (BITMAPWORD BITS) of FIRSTADDR)))
                                               (PROGN                ; case is PAINT or REPLACE.  Legality of OPERATION has
                                                                     ; been checked by \CLIPANDDRAWLINE1
                                                      (LOGOR MASK (fetch (BITMAPWORD BITS) of FIRSTADDR]
                                  [COND
                                     ([NOT (IGREATERP YDELTA (SETQ INITIALBUCKET (IPLUS INITIALBUCKET XDELTA]
                                      (COND
                                         ((EQ (SETQ NX (ADD1 NX))
                                              PIXELSINX)
                                          (RETURN)))
                                      (SETQ INITIALBUCKET (IDIFFERENCE INITIALBUCKET YDELTA))
                                      (SETQ MASK (LRSH MASK 1))
                                      (COND
                                         ((EQ 0 MASK)                ; crossed word boundary
                                          (SETQ FIRSTADDR (\ADDBASE FIRSTADDR 1))
                                          (SETQ MASK 32768]
                                  (SETQ FIRSTADDR (\ADDBASE FIRSTADDR YINCR])
)
)

;; Curves

(DEFINEQ
```

(**DRAWCIRCLE.DISPLAY**

```
  [LAMBDA (DISPLAYSTREAM CENTERX CENTERY RADIUS BRUSH DASHING)    (* kbr%: "15-Feb-86 22:24")
```

;; \DRAWCIRCLE.DISPLAY extended for color. Color is specified by either BRUSH or the DSPCOLOR of DS.

```
    (DECLARE (LOCALVARS . T))
    (COND
       ((OR (NOT (NUMBERP RADIUS))
            (ILESSP (SETQ RADIUS (FIXR RADIUS))
                    0))
        (\ILLEGAL.ARG RADIUS))
       ((EQ RADIUS 0)                                              ; don't draw anything.
        NIL)
       (DASHING                                                    ; draw it with the arc drawing code which does dashing.  Slow
                                                                   ; but effective.
```

;; the CDR removes the first point to work around a bug in curve drawing when closed and first and last points the same. AR 4623.0

```
              (DRAWCURVE (CDR (\COMPUTE.ARC.POINTS CENTERX CENTERY RADIUS 0 360))
                     T BRUSH DASHING DISPLAYSTREAM))
       (T (GLOBALRESOURCE \BRUSHBBT
```

```
(PROG (X Y D DestinationBitMap LEFT RIGHTPLUS1 TOP BOTTOM BRUSHWIDTH BRUSHHEIGHT LEFTMINUSBRUSH
         BOTTOMMINUSBRUSH TOPMINUSBRUSH BRUSHBM DESTINATIONBASE BRUSHBASE RASTERWIDTH
         BRUSHRASTERWIDTH NBITSRIGHTPLUS1 OPERATION HEIGHTMINUS1 CX CY BBT COLOR COLORBRUSHBASE
         NBITS DISPLAYDATA USERFN)
      (SETQ X 0)
      (SETQ Y RADIUS)
      (SETQ D (ITIMES 2 (IDIFFERENCE 1 RADIUS)))
      (SETQ BBT \BRUSHBBT)
      (SETQ DISPLAYDATA (fetch (STREAM IMAGEDATA) of DISPLAYSTREAM))
      (SETQ USERFN (AND (LITATOM BRUSH)
                        BRUSH))
```

;; many of these variables are used by the macro for \CURVEPT that passes them to \BBTCURVEPT and
;; .SETUP.FOR.\BBTCURVEPT. sets them up.

```
      (COND
         (USERFN                                    ; if calling user fn, don't bother with set up and leave points in
                                                    ; stream coordinates.
                 (SETQ CX CENTERX)
                 (SETQ CY CENTERY))
         (T (.SETUP.FOR.\BBTCURVEPT.)
            (SELECTQ NBITS
                 (1 (SETQ CX (\DSPTRANSFORMX (IDIFFERENCE CENTERX (FOLDLO BRUSHWIDTH 2))
                                             DISPLAYDATA)))
                 (4 (SETQ CX (\DSPTRANSFORMX (IDIFFERENCE CENTERX (FOLDLO (LRSH BRUSHWIDTH 2)
                                                                          2))
                                             DISPLAYDATA)))
                 (8 (SETQ CX (\DSPTRANSFORMX (IDIFFERENCE CENTERX (FOLDLO (LRSH BRUSHWIDTH 3)
                                                                          2))
                                             DISPLAYDATA)))
                 (24                                ; I doubt that this will be right.
                     (SETQ CX (\DSPTRANSFORMX (IDIFFERENCE CENTERX (FOLDLO (IQUOTIENT
                                                                                BRUSHWIDTH 24
                                                                                )
                                                                           2))
                                              DISPLAYDATA)))
                 (SHOULDNT))                        ; take into account the brush thickness.
            (SETQ CY (\DSPTRANSFORMY (IDIFFERENCE CENTERY (FOLDLO BRUSHHEIGHT 2))
                                     DISPLAYDATA))
```

;; Move the window to top while interruptable, but verify that it is still there uninterruptably with drawing points

```
            (\INSURETOPWDS DISPLAYSTREAM)))
      [COND
         ((EQ RADIUS 1)                             ; put a single brush down.
                                                    ; draw the top and bottom most points.
          [COND
             (USERFN (APPLY* USERFN CX CY DISPLAYSTREAM))
             (T (.WHILE.TOP.DS. DISPLAYSTREAM (\CURVEPT CX CY]
          (RETURN))
         (T                                         ; draw the top and bottom most points.
            (COND
               (USERFN (APPLY* USERFN CX (IPLUS CY RADIUS)
                               DISPLAYSTREAM)
                       (APPLY* USERFN CX (IDIFFERENCE CY RADIUS)
                               DISPLAYSTREAM))
               (T (.WHILE.TOP.DS. DISPLAYSTREAM (\CURVEPT CX (IPLUS CY RADIUS))
                                  (\CURVEPT CX (IDIFFERENCE CY RADIUS]
LP                                                  ; (UNFOLD x 2) is used instead of (ITIMES x 2)
      [COND
         [(IGREATERP 0 D)
          (SETQ X (ADD1 X))
          (COND
             ((IGREATERP (UNFOLD (IPLUS D Y)
                                 2)
                         1)
              (SETQ D (IPLUS D (UNFOLD (IDIFFERENCE X Y)
                                       2)
                             4))
              (SETQ Y (SUB1 Y)))
             (T (SETQ D (IPLUS D (UNFOLD X 2)
                                 1]
         ((OR (EQ 0 D)
              (IGREATERP X D))
          (SETQ X (ADD1 X))
          (SETQ D (IPLUS D (UNFOLD (IDIFFERENCE X Y)
                                   2)
                         4))
          (SETQ Y (SUB1 Y)))
         (T (SETQ D (IPLUS (IDIFFERENCE D (UNFOLD Y 2))
                           3))
            (SETQ Y (SUB1 Y]
      (COND
         [(EQ Y 0)
```

;; left most and right most points are drawn specially so that they are not duplicated which leaves a hole in XOR
;; mode.

```
          (COND
             (USERFN (APPLY* USERFN (IPLUS CX X)
```

```
                                        CY DISPLAYSTREAM)
                                    (APPLY* USERFN (IDIFFERENCE CX X)
                                           CY DISPLAYSTREAM))
                                (T (.WHILE.TOP.DS. DISPLAYSTREAM (\CURVEPT (IPLUS CX X)
                                                                          CY)
                                          (\CURVEPT (IDIFFERENCE CX X)
                                                 CY]
                            (T [COND
                                  (USERFN (APPLY* USERFN (IPLUS CX X)
                                                 (IPLUS CY Y)
                                                 DISPLAYSTREAM)
                                          (APPLY* USERFN (IDIFFERENCE CX X)
                                                 (IPLUS CY Y)
                                                 DISPLAYSTREAM)
                                          (APPLY* USERFN (IPLUS CX X)
                                                 (IDIFFERENCE CY Y)
                                                 DISPLAYSTREAM)
                                          (APPLY* USERFN (IDIFFERENCE CX X)
                                                 (IDIFFERENCE CY Y)
                                                 DISPLAYSTREAM))
                                  (T (.WHILE.TOP.DS. DISPLAYSTREAM (\CIRCLEPTS CX CY X Y]
                               (GO LP)))
                        (MOVETO CENTERX CENTERY DISPLAYSTREAM)
                        (RETURN NIL])
```

(**DRAWARC.DISPLAY**

```
[LAMBDA (STREAM CENTERX CENTERY RADIUS STARTANGLE NDEGREES BRUSH DASHING)
                                                                  ; draws an arc on the display
   (\DRAWARC.GENERIC STREAM CENTERX CENTERY RADIUS STARTANGLE NDEGREES BRUSH DASHING])
```

(**DRAWARC.GENERIC**

```
[LAMBDA (STREAM CENTERX CENTERY RADIUS STARTANGLE NDEGREES BRUSH DASHING)
                                                                  (* rrb " 4-Oct-85 18:23")
                                                                  ; draws an arc by drawing a curve.
   (COND
      ((AND (GREATERP 360 NDEGREES)
            (LESSP -360 NDEGREES))
       (DRAWCURVE (\COMPUTE.ARC.POINTS CENTERX CENTERY RADIUS STARTANGLE NDEGREES)
              NIL BRUSH DASHING STREAM))
      (T                                                          ; use circle drawing which could be faster
         (DRAWCIRCLE CENTERX CENTERY RADIUS BRUSH DASHING STREAM])
```

(**COMPUTE.ARC.POINTS**

```
[LAMBDA (CENTERX CENTERY RADIUS STARTANGLE NDEGREES)              (* DECLARATIONS%: FLOATING)
                                                                  (* rrb "30-Oct-85 11:48")
```

;; computes a list of knots that a spline goes through to make an arc

```
(PROG ((ANGLESIZE (COND
                      ((OR (GREATERP NDEGREES 360.0)
                           (GREATERP -360.0 NDEGREES))
                       360.0)
                      (T NDEGREES)))
       ANGLEINCR)
```

;; calculate an increment close to 10.0 that is exact but always have at least 5 knots and don't have more than a knot every 5 pts

```
      [SETQ ANGLEINCR (FQUOTIENT ANGLESIZE (IMIN (IMAX (ABS (FIX (FQUOTIENT ANGLESIZE 10.0)))
                                                       5)
                                                 (PROGN      ; don't have more than a knot every 5 pts
                                                        (IMAX (ABS (FIX (QUOTIENT (TIMES RADIUS 6.3
                                                                                         (QUOTIENT ANGLESIZE
                                                                                                360.0))
                                                                                  4)))
                                                              3]
```

;; go from initial point to just past the last point. The just past (PLUS BETA (QUOTIENT ANGLEINCR 5.0)) picks up the case where the floating pt
;; rounding error accumulates to be greater than the last point when it is very close to it.

```
      (RETURN (for ANGLE from STARTANGLE to (PLUS STARTANGLE ANGLESIZE (QUOTIENT ANGLEINCR 5.0)) by ANGLEINCR
                 collect (create POSITION
                                 XCOORD _ [FIXR (PLUS CENTERX (TIMES RADIUS (COS ANGLE]
                                 YCOORD _ (FIXR (PLUS CENTERY (TIMES RADIUS (SIN ANGLE])
```

(**DRAWELLIPSE.DISPLAY**

```
[LAMBDA (DISPLAYSTREAM CENTERX CENTERY SEMIMINORRADIUS SEMIMAJORRADIUS ORIENTATION BRUSH DASHING)
                                                                  ; Edited 12-Apr-88 23:58 by FS
   (DECLARE (LOCALVARS . T))
```

;; Draws an ellipse. At ORIENTATION 0, the semimajor axis is horizontal, the semiminor axis vertical. Orientation is positive in the
;; counterclockwise direction. The current location in the stream is left at the center of the ellipse.

```
(PROG ((CENTERX (FIXR CENTERX))
       (CENTERY (FIXR CENTERY))
       (SEMIMINORRADIUS (FIXR SEMIMINORRADIUS))
       (SEMIMAJORRADIUS (FIXR SEMIMAJORRADIUS)))
      (COND
```

```
            ((OR (EQ 0 SEMIMINORRADIUS)
                 (EQ 0 SEMIMAJORRADIUS))
             (MOVETO CENTERX CENTERY DISPLAYSTREAM)
             (RETURN)))
          (COND
             ((ILESSP SEMIMINORRADIUS 1)
              (\ILLEGAL.ARG SEMIMINORRADIUS))
             ((ILESSP SEMIMAJORRADIUS 1)
              (\ILLEGAL.ARG SEMIMAJORRADIUS))
             ((OR (NULL ORIENTATION)
                  (EQ SEMIMINORRADIUS SEMIMAJORRADIUS))
              (SETQ ORIENTATION 0))
             ((NULL (NUMBERP ORIENTATION))
              (\ILLEGAL.ARG ORIENTATION)))
```

;; If dashing, draw it with the curve drawing code which can do dashing

```
          (COND
             (DASHING (\DRAWELLIPSE.GENERIC DISPLAYSTREAM CENTERX CENTERY SEMIMINORRADIUS SEMIMAJORRADIUS
                           ORIENTATION BRUSH DASHING)
                      (RETURN)))
```

;; If degenerate ellipse, attempt circumvention of Pitteway breakdown by trying spline code instead, which appears more numerically stable (see ;; AR6502)

```
          (COND
             ((< 40 (/ SEMIMAJORRADIUS SEMIMINORRADIUS))
              (\DRAWELLIPSE.GENERIC DISPLAYSTREAM CENTERX CENTERY SEMIMINORRADIUS SEMIMAJORRADIUS ORIENTATION
                     BRUSH DASHING)
              (RETURN)))
```

;;; This function is the implementation of the algorithm given in 'Algorithm for drawing ellipses or hyperbolae with a digital plotter' by Pitteway appearing ;;; in Computer Journal 10: (3) Nov 1967.0 The input parameters are used to determine the ellipse equation (1/8) Ayy+ (1/8) Bxx+ (1/4) Gxy+ (1/4) Ux+ ;;; (1/4) Vy= (1/4) K which specifies a translated version of the desired ellipse. This ellipse passes through the mesh point (0,0), the initial point of the ;;; algorithm. The power of 2 factors reflect an implementation convenience.

```
          (GLOBALRESOURCE \BRUSHBBT
                 (PROG (DestinationBitMap LEFT RIGHTPLUS1 BOTTOM TOP BOTTOMMINUSBRUSH TOPMINUSBRUSH
                              LEFTMINUSBRUSH DESTINATIONBASE BRUSHBASE BRUSHHEIGHT BRUSHWIDTH RASTERWIDTH
                              BRUSHRASTERWIDTH BRUSHBM OPERATION HEIGHTMINUS1 (BBT \BRUSHBBT)
                              (cosOrientation (COS ORIENTATION))
                              (sinOrientation (SIN ORIENTATION))
                              (SEMIMINORRADIUSSQUARED (ITIMES SEMIMINORRADIUS SEMIMINORRADIUS))
                              (SEMIMAJORRADIUSSQUARED (ITIMES SEMIMAJORRADIUS SEMIMAJORRADIUS))
                              (x 0)
                              (y 0)
                              (x2 1)
                              x1 y1 y2 k1 k2 k3 a b d w A B G U V K CX CY yOffset CYPlusOffset CYMinusOffset
                              NBITSRIGHTPLUS1 COLORBRUSHBASE COLOR NBITS (DISPLAYDATA (fetch IMAGEDATA
                                                                                      of DISPLAYSTREAM))
                              (USERFN (AND (LITATOM BRUSH)
                                           BRUSH)))
```

;; many of these variables are used by the macro for \CURVEPT that passes them to \BBTCURVEPT and ;; .SETUP.FOR.\BBTCURVEPT. sets them up.

```
                       (COND
                          (USERFN                                  ; if calling user fn, don't bother with set up and leave points in
                                                                   ; window coordinates.
                                  (SETQ CX CENTERX)
                                  (SETQ CY CENTERY))
                          (T (.SETUP.FOR.\BBTCURVEPT.)             ; take into account the brush thickness.
                             (SELECTQ NBITS
                                  (1 (SETQ CX (\DSPTRANSFORMX (IDIFFERENCE CENTERX (FOLDLO BRUSHWIDTH 2))
                                                              DISPLAYDATA)))
                                  (4 (SETQ CX (\DSPTRANSFORMX (IDIFFERENCE CENTERX (FOLDLO (LRSH BRUSHWIDTH 2)
                                                                                           2))
                                                              DISPLAYDATA)))
                                  (8 (SETQ CX (\DSPTRANSFORMX (IDIFFERENCE CENTERX (FOLDLO (LRSH BRUSHWIDTH 3)
                                                                                           2))
                                                              DISPLAYDATA)))
                                  (SHOULDNT))
                             (SETQ CY (\DSPTRANSFORMY (IDIFFERENCE CENTERY (FOLDLO BRUSHHEIGHT 2))
                                                      DISPLAYDATA))
                             ;; Move the window to top while interruptable, but verify that it is still there uninterruptably with drawing points
                             (\INSURETOPWDS DISPLAYSTREAM)))
                       (SETQ A (FPLUS (FTIMES SEMIMAJORRADIUSSQUARED cosOrientation cosOrientation)
                                      (FTIMES SEMIMINORRADIUSSQUARED sinOrientation sinOrientation)))
                       (SETQ B (LSH (FIXR (FPLUS (FTIMES SEMIMINORRADIUSSQUARED cosOrientation cosOrientation)
                                                 (FTIMES SEMIMAJORRADIUSSQUARED sinOrientation sinOrientation)))
                                    3))
                       (SETQ G (FTIMES cosOrientation sinOrientation (LSH (IDIFFERENCE SEMIMINORRADIUSSQUARED
                                                                                       SEMIMAJORRADIUSSQUARED)
                                                                          1)))
                       [SETQ yOffset (FIXR (FQUOTIENT (ITIMES SEMIMINORRADIUS SEMIMAJORRADIUS)
                                                      (SQRT A]
                       (SETQ CYPlusOffset (IPLUS CY yOffset))
                       (SETQ CYMinusOffset (IDIFFERENCE CY yOffset))
                       (SETQ U (LSH (FIXR (FTIMES A (LSH yOffset 1)))
```

```
                  2))
      (SETQ V (LSH (FIXR (FTIMES G yOffset))
                  2))
      (SETQ K (LSH [FIXR (FDIFFERENCE (ITIMES SEMIMINORRADIUSSQUARED SEMIMAJORRADIUSSQUARED)
                               (FTIMES A (ITIMES yOffset yOffset]
                  2))
      (SETQ A (LSH (FIXR A)
                  3))
      (SETQ G (LSH (FIXR G)
                  2))
```

;; The algorithm is incremental and iterates through the octants of a cartesian plane. The octants are labeled from 1 through 8 ;; beginning above the positive X axis and proceeding counterclockwise. Decisions in making the incremental steps are ;; determined according to the error term d which is updated according to the curvature terms a and b. k1, k2, and k3 are used ;; to correct the error and curvature terms at octant boundaries. The initial values of these terms depends on the octant in which ;; drawing begins. The initial move steps (x1,y1) and (x2,y2) also depend on the starting octant.

```
      [COND
         [(ILESSP (ABS U)
                  (ABS V))
          (SETQ x1 0)
          (COND
             [(MINUSP V)                          ; start in octant 2
              (SETQ y1 1)
              (SETQ y2 1)
              (SETQ k1 (IMINUS A))
              (SETQ k2 (IDIFFERENCE k1 G))
              (SETQ k3 (IDIFFERENCE k2 (IPLUS B G)))
              (SETQ b (IPLUS U (RSH (IPLUS A G)
                                    1)))
              (SETQ a (IMINUS (IPLUS b V)))
              (SETQ d (IPLUS b (RSH B 3)
                             (RSH V 1)
                             (IMINUS K]
             (T                                   ; start in octant 7
                (SETQ y1 -1)
                (SETQ y2 -1)
                (SETQ k1 A)
                (SETQ k2 (IDIFFERENCE k1 G))
                (SETQ k3 (IPLUS k2 B (IMINUS G)))
                (SETQ b (IPLUS U (RSH (IDIFFERENCE G A)
                                      1)))
                (SETQ a (IDIFFERENCE V b))
                (SETQ d (IPLUS b K (IMINUS (IPLUS (RSH V 1)
                                                  (RSH B 3]
         (T (SETQ x1 1)
            (SETQ y1 0)
            (COND
               [(MINUSP V)                        ; start in octant 1
                (SETQ y2 1)
                (SETQ k1 B)
                (SETQ k2 (IPLUS k1 G))
                (SETQ k3 (IPLUS k2 A G))
                [SETQ b (IMINUS (IPLUS V (RSH (IPLUS B G)
                                              1]
                (SETQ a (IDIFFERENCE U b))
                (SETQ d (IPLUS b K (IMINUS (IPLUS (RSH A 3)
                                                  (RSH U 1]
               (T                                 ; start in octant 8
                  (SETQ y2 -1)
                  (SETQ k1 (IMINUS B))
                  (SETQ k2 (IPLUS k1 G))
                  (SETQ k3 (IPLUS k2 G (IMINUS A)))
                  (SETQ b (IPLUS V (RSH (IDIFFERENCE B G)
                                        1)))
                  (SETQ a (IDIFFERENCE U b))
                  (SETQ d (IPLUS b (RSH A 3)
                                 (IMINUS (IPLUS K (RSH U 1]
```

;; The ellipse equation describes an ellipse of the desired size and ORIENTATION centered at (0,0) and then dropped yOffset ;; mesh points so that it will pass through (0,0). Thus, the intended starting point is (CX, CY+yOffset) where (CX, CY) is the ;; center of the desired ellipse. Drawing is accomplished with point relative steps. In each octant, the error term d is used to ;; choose between move 1 (an axis move) and move 2 (a diagonal move).

```
MOVE
      [COND
         ((MINUSP d)                              ; move 1
          (SETQ x (IPLUS x x1))
          (SETQ y (IPLUS y y1))
          (SETQ b (IDIFFERENCE b k1))
          (SETQ a (IPLUS a k2))
          (SETQ d (IPLUS b d)))
         (T                                       ; move 2
            (SETQ x (IPLUS x x2))
            (SETQ y (IPLUS y y2))
            (SETQ b (IDIFFERENCE b k2))
            (SETQ a (IPLUS a k3))
            (SETQ d (IDIFFERENCE d a]
      (COND
```

```
                        ((MINUSP x)
                         (MOVETO CENTERX CENTERY DISPLAYSTREAM)
                         (RETURN NIL)))
                     [COND
                        (USERFN (APPLY* USERFN (IPLUS CX x)
                                       (IPLUS CYPlusOffset y)
                                       DISPLAYSTREAM)
                                (APPLY* USERFN (IDIFFERENCE CX x)
                                       (IDIFFERENCE CYMinusOffset y)
                                       DISPLAYSTREAM))
                        (T (.WHILE.TOP.DS. DISPLAYSTREAM (\CURVEPT (IPLUS CX x)
                                                                   (IPLUS CYPlusOffset y))
                                   (\CURVEPT (IDIFFERENCE CX x)
                                             (IDIFFERENCE CYMinusOffset y]
                     (AND (MINUSP b)
                          (GO SQUARE))
                DIAGONAL
                     (OR (MINUSP a)
                         (GO MOVE))                              ; diagonal octant change
                     (SETQ x1 (IDIFFERENCE x2 x1))
                     (SETQ y1 (IDIFFERENCE y2 y1))
                     (SETQ w (IDIFFERENCE (LSH k2 1)
                                  k3))
                     (SETQ k1 (IDIFFERENCE w k1))
                     (SETQ k2 (IDIFFERENCE k2 k3))
                     (SETQ k3 (IMINUS k3))
                     [SETQ b (IPLUS b a (IMINUS (RSH (ADD1 k2)
                                                     1]
                     [SETQ d (IPLUS b (RSH (IPLUS k3 4)
                                           3)
                                  (IMINUS d)
                                  (IMINUS (RSH (ADD1 a)
                                               1]
                     (SETQ a (IDIFFERENCE (RSH (ADD1 w)
                                               1)
                                  a))
                     (OR (MINUSP b)
                         (GO MOVE))
                SQUARE
                                                                 ; square octant change
                     [COND
                        ((EQ 0 x1)
                         (SETQ x2 (IMINUS x2)))
                        (T (SETQ y2 (IMINUS y2]
                     (SETQ w (IDIFFERENCE k2 k1))
                     (SETQ k1 (IMINUS k1))
                     (SETQ k2 (IPLUS w k1))
                     (SETQ k3 (IDIFFERENCE (LSH w 2)
                                   k3))
                     (SETQ b (IDIFFERENCE (IMINUS b)
                                  w))
                     (SETQ d (IDIFFERENCE (IDIFFERENCE b a)
                                  d))
                     (SETQ a (IDIFFERENCE (IDIFFERENCE a w)
                                  (LSH b 1)))
                     (GO DIAGONAL])
```

(**DRAWCURVE.DISPLAY**

```
[LAMBDA (DISPLAYSTREAM KNOTS CLOSED BRUSH DASHING)               ; Edited 9-Jan-87 16:49 by rrb
```

;; draws a spline curve with a given brush.

```
(GLOBALRESOURCE \BRUSHBBT (PROG ((BBT \BRUSHBBT)
                                 (DASHLST (\GOOD.DASHLST DASHING BRUSH)))
                                (SELECTQ (LENGTH KNOTS)
                                     (0                          ; No knots => empty curve rather than error?
                                        NIL)
                                     (1                          ; only one knot, put down a brush shape
                                        (OR (type? POSITION (CAR KNOTS))
                                            (ERROR "bad knot" (CAR KNOTS)))
                                        (\DRAWPOINT.DISPLAY DISPLAYSTREAM (fetch XCOORD of (CAR KNOTS))
                                                (fetch YCOORD of (CAR KNOTS))
                                                BRUSH))
                                     (2 (OR (type? POSITION (CAR KNOTS))
                                            (ERROR "bad knot" (CAR KNOTS)))
                                        (OR (type? POSITION (CADR KNOTS))
                                            (ERROR "bad knot" (CADR KNOTS)))
                                        (\LINEWITHBRUSH (fetch XCOORD of (CAR KNOTS))
                                                (fetch YCOORD of (CAR KNOTS))
                                                (fetch XCOORD of (CADR KNOTS))
                                                (fetch YCOORD of (CADR KNOTS))
                                                BRUSH DASHLST DISPLAYSTREAM BBT))
                                     (\CURVE2 (PARAMETRICSPLINE KNOTS CLOSED)
                                           BRUSH DASHLST BBT DISPLAYSTREAM))
                                (RETURN DISPLAYSTREAM])
```

(\DRAWPOINT.DISPLAY

```
[LAMBDA (DISPLAYSTREAM X Y BRUSH OPERATION)                    (* rrb "17-Sep-86 17:51")
```

;; draws a brush point at position X Y

;; this is used in 4, 8, and 24 bit per pixel bitmaps as well. For these, it may be should call BITMAPWIDTH instead of fetching.

```
(PROG ((BRUSHBM (\GETBRUSH BRUSH)))                    ; SUB1 is to put extra bit of even brush on the top or left.
      (RETURN (BITBLT BRUSHBM 0 0 DISPLAYSTREAM [IDIFFERENCE X (HALF (SUB1 (fetch (BITMAP BITMAPWIDTH)
                                                                              of BRUSHBM]
                      [IDIFFERENCE Y (HALF (SUB1 (fetch (BITMAP BITMAPHEIGHT) of BRUSHBM]
                      NIL NIL NIL (SELECTQ (OR OPERATION (DSPOPERATION NIL DISPLAYSTREAM))
                                           (REPLACE 'PAINT)
                                           OPERATION])
```

(\DRAWPOLYGON.DISPLAY

```
[LAMBDA (STREAM POINTS CLOSED BRUSH DASHING)                   ; Edited 13-Apr-88 14:14 by FS
```

;; Somewhat less generic version of drawpolygon that calls \drawline.display. Brush must be a brush (guaranteed in DRAWPOLYGON) other ;; users must also ensure.

;; This is different than drawline.generic, because drawline.display will use width argument instead of bltting brushes around. That way you can get ;; shades, dspoperation, eventually.

```
(PROG [COLOR (PTBRUSH (COND
                          ((EQ (fetch (BRUSH BRUSHSHAPE) of BRUSH)
                               'ROUND)
                           BRUSH)
                          (T (create BRUSH using BRUSH BRUSHSHAPE _ 'ROUND]
      (SETQ COLOR (fetch (BRUSH BRUSHCOLOR) of PTBRUSH))
      (for PTAIL on POINTS while (CDR PTAIL) do (\DRAWLINE.DISPLAY STREAM (fetch (POSITION XCOORD)
                                                                             of (CAR PTAIL))
                                                                 (ffetch (POSITION YCOORD) of (CAR PTAIL))
                                                                 (fetch (POSITION XCOORD) of (CADR PTAIL))
                                                                 (ffetch (POSITION YCOORD) of (CADR PTAIL))
                                                                 (fetch (BRUSH BRUSHSIZE) of BRUSH)
                                                                 NIL COLOR DASHING)
                                                       ; put a brush between lines so it looks better. It's not mitered this
                                                       ; way but better than not.
                                                (\DRAWPOINT.DISPLAY STREAM (fetch (POSITION XCOORD)
                                                                              of (CADR POINTS))
                                                                 (fetch (POSITION YCOORD) of (CADR POINTS))
                                                                 PTBRUSH
                                                                 'NIL)
        finally (COND
                    ((AND CLOSED (CDDR POINTS))                ; draw the closing line.
                     (\DRAWLINE.DISPLAY STREAM (fetch (POSITION XCOORD) of (CAR PTAIL))
                            (ffetch (POSITION YCOORD) of (CAR PTAIL))
                            (fetch (POSITION XCOORD) of (CAR POINTS))
                            (ffetch (POSITION YCOORD) of (CAR POINTS))
                            (fetch (BRUSH BRUSHSIZE) of BRUSH)
                            NIL COLOR DASHING)))
                (OR (NULL (CDR POINTS))
                    (\DRAWPOINT.DISPLAY STREAM (fetch (POSITION XCOORD) of (CAR POINTS))
                            (fetch (POSITION YCOORD) of (CAR POINTS))
                            PTBRUSH NIL])
```

(\LINEWITHBRUSH

```
[LAMBDA (X1 Y1 X2 Y2 BRUSH DASHLST DISPLAYSTREAM BBT OPERATION)
                                                               ; Edited 29-Oct-87 17:40 by scp
```

;; draws a line with a brush on a guaranteed display-stream DISPLAYSTREAM

```
(DECLARE (LOCALVARS . T))
(PROG (DestinationBitMap LEFT RIGHTPLUS1 TOP BOTTOM BRUSHWIDTH BRUSHHEIGHT LEFTMINUSBRUSH BOTTOMMINUSBRUSH
             TOPMINUSBRUSH BRUSHBM DESTINATIONBASE BRUSHBASE RASTERWIDTH BRUSHRASTERWIDTH NBITSRIGHTPLUS1
             HEIGHTMINUS1 COLOR COLORBRUSHBASE NBITS HALFBRUSHWIDTH HALFBRUSHHEIGHT DX DY YINC CDL
             (DASHON T)
             (DASHTAIL DASHLST)
             (DASHCNT (CAR DASHLST))
             (DISPLAYDATA (fetch IMAGEDATA of DISPLAYSTREAM))
             (USERFN (AND (LITATOM BRUSH)
                          BRUSH))
             (DISPLAYDATA (fetch IMAGEDATA of DISPLAYSTREAM)))
```

;; many of these variables are used by the macro for \CURVEPT that passes them to \BBTCURVEPT and .SETUP.FOR.\BBTCURVEPT. sets ;; them up. ; move the display stream position before the coordinates are ; clobbered.

```
      (COND
         ((NOT USERFN)
          (.SETUP.FOR.\BBTCURVEPT.)
          (SELECTQ NBITS
               (1                                              ; SUB1 is so that the extra bit goes on the top and right as it is
                                                               ; documented as doing for lines.
                  (SETQ X1 (\DSPTRANSFORMX (IDIFFERENCE X1 (SETQ HALFBRUSHWIDTH (FOLDLO (SUB1 BRUSHWIDTH)
                                                                                        2)))
                           DISPLAYDATA)))
               (4 (SETQ X1 (\DSPTRANSFORMX (IDIFFERENCE X1 (SETQ HALFBRUSHWIDTH (FOLDLO (LRSH (SUB1
                                                                                                BRUSHWIDTH
                                                                                                )
```

```
                                                                                                    2)
                                                                                                2)))
                                    DISPLAYDATA)))
                (8 (SETQ X1 (\DSPTRANSFORMX (IDIFFERENCE X1 (SETQ HALFBRUSHWIDTH (FOLDLO (LRSH (SUB1
                                                                                                        BRUSHWIDTH
                                                                                                         )
                                                                                                    3)
                                                                                                2)))
                                    DISPLAYDATA)))
                (SHOULDNT))
            (SETQ X2 (\DSPTRANSFORMX (IDIFFERENCE X2 HALFBRUSHWIDTH)
                             DISPLAYDATA))
            (SETQ Y1 (\DSPTRANSFORMY (IDIFFERENCE Y1 (SETQ HALFBRUSHHEIGHT (FOLDLO (SUB1 BRUSHHEIGHT)
                                                                                    2)))
                             DISPLAYDATA))                          ; take into account the brush thickness.
            (SETQ Y2 (\DSPTRANSFORMY (IDIFFERENCE Y2 HALFBRUSHHEIGHT)
                             DISPLAYDATA))

            ;; Move the window to top while interruptable, but verify that it is still there uninterruptably with drawing points
            (\INSURETOPWDS DISPLAYSTREAM)))                         ; arrange things so that dx is positive.
        (COND
           ((IGREATERP X1 X2)                                       ; switch points
            (swap X1 X2)
            (swap Y1 Y2)))
        (SETQ DX (ADD1 (IDIFFERENCE X2 X1)))
        [SETQ DY (ADD1 (COND
                          ((IGREATERP Y2 Y1)
                           (SETQ YINC 1)
                           (IDIFFERENCE Y2 Y1))
                          (T (SETQ YINC -1)
                             (IDIFFERENCE Y1 Y2]
        [SETQ CDL (HALF (COND
                           ((IGREATERP DX DY)                       ; set up the bucket so that the ends will be the same.
                            (IREMAINDER DX DY))
                           (T (IREMAINDER DY DX]
        [COND
           [USERFN                                                  ; if user function is being called, don't bother bringing window to
                                                                    ; top uninterruptably.
                   (COND
                      ((IGEQ DX DY)                                 ; X is the fastest mover.
                       (until (IGREATERP X1 X2) do                  ; main loop
                                                (COND
                                                   (DASHON (APPLY* USERFN X1 Y1 DISPLAYSTREAM)))
                                                [COND
                                                   (DASHTAIL  ; do dashing.
                                                           (COND
                                                              ((EQ 0 (SETQ DASHCNT (SUB1 DASHCNT)))
                                                               (SETQ DASHON (NOT DASHON))
                                                               (SETQ DASHTAIL (OR (LISTP (CDR DASHTAIL))
                                                                                  DASHLST))
                                                               (SETQ DASHCNT (CAR DASHTAIL]
                                                [COND
                                                   ((NOT (IGREATERP DX (add CDL DY)))
                                                    (add Y1 YINC)
                                                    (COND
                                                       ((COND
                                                           ((EQ YINC -1)
                                                            (ILESSP Y1 Y2))
                                                           ((IGREATERP Y1 Y2)))
                                                        (RETURN)))
                                                    (SETQ CDL (IDIFFERENCE CDL DX]
                                                (add X1 1)))
                      (T                                            ; Y is the fastest mover.
                         (until (COND
                                   ((EQ YINC -1)
                                    (ILESSP Y1 Y2))
                                   ((IGREATERP Y1 Y2)))
                            do                                      ; main loop
                                (COND
                                   (DASHON (APPLY* USERFN X1 Y1 DISPLAYSTREAM)))
                                [COND
                                   (DASHTAIL                        ; do dashing.
                                           (COND
                                              ((EQ 0 (SETQ DASHCNT (SUB1 DASHCNT)))
                                               (SETQ DASHON (NOT DASHON))
                                               (SETQ DASHTAIL (OR (LISTP (CDR DASHTAIL))
                                                                  DASHLST))
                                               (SETQ DASHCNT (CAR DASHTAIL]
                                [COND
                                   ([NOT (IGREATERP DY (SETQ CDL (IPLUS CDL DX]
                                    (COND
                                       ((IGREATERP (SETQ X1 (ADD1 X1))
                                                   X2)
                                        (RETURN)))
                                    (SETQ CDL (IDIFFERENCE CDL DY]
                                (add Y1 YINC]
           (T                                                       ; when we put the points down make it uninterruptable
```

```
                (.WHILE.TOP.DS. DISPLAYSTREAM (COND
                                                [(IGEQ DX DY)   ; X is the fastest mover.
                                                 (until (IGREATERP X1 X2)
                                                    do          ; main loop
                                                       (COND
                                                          (DASHON (\CURVEPT X1 Y1)))
                                                       [COND
                                                          (DASHTAIL
                                                                ; do dashing.
                                                                  (COND
                                                                     ((EQ 0 (SETQ DASHCNT (SUB1 DASHCNT)))
                                                                      (SETQ DASHON (NOT DASHON))
                                                                      (SETQ DASHTAIL (OR (LISTP (CDR DASHTAIL))
                                                                                         DASHLST))
                                                                      (SETQ DASHCNT (CAR DASHTAIL]
                                                       [COND
                                                          ([NOT (IGREATERP DX (SETQ CDL (IPLUS CDL DY]
                                                           (SETQ Y1 (IPLUS Y1 YINC))
                                                           (COND
                                                              ((COND
                                                                  ((EQ YINC -1)
                                                                   (ILESSP Y1 Y2))
                                                                  ((IGREATERP Y1 Y2)))
                                                               (RETURN)))
                                                           (SETQ CDL (IDIFFERENCE CDL DX]
                                                       (SETQ X1 (ADD1 X1]
                                                (T              ; Y is the fastest mover.
                                                   (until (COND
                                                             ((EQ YINC -1)
                                                              (ILESSP Y1 Y2))
                                                             ((IGREATERP Y1 Y2)))
                                                      do        ; main loop
                                                         (COND
                                                            (DASHON (\CURVEPT X1 Y1)))
                                                         [COND
                                                            (DASHTAIL
                                                                ; do dashing.
                                                                    (COND
                                                                       ((EQ 0 (SETQ DASHCNT (SUB1 DASHCNT)))
                                                                        (SETQ DASHON (NOT DASHON))
                                                                        (SETQ DASHTAIL
                                                                         (OR (LISTP (CDR DASHTAIL))
                                                                             DASHLST))
                                                                        (SETQ DASHCNT (CAR DASHTAIL]
                                                         [COND
                                                            ([NOT (IGREATERP DY (SETQ CDL (IPLUS CDL DX]
                                                             (COND
                                                                ((IGREATERP (SETQ X1 (ADD1 X1))
                                                                            X2)
                                                                 (RETURN)))
                                                              (SETQ CDL (IDIFFERENCE CDL DY]
                                                         (SETQ Y1 (IPLUS Y1 YINC]
                        (RETURN NIL])

)

(DEFINEQ

(LOADPOLY
  [LAMBDA (POLY POLYPRIME A B C D)                          (* hdj "13-Mar-85 18:01")
    (replace (POLYNOMIAL A) of POLY with (FQUOTIENT A 6.0))
    (replace (POLYNOMIAL B) of POLY with (FQUOTIENT B 2.0))
    (replace (POLYNOMIAL C) of POLY with C)
    (replace (POLYNOMIAL D) of POLY with D)
    (replace (POLYNOMIAL A) of POLYPRIME with (FQUOTIENT A 2.0))
    (replace (POLYNOMIAL B) of POLYPRIME with B)
    (replace (POLYNOMIAL C) of POLYPRIME with C])

(PARAMETRICSPLINE
  [LAMBDA (KNOTS CLOSEDFLG SPLINE)                          (* rmk%: "30-Nov-84 17:02")
    ;; KNOTS is a non-NIL list of knots, CLOSEDFLG => closed curve
    (PROG (DX DY DDX DDY DDDX DDDY %#KNOTS A BX BY X Y SX SY A C R D2X D2Y I)
          [COND
             (CLOSEDFLG                                     ; Wrap around
                      (push KNOTS (CAR (LAST KNOTS]
          (SETQ %#KNOTS (LENGTH KNOTS))
          (SETQ DX (ARRAY %#KNOTS 0 0.0))
          (SETQ DDX (ARRAY %#KNOTS 0 0.0))
          (SETQ DDDX (ARRAY %#KNOTS 0 0.0))
          (SETQ DY (ARRAY %#KNOTS 0 0.0))
          (SETQ DDY (ARRAY %#KNOTS 0 0.0))
          (SETQ DDDY (ARRAY %#KNOTS 0 0.0))
          (SETQ X (ARRAY %#KNOTS 0 0.0))
          (SETQ Y (ARRAY %#KNOTS 0 0.0))
          (for KNOT in KNOTS as I from 1 to %#KNOTS do (OR (type? POSITION KNOT)
```

```
                                                    (ERROR "bad knot" KNOT))
                                                (SETA X I (CAR KNOT))
                                                (SETA Y I (CDR KNOT)))
        (SETQ A (ARRAY %#KNOTS 0 0.0))
        (SETQ BX (ARRAY %#KNOTS 0 0.0))
        (SETQ BY (ARRAY %#KNOTS 0 0.0))
        [COND
           (CLOSEDFLG (SETQ C (ARRAY %#KNOTS 0 0.0))
                  (SETQ R (ARRAY %#KNOTS 0 0.0))
                  (SETQ SX (ARRAY %#KNOTS 0 0.0))
                  (SETQ SY (ARRAY %#KNOTS 0 0.0]
        (SETA A 1 4.0)
        [for I from 2 to (IDIFFERENCE %#KNOTS 2) do (SETA A I (FDIFFERENCE 4.0 (FQUOTIENT 1.0
                                                                                    (ELT A (SUB1 I]
        [COND
           (CLOSEDFLG (SETA C 1 1.0)
                  (for I from 2 to (IDIFFERENCE %#KNOTS 2) do (SETA C I (FMINUS (FQUOTIENT (ELT C (SUB1 I))
                                                                                    (ELT A (SUB1 I]
        [COND
           ((IGEQ %#KNOTS 3)
            (COND
               [CLOSEDFLG [SETA BX 1 (FTIMES 6.0 (FPLUS (ELT X 2)
                                                        (FMINUS (FTIMES 2.0 (ELT X 1)))
                                                        (ELT X (SUB1 %#KNOTS]
                     [SETA BY 1 (FTIMES 6.0 (FPLUS (ELT Y 2)
                                                   (FMINUS (FTIMES 2.0 (ELT Y 1)))
                                                   (ELT Y (SUB1 %#KNOTS]
                     [for I from 2 to (IDIFFERENCE %#KNOTS 2)
                        do [SETA BX I (FDIFFERENCE [FTIMES 6.0 (FPLUS (ELT X (ADD1 I))
                                                                      (FMINUS (FTIMES 2.0 (ELT X I)))
                                                                      (ELT X (SUB1 I]
                                                   (FQUOTIENT (ELT BX (SUB1 I))
                                                          (ELT A (SUB1 I]
                           (SETA BY I (FDIFFERENCE [FTIMES 6.0 (FPLUS (ELT Y (ADD1 I))
                                                                      (FMINUS (FTIMES 2.0 (ELT Y I)))
                                                                      (ELT Y (SUB1 I]
                                                   (FQUOTIENT (ELT BY (SUB1 I))
                                                          (ELT A (SUB1 I]
                     (SETA R (SUB1 %#KNOTS)
                           1.0)
                     (SETA SX (SUB1 %#KNOTS)
                           0.0)
                     (SETA SY (SUB1 %#KNOTS)
                           0.0)
                     (for I from (IDIFFERENCE %#KNOTS 2) to 1 by -1
                        do [SETA R I (FMINUS (FQUOTIENT (FPLUS (ELT R (ADD1 I))
                                                               (ELT C I))
                                                        (ELT A I]
                           (SETA SX I (FQUOTIENT (FDIFFERENCE (ELT BX I)
                                                              (ELT SX (ADD1 I)))
                                                 (ELT A I)))
                           (SETA SY I (FQUOTIENT (FDIFFERENCE (ELT BY I)
                                                              (ELT SY (ADD1 I)))
                                                 (ELT A I]
               (T [SETA BX 1 (FTIMES 6.0 (FPLUS (FDIFFERENCE (ELT X 3)
                                                             (FTIMES 2.0 (ELT X 2)))
                                                 (ELT X 1]
                  [SETA BY 1 (FTIMES 6.0 (FPLUS (FDIFFERENCE (ELT Y 3)
                                                             (FTIMES 2.0 (ELT Y 2)))
                                                 (ELT Y 1]
                  (for I from 2 to (IDIFFERENCE %#KNOTS 2)
                     do [SETA BX I (FDIFFERENCE (FTIMES 6.0 (FPLUS [FDIFFERENCE (ELT X (IPLUS I 2))
                                                                               (FTIMES 2 (ELT X (ADD1 I]
                                                                   (ELT X I)))
                                                (FQUOTIENT (ELT BX (SUB1 I))
                                                       (ELT A (SUB1 I]
                        (SETA BY I (FDIFFERENCE (FTIMES 6.0 (FPLUS [FDIFFERENCE (ELT Y (IPLUS I 2))
                                                                               (FTIMES 2 (ELT Y (ADD1 I]
                                                                   (ELT Y I)))
                                                (FQUOTIENT (ELT BY (SUB1 I))
                                                       (ELT A (SUB1 I]
        [COND
           (CLOSEDFLG [SETQ D2X (FPLUS (ELT X %#KNOTS)
                                       [FMINUS (FTIMES 2.0 (ELT X (SUB1 %#KNOTS]
                                       (ELT X (IDIFFERENCE %#KNOTS 2]
                  [SETQ D2Y (FPLUS (ELT Y %#KNOTS)
                                   [FMINUS (FTIMES 2.0 (ELT Y (SUB1 %#KNOTS]
                                   (ELT Y (IDIFFERENCE %#KNOTS 2]
                  (SETA DDX (SUB1 %#KNOTS)
                        (FQUOTIENT (FDIFFERENCE (FDIFFERENCE (FTIMES D2X 6.0)
                                                             (ELT SX 1))
                                                (ELT SX (IDIFFERENCE %#KNOTS 2)))
                                   (FPLUS (ELT R 1)
                                          (ELT R (IDIFFERENCE %#KNOTS 2))
                                          4.0)))
                  (SETA DDY (SUB1 %#KNOTS)
                        (FQUOTIENT (FDIFFERENCE (FDIFFERENCE (FTIMES D2Y 6.0)
```

```
                                                    (ELT SY 1))
                                          (ELT SY (IDIFFERENCE %#KNOTS 2)))
                                 (FPLUS (ELT R 1)
                                        (ELT R (IDIFFERENCE %#KNOTS 2))
                                        4.0)))
                        [for I from 1 to (IDIFFERENCE %#KNOTS 2) do [SETA DDX I (FPLUS (ELT SX I)
                                                                                       (FTIMES (ELT R I)
                                                                                               (ELT DDX (SUB1 %#KNOTS]
                                                                    (SETA DDY I (FPLUS (ELT SY I)
                                                                                       (FTIMES (ELT R I)
                                                                                               (ELT DDY (SUB1 %#KNOTS]
                        (SETA DDX %#KNOTS (ELT DDX 1))
                        (SETA DDY %#KNOTS (ELT DDY 1)))
                  (T                                                    ; COMPUTE SECOND DERIVATIVES.
                     [SETA DDX 1 (SETA DDY 1 (SETA DDX %#KNOTS (SETA DDY %#KNOTS 0.0]
                     (for I from (SUB1 %#KNOTS) to 2 by -1 do [SETA DDX I (FQUOTIENT (FDIFFERENCE (ELT BX (SUB1 I))
                                                                                                  (ELT DDX (ADD1 I)))
                                                                                      (ELT A (SUB1 I]
                                                              (SETA DDY I (FQUOTIENT (FDIFFERENCE (ELT BY (SUB1 I))
                                                                                                  (ELT DDY (ADD1 I)))
                                                                                      (ELT A (SUB1 I]
            [for I from 1 to (SUB1 %#KNOTS) do                          ; COMPUTE 1ST & 3RD DERIVATIVES
                                          (SETA DX I (FDIFFERENCE (FDIFFERENCE (ELT X (ADD1 I))
                                                                               (ELT X I))
                                                                  (FQUOTIENT (FPLUS (FTIMES 2 (ELT DDX I))
                                                                                    (ELT DDX (ADD1 I)))
                                                                             6.0)))
                                          (SETA DY I (FDIFFERENCE (FDIFFERENCE (ELT Y (ADD1 I))
                                                                               (ELT Y I))
                                                                  (FQUOTIENT (FPLUS (FTIMES 2 (ELT DDY I))
                                                                                    (ELT DDY (ADD1 I)))
                                                                             6.0)))
                                          (SETA DDDX I (FDIFFERENCE (ELT DDX (ADD1 I))
                                                                    (ELT DDX I)))
                                          (SETA DDDY I (FDIFFERENCE (ELT DDY (ADD1 I))
                                                                    (ELT DDY I]
            (SETQ SPLINE
             (create SPLINE
                    %#KNOTS _ %#KNOTS
                    SPLINEX _ X
                    SPLINEY _ Y
                    SPLINEDX _ DX
                    SPLINEDY _ DY
                    SPLINEDDX _ DDX
                    SPLINEDDY _ DDY
                    SPLINEDDDX _ DDDX
                    SPLINEDDDY _ DDDY))
            (RETURN SPLINE])
```

(\CURVE

```
[LAMBDA (X0 Y0 X1 Y1 DX DY DDX DDY DDDX DDDY N BRUSHBM DISPLAYDATA BBT ENDING USERFN DISPLAYSTREAM)
                                                                        (* rrb "30-Apr-85 12:44")
   (DECLARE (LOCALVARS . T))
   ;; Puts a spline segment down.  Since it calls BitBlt1 directly, it must clip to both clipping region and the size of the destination bit map.
   (PROG (OLDX X Y OLDY DELTAX DELTAY DELTA TX TY OOLDX OOLDY)
         [COND
            ((NEQ N 0)
             [COND
                (USERFN                                                 ; if there is a user fn, stay in his coordinates.
                       (SETQ OLDX X0)
                       (SETQ OLDY Y0))
                (T ;; SUB1 on brush size is to cause the extra bit to be in the top left direction as is documented for lines.
                   (SETQ OLDX (\DSPTRANSFORMX (IDIFFERENCE X0 (LRSH (SUB1 BRUSHWIDTH)
                                                                    1))
                                              DISPLAYDATA))
                   (SETQ OLDY (\DSPTRANSFORMY (IDIFFERENCE Y0 (LRSH (SUB1 BRUSHHEIGHT)
                                                                    1))
                                              DISPLAYDATA]              ; draw origin point
             (\CURVESMOOTH OLDX OLDY USERFN DISPLAYSTREAM)              ; convert the derivatives to fractional representation.
             ;; \CONVERTTOFRACTION always returns a large number box.  This uses 0.49 because 0.5 causes rounding up.
             (SETQ X (\CONVERTTOFRACTION (FPLUS OLDX 0.49)))
             (SETQ Y (\CONVERTTOFRACTION (FPLUS OLDY 0.49)))
             (SETQ DX (\CONVERTTOFRACTION DX))
             (SETQ DY (\CONVERTTOFRACTION DY))
             (SETQ DDX (\CONVERTTOFRACTION DDX))
             (SETQ DDY (\CONVERTTOFRACTION DDY))
             (SETQ DDDX (\CONVERTTOFRACTION DDDX))
             (SETQ DDDY (\CONVERTTOFRACTION DDDY))
             [for I from 1 to N do                                      ; uses \BOXIPLUS to save box and also set the new value of the
                                                                        ; variable.
                           (\BOXIPLUS X DX)
                           (\BOXIPLUS DX DDX)
                           (\BOXIPLUS DDX DDDX)
```

```
                                (\BOXIPLUS Y DY)
                                (\BOXIPLUS DY DDY)
                                (\BOXIPLUS DDY DDDY)
                                (SETQ OOLDX OLDX)
                                (SETQ OOLDY OLDY)
                                (SETQ DELTAX (IDIFFERENCE (SETQ OLDX (\GETINTEGERPART X))
                                                          OOLDX))
                                (SETQ DELTAY (IDIFFERENCE (SETQ OLDY (\GETINTEGERPART Y))
                                                          OOLDY))
                                (SETQ DELTA (IMAX (IABS DELTAX)
                                                  (IABS DELTAY)))
                                (COND
                                   ((EQ DELTA 1)
                                    (\CURVESMOOTH OLDX OLDY USERFN DISPLAYSTREAM)))
                                (COND
                                   ((IGREATERP DELTA 1)
                                    (SETQ DELTAX (\CONVERTTOFRACTION (FQUOTIENT DELTAX DELTA)))
                                    (SETQ DELTAY (\CONVERTTOFRACTION (FQUOTIENT DELTAY DELTA)))
                                    (SETQ TX (\CONVERTTOFRACTION OOLDX))
                                    (SETQ TY (\CONVERTTOFRACTION OOLDY))
                                    (for I from 0 to DELTA do (\CURVESMOOTH (\GETINTEGERPART TX)
                                                                            (\GETINTEGERPART TY)
                                                                            USERFN DISPLAYSTREAM)
                                                              (\BOXIPLUS TX DELTAX)
                                                              (\BOXIPLUS TY DELTAY]
                                                               ; draw the end point
            (COND
               (USERFN (\CURVESMOOTH X1 Y1 USERFN DISPLAYSTREAM))
               (T (\CURVESMOOTH (\DSPTRANSFORMX (IDIFFERENCE X1 (LRSH (SUB1 BRUSHWIDTH)
                                                                      1))
                                                DISPLAYDATA)
                                (\DSPTRANSFORMY (IDIFFERENCE Y1 (LRSH (SUB1 BRUSHHEIGHT)
                                                                      1))
                                                DISPLAYDATA)
                                NIL DISPLAYSTREAM)))
            (AND DISPLAYSTREAM (MOVETO X1 Y1 DISPLAYSTREAM]
         (COND
            (ENDING (\CURVESMOOTH (IPLUS \CURX \CURX (IMINUS \OLDX))
                                  (IPLUS \CURY \CURY (IMINUS \OLDY))
                                  USERFN DISPLAYSTREAM)
                    (\CURVESMOOTH (IPLUS \CURX \CURX (IMINUS \OLDX))
                                  (IPLUS \CURY \CURY (IMINUS \OLDY))
                                  USERFN DISPLAYSTREAM)))
         (RETURN NIL])
```

(\CURVE2

```
  [LAMBDA (SPLINE BRUSH DASHLST BBT DISPLAYSTREAM)              (* jds "26-Nov-85 12:21")
```

;;; Given a spline curve, represented as a set of derivatives for each segment, draw it on DISPLAYSTREAM using the brush BRUSH, and dashing it
;;; according to DASHLST. For speed, use the bitblt table BBT.

```
    (DECLARE (SPECVARS . T))
```

;; DISPLAYSTREAM is guaranteed to be a display-stream. Should declare most of these variables local but currently have the \CURVE function
;; between here and \CURVEBBT so can't

```
    (PROG (BRUSHBM DestinationBitMap OPERATION BRUSHWIDTH BRUSHHEIGHT BRUSHBASE BRUSHRASTERWIDTH LEFT RIGHTPLUS1
                TOP BOTTOM DESTINATIONBASE LEFTMINUSBRUSH BOTTOMMINUSBRUSH TOPMINUSBRUSH RASTERWIDTH
                NBITSRIGHTPLUS1 HEIGHTMINUS1 COLOR COLORBRUSHBASE NBITS \CURX \CURY \OLDX \OLDY \OLDERX \OLDERY
                LKNOT (DASHON T)
                (DASHTAIL DASHLST)
                (DASHCNT (CAR DASHLST))
                NPOINTS NSEGS POINTSPERSEG DX D2X D3X DY D2Y D3Y D1 D2 D3 X0 Y0 X1 Y1 DX DDX DDDX DY DDY DDDY
                (XPOLY (create POLYNOMIAL))
                (X/PRIME/POLY (create POLYNOMIAL))
                (YPOLY (create POLYNOMIAL))
                (Y/PRIME/POLY (create POLYNOMIAL))
                (DISPLAYDATA (fetch IMAGEDATA of DISPLAYSTREAM))
                (USERFN (AND (LITATOM BRUSH)
                             BRUSH)))
```

;; many of these variables are used by the macro for \CURVEPT that passes them to \BBTCURVEPT and .SETUP.FOR.\BBTCURVEPT. sets
;; them up.

```
          [COND
             (USERFN                                            ; if calling user fn, don't bother with set up and leave points in
                                                                ; window coordinates.
                     (\CURVESTART (ELT (fetch (SPLINE SPLINEX) of SPLINE)
                                       1)
                                  (ELT (fetch (SPLINE SPLINEY) of SPLINE)
                                       1)))
             (T (.SETUP.FOR.\BBTCURVEPT.)                       ; Do it interruptably here to get set up, then uninterruptably when
                                                                ; drawing points
                (\INSURETOPWDS DISPLAYSTREAM)                   ; curve pts will be kept in screen coordinates, start smoothing
                                                                ; values there.
                (\CURVESTART (\DSPTRANSFORMX (IDIFFERENCE (ELT (fetch (SPLINE SPLINEX) of SPLINE)
                                                               1)
                                                          (LRSH (SUB1 BRUSHWIDTH)
```

```
                                                    1))
                                DISPLAYDATA)
                        (\DSPTRANSFORMY (IDIFFERENCE (ELT (fetch (SPLINE SPLINEY) of SPLINE)
                                                          1)
                                                     (LRSH (SUB1 BRUSHHEIGHT)
                                                           1))
                                DISPLAYDATA]
        [bind PERSEG for KNOT from 1 to (SUB1 (fetch %#KNOTS of SPLINE))
           when (PROGN ;; Loop thru the segments of the spline curve, drawing each in turn.
                       (SETQ X0 (ELT (fetch (SPLINE SPLINEX) of SPLINE)
                                     KNOT))                    ; Set up X0,Y0 -- the starting point of this segment
                       (SETQ Y0 (ELT (fetch (SPLINE SPLINEY) of SPLINE)
                                     KNOT))
                       (SETQ X1 (ELT (fetch (SPLINE SPLINEX) of SPLINE)
                                     (ADD1 KNOT)))             ; And X1,Y1 -- the ending point
                       (SETQ Y1 (ELT (fetch (SPLINE SPLINEY) of SPLINE)
                                     (ADD1 KNOT)))
                       (SETQ DX (ELT (fetch (SPLINE SPLINEDX) of SPLINE)
                                     KNOT))                    ; And the initial derivatives -- first
                       (SETQ DY (ELT (fetch (SPLINE SPLINEDY) of SPLINE)
                                     KNOT))
                       (SETQ DDX (ELT (fetch SPLINEDDX of SPLINE)
                                      KNOT))                   ; Second
                       (SETQ DDY (ELT (fetch SPLINEDDY of SPLINE)
                                      KNOT))
                       (SETQ DDDX (ELT (fetch SPLINEDDDX of SPLINE)
                                       KNOT))                  ; And third.
                       (SETQ DDDY (ELT (fetch SPLINEDDDY of SPLINE)
                                       KNOT))
                       (SETQ NPOINTS (FOLDLO (ITIMES (IMAX (IABS (IDIFFERENCE X1 X0))
                                                           (IABS (IDIFFERENCE Y1 Y0)))
                                                     3)
                                             2))
                       ;; Establish an upper bound on the number of points we'll draw while painting this segment.  We know that 3/2 the
                       ;; maximum DX or DY is the right amount.
                       (NOT (ZEROP NPOINTS)))
           do ;; NPOINTS can be zero if a knot is duplicated in the spline curve to produce a discontinuity.  Skip over zero-length segments to
              ;; avoid divide-by-zero trouble

              ;; To prevent round-off errors from accumulating, we'll draw this segment as runs of no more than 64 points each -- recomputing
              ;; completely at the start of each run.  This is a trade off of speed and accuracy.

              [COND
                 ((ILEQ NPOINTS 64)                          ; Fewer than 64 points to draw.  Do it in one run.
                  (SETQ NSEGS 1)
                  (SETQ POINTSPERSEG NPOINTS))
                 (T                                          ; Figure out how many runs to do it in.
                    (SETQ NSEGS (FOLDLO NPOINTS 64))
                    (SETQ POINTSPERSEG 64)
                    (SETQ NPOINTS (UNFOLD NSEGS 64]
              (SETQ D1 (FQUOTIENT 1.0 NPOINTS))              ; Set up Δt, Δt**2 and Δt**3, for computing the next point.
              (SETQ D2 (FTIMES D1 D1))
              (SETQ D3 (FTIMES D2 D1))
              (SETQ D3X (FTIMES D3 DDDX))
              (SETQ D3Y (FTIMES D3 DDDY))
              (COND
                 [(EQ NSEGS 1)                               ; Just one segment to draw.
                  [SETQ DX (FPLUS (FTIMES D1 DX)
                                  (FTIMES DDX D2 0.5)
                                  (FTIMES DDDX D3 (CONSTANT (FQUOTIENT 1.0 6.0]
                  (SETQ D2X (FPLUS (FTIMES D2 DDX)
                                   (FTIMES D3 DDDX)))
                  [SETQ DY (FPLUS (FTIMES D1 DY)
                                  (FTIMES D2 DDY 0.5)
                                  (FTIMES D3 DDDY (CONSTANT (FQUOTIENT 1.0 6.0]
                  (SETQ D2Y (FPLUS (FTIMES D2 DDY)
                                   (FTIMES D3 DDDY)))
                  (COND
                     (USERFN                                 ; Draw this run of points, using the user's supplied function.
                            (\CURVE X0 Y0 X1 Y1 DX DY D2X D2Y D3X D3Y NPOINTS BRUSHBM DISPLAYDATA BBT NIL
                                    USERFN DISPLAYSTREAM))
                     (T                                      ; Draw this run of points, using the brush.
                        (.WHILE.TOP.DS. DISPLAYSTREAM
                                (\CURVE X0 Y0 X1 Y1 DX DY D2X D2Y D3X D3Y NPOINTS BRUSHBM DISPLAYDATA BBT NIL
                                        NIL DISPLAYSTREAM]
                 (T                                          ; Have to do this segment in several runs.
                    (SETQ PERSEG (FQUOTIENT 1.0 NSEGS))
                    (LOADPOLY XPOLY X/PRIME/POLY DDDX DDX DX X0)
                    (LOADPOLY YPOLY Y/PRIME/POLY DDDY DDY DY Y0)
                    (bind (TT _ 0.0)
                          (DDDX/PER/SEG _ (FTIMES DDDX PERSEG))
                          (DDDY/PER/SEG _ (FTIMES DDDY PERSEG))
                          [D3XFACTOR _ (FTIMES D3 DDDX (CONSTANT (FQUOTIENT 1.0 6.0]
                          [D3YFACTOR _ (FTIMES D3 DDDY (CONSTANT (FQUOTIENT 1.0 6.0] for I from 0
                       to (SUB1 NSEGS)
                       do
```

```
                    ;; TT is the parameter, and runs from 0 to 1 as the curve segment runs from beginning to end.
                    (SETQ TT (FPLUS TT PERSEG))
                    (SETQ X1 (POLYEVAL TT XPOLY 3))
                    (SETQ Y1 (POLYEVAL TT YPOLY 3))
                    (SETQ DX (FPLUS (FTIMES D1 DX)
                                    (FTIMES D2 DDX 0.5)
                                    D3XFACTOR))
                    (SETQ D2X (FPLUS (FTIMES D2 DDX)
                                     (FTIMES D3 DDDX)))
                    (SETQ DY (FPLUS (FTIMES D1 DY)
                                    (FTIMES D2 DDY 0.5)
                                    D3YFACTOR))
                    (SETQ D2Y (FPLUS (FTIMES D2 DDY)
                                     (FTIMES D3 DDDY)))
                    [COND
                       (USERFN (\CURVE X0 Y0 X1 Y1 DX DY D2X D2Y D3X D3Y 64 BRUSHBM DISPLAYDATA BBT NIL
                                       USERFN DISPLAYSTREAM))
                       (T (.WHILE.TOP.DS. DISPLAYSTREAM
                                 (\CURVE X0 Y0 X1 Y1 DX DY D2X D2Y D3X D3Y 64 BRUSHBM DISPLAYDATA BBT
                                         NIL NIL DISPLAYSTREAM]
                    (SETQ X0 X1)
                    (SETQ Y0 Y1)
                    (SETQ DDX (FPLUS DDX DDDX/PER/SEG))
                    (SETQ DDY (FPLUS DDY DDDY/PER/SEG))
                    (SETQ DX (POLYEVAL TT X/PRIME/POLY 2))
                    (SETQ DY (POLYEVAL TT Y/PRIME/POLY 2]
```

;; Draw the final point on the curve.

```
(COND
   (USERFN (\CURVE 0 0 0 0 0 0 0 0 0 0 0 BRUSHBM DISPLAYDATA BBT T USERFN DISPLAYSTREAM))
   (T (.WHILE.TOP.DS. DISPLAYSTREAM
             (\CURVE 0 0 0 0 0 0 0 0 0 0 0 BRUSHBM DISPLAYDATA BBT T NIL DISPLAYSTREAM])
```

(\CURVEEND

```
[LAMBDA NIL                                                    (* rrb " 5-JAN-82 17:24")
```

;; Put out the last two points, using \CURVEPT, since they were held back for smoothing.

```
(PROG ((X \CURX)
       (Y \CURY)
       (DX (IDIFFERENCE \CURX \OLDX))
       (DY (IDIFFERENCE \CURY \OLDY)))
      (for I from 1 to 2 do (\CURVESMOOTH (SETQ X (IPLUS X DX))
                                          (SETQ Y (IPLUS Y DY])
```

(\CURVESLOPE

```
[LAMBDA (KNOTS ENDFLG)                                         (* rrb "30-Nov-84 18:17")
```

;; returns a CONS of DX DY that gives the slope of the curve thru KNOTS. If ENDFLG is NIL, it is at the beginning. If ENDFLG is T, it is at the last ;; point.

```
(PROG (DX DY PARAMS (%#KNOTS (LENGTH KNOTS)))
      (RETURN (SELECTQ %#KNOTS
                  ((0 1)                                       ; define slope as horizontal
                      '(1 . 0))
                  (2 [CONS (DIFFERENCE (fetch (POSITION XCOORD) of (CADR KNOTS))
                                       (fetch (POSITION XCOORD) of (CAR KNOTS)))
                           (DIFFERENCE (fetch (POSITION YCOORD) of (CADR KNOTS))
                                       (fetch (POSITION YCOORD) of (CAR KNOTS])
                  (PROGN [SETQ PARAMS (COND
                                        [ENDFLG (PARAMETRICSPLINE (REVERSE (NLEFT KNOTS (IMIN %#KNOTS 4]
                                        (T (PARAMETRICSPLINE (COND
                                                               ((EQ %#KNOTS 3)
                                                                (LIST (CAR KNOTS)
                                                                      (CADR KNOTS)
                                                                      (CADDR KNOTS)))
                                                               (T (LIST (CAR KNOTS)
                                                                        (CADR KNOTS)
                                                                        (CADDR KNOTS)
                                                                        (CADDDR KNOTS]
                         (SETQ DX (ELT (fetch (SPLINE SPLINEDX) of PARAMS)
                                       1))
                         (SETQ DY (ELT (fetch (SPLINE SPLINEDY) of PARAMS)
                                       1))
                         (if ENDFLG
                             then (CONS (MINUS DX)
                                        (MINUS DY))
                           else (CONS DX DY])
```

(\CURVESTART

```
[LAMBDA (X Y)                                                  (* jds "27-OCT-81 15:48")
```

;; Set up the init vals for \OLDER* \OLD* \CUR*, for curve smoothing in \CURVEPT.

```
(SETQ \OLDERX X)
(SETQ \OLDX X)
(SETQ \CURX X)
```

```
    (SETQ \OLDERY Y)
    (SETQ \OLDY Y)
    (SETQ \CURY Y])
```

(\FDIFS/FROM/DERIVS

```
  [LAMBDA (DZ DDZ DDDZ RAD NSTEPS)                                    (* rrb "12-MAY-81 10:59")
```

;; the derivatives of the function, plus a scale factor (radius for drawing circles) See 'Spline Curve Techniques' , equations 2.18.

```
    (PROG (S SS SSS)
          (SETQ S (FQUOTIENT 1.0 NSTEPS))
          (SETQ SS (FTIMES S S))
          (SETQ SSS (FTIMES SS S))
          (SETQ S (FTIMES S DZ RAD))
          (SETQ SS (FTIMES SS DDZ RAD))
          (SETQ SSS (FTIMES SSS DDDZ RAD))
          (RETURN (LIST (FPLUS S (FQUOTIENT SS 2.0)
                                 (FQUOTIENT SSS 6.0))
                        (FPLUS SS SSS)
                        SSS])

)

(DECLARE%: DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

(ARRAYRECORD POLYNOMIAL (A B C D)
       (CREATE (ARRAY 4 'FLOATP))
       (SYSTEM))

(RECORD SPLINE (%#KNOTS SPLINEX SPLINEY SPLINEDX SPLINEDY SPLINEDDX SPLINEDDY SPLINEDDDX SPLINEDDDY))
)
)
```

;; END EXPORTED DEFINITIONS

```
(DECLARE%: DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

(PUTPROPS HALF MACRO ((X)
                      (LRSH X 1)))

(PUTPROPS \FILLCIRCLEBLT MACRO (OPENLAMBDA (CX CY X Y)          ; calls bitblt twice to fill in one line of the circle.
                                  (\LINEBLT FCBBT (IDIFFERENCE CX X)
                                         (IPLUS CY Y)
                                         (IPLUS CX X)
                                         DESTINATIONBASE RASTERWIDTH LEFT RIGHT BOTTOM TOP GRAYWIDTH GRAYHEIGHT
                                         GRAYBASE NBITS)
                                  (\LINEBLT FCBBT (IDIFFERENCE CX X)
                                         (IDIFFERENCE CY Y)
                                         (IPLUS CX X)
                                         DESTINATIONBASE RASTERWIDTH LEFT RIGHT BOTTOM TOP GRAYWIDTH GRAYHEIGHT
                                         GRAYBASE NBITS)))
)
```

;; END EXPORTED DEFINITIONS

```
(DECLARE%: EVAL@COMPILE

(PUTPROPS \CURVEPT MACRO [OPENLAMBDA (X Y)
                           (COND
                              ((OR (ILEQ X LEFTMINUSBRUSH)
                                   (IGEQ X RIGHTPLUS1)
                                   (ILEQ Y BOTTOMMINUSBRUSH)
                                   (IGEQ Y TOP))
                               NIL)
                              ((NULL BBT)
                               (\FBITMAPBIT DESTINATIONBASE X Y OPERATION HEIGHTMINUS1 RASTERWIDTH))
                              (T ;; This should have been done in .SETUP.FOR.\BBTCURVEPT., under \GETBRUSHBBT.
                                 ;; Its a bug here, because brushes can't use operation REPLACE.
                                 ;; (\SETPBTFUNCTION BBT (ffetch DDSOURCETYPE of DISPLAYDATA) OPERATION)
                                 (\BBTCURVEPT X Y BBT LEFT BRUSHWIDTH LEFTMINUSBRUSH RIGHTPLUS1
                                        NBITSRIGHTPLUS1 TOPMINUSBRUSH DestinationBitMap BRUSHHEIGHT
                                        BOTTOMMINUSBRUSH TOP BRUSHBASE DESTINATIONBASE RASTERWIDTH
                                        BRUSHRASTERWIDTH COLORBRUSHBASE NBITS DISPLAYDATA])

(PUTPROPS .SETUP.FOR.\BBTCURVEPT. MACRO [NIL (PROGN (SETQ BOTTOM (ffetch (\DISPLAYDATA DDClippingBottom)
```

```
                                                    of DISPLAYDATA))
                                    (SETQ TOP (ffetch (\DISPLAYDATA DDClippingTop) of DISPLAYDATA
                                                      ))
                                    (SETQ RIGHTPLUS1 (ffetch (\DISPLAYDATA DDClippingRight)
                                                        of DISPLAYDATA))
                                    (SETQ LEFT (ffetch (\DISPLAYDATA DDClippingLeft)
                                                  of DISPLAYDATA))
                                    (SETQ DestinationBitMap (ffetch (\DISPLAYDATA DDDestination)
                                                               of DISPLAYDATA))
                                    (SETQ OPERATION (OR OPERATION (ffetch (\DISPLAYDATA
                                                                           DDOPERATION)
                                                                     of DISPLAYDATA)))
                                    (SETQ NBITS (fetch (BITMAP BITMAPBITSPERPIXEL) of
                                                                                DestinationBitMap
                                                       ))
                                    [COND
                                       [(NOT (EQ NBITS 1))
                                        (SETQ BRUSHBM (\GETCOLORBRUSH BRUSH (MAXIMUMCOLOR NBITS
                                                                                          )
                                                                            NBITS))
                                        [SETQ COLOR (COND
                                                       [(AND (LISTP BRUSH)
                                                             (CAR (LISTP (CDDR BRUSH]
                                                       ((DSPCOLOR NIL DISPLAYSTREAM))
                                                       (T (MAXIMUMCOLOR NBITS]
                                        [COND
                                           ((EQ OPERATION 'ERASE)
                                            (SETQ COLOR (OPPOSITECOLOR COLOR NBITS]
                                        (SETQ COLORBRUSHBASE (fetch (BITMAP BITMAPBASE)
                                                                  of (\GETCOLORBRUSH BRUSH COLOR
                                                                                     NBITS]
                                       (T (SETQ BRUSHBM (\GETBRUSH BRUSH]
                                    (SETQ RASTERWIDTH (ffetch (BITMAP BITMAPRASTERWIDTH)
                                                         of DestinationBitMap))
                                    (SETQ DESTINATIONBASE (ffetch (BITMAP BITMAPBASE)
                                                             of DestinationBitMap))
                                    (SETQ BBT (\GETBRUSHBBT BRUSHBM DISPLAYDATA BBT))
                                    (SETQ BRUSHBASE (fetch (BITMAP BITMAPBASE) of BRUSHBM))
                                    (SETQ BRUSHRASTERWIDTH (ffetch (BITMAP BITMAPRASTERWIDTH)
                                                              of BRUSHBM))
                                    [COND
                                       ((NULL BBT)
                                        (SETQ HEIGHTMINUS1 (SUB1 (ffetch (BITMAP BITMAPHEIGHT)
                                                                    of DestinationBitMap)))
                                        (COND
                                           ((EQ (ffetch (\DISPLAYDATA DDOPERATION) of DISPLAYDATA
                                                        )
                                                'INVERT)
                                            (SETQ OPERATION 'INVERT]
                                    (SETQ BRUSHWIDTH (ffetch (BITMAP BITMAPWIDTH) of BRUSHBM))
                                    (SETQ BRUSHHEIGHT (ffetch (BITMAP BITMAPHEIGHT) of BRUSHBM))
                                    (SETQ LEFTMINUSBRUSH (IDIFFERENCE LEFT BRUSHWIDTH))
                                    (SETQ BOTTOMMINUSBRUSH (IDIFFERENCE BOTTOM BRUSHHEIGHT))
                                    (SETQ TOPMINUSBRUSH (IDIFFERENCE TOP BRUSHHEIGHT))
                                    (SETQ NBITSRIGHTPLUS1 (ITIMES RIGHTPLUS1 NBITS))
                                    (SETQ BRUSHWIDTH (ITIMES BRUSHWIDTH NBITS])

(PUTPROPS \CIRCLEPTS MACRO (OPENLAMBDA (CX CY X Y)
                             (\CURVEPT (IPLUS CX X)
                                    (IPLUS CY Y))
                             (\CURVEPT (IDIFFERENCE CX X)
                                    (IPLUS CY Y))
                             (\CURVEPT (IPLUS CX X)
                                    (IDIFFERENCE CY Y))
                             (\CURVEPT (IDIFFERENCE CX X)
                                    (IDIFFERENCE CY Y))))

(PUTPROPS \CURVESMOOTH MACRO (OPENLAMBDA (NEWX NEWY USERFN DISPLAYSTREAM)
                               (PROG [(DX (IABS (IDIFFERENCE NEWX \OLDX)))
                                      (DY (IABS (IDIFFERENCE NEWY \OLDY]
                                     (COND
                                        ((OR (IGREATERP DX 1)
                                             (IGREATERP DY 1))
                                         [COND
                                            ((NEQ [IPLUS (ADD1 (IDIFFERENCE \OLDX \OLDERX))
                                                         (ITIMES 3 (ADD1 (IDIFFERENCE \OLDY \OLDERY]
                                                  4)
                                             [COND
                                                (DASHON (COND
                                                           (USERFN (APPLY* USERFN \OLDX \OLDY DISPLAYSTREAM))
                                                           (T (.WHILE.TOP.DS. DISPLAYSTREAM (\CURVEPT \OLDX
                                                                                                     \OLDY]
                                             (COND
                                                (DASHTAIL (COND
                                                             ((EQ 0 (SETQ DASHCNT (SUB1 DASHCNT)))
                                                              (SETQ DASHON (NOT DASHON))
                                                              (SETQ DASHTAIL (OR (LISTP (CDR DASHTAIL))
```

```
                                                                    DASHLST))
                                                          (SETQ DASHCNT (CAR DASHTAIL]
                                              (SETQ \OLDERX \OLDX)
                                              (SETQ \OLDERY \OLDY)
                                              (SETQ \OLDX \CURX)
                                              (SETQ \OLDY \CURY)))
                                        (SETQ \CURX NEWX)
                                        (SETQ \CURY NEWY))))
)
)

(DEFINEQ
```

(\FILLCIRCLE.DISPLAY

```
  [LAMBDA (DISPLAYSTREAM CENTERX CENTERY RADIUS TEXTURE)          (* kbr%: "24-Jan-86 19:12")
    ;; Fill in area bounded by circle DRAWCIRCLE would draw.
    (COND
       ((OR (NOT (NUMBERP RADIUS))
            (ILESSP (SETQ RADIUS (FIXR RADIUS))
                    0))
        (\ILLEGAL.ARG RADIUS))
       (T (GLOBALRESOURCE \BRUSHBBT
                 (PROG (TOP BOTTOM RIGHT LEFT OPERATION DestinationBitMap DISPLAYDATA X Y D DESTINATIONBASE
                            RASTERWIDTH CX CY TEXTUREBM GRAYHEIGHT GRAYWIDTH GRAYBASE NBITS FCBBT)
                       (SETQ DISPLAYDATA (fetch (STREAM IMAGEDATA) of DISPLAYSTREAM))
                       (SETQ X 0)
                       (SETQ Y RADIUS)
                       (SETQ D (ITIMES 2 (IDIFFERENCE 1 RADIUS)))
                       (SETQ FCBBT \BRUSHBBT)
                       (SETQ LEFT (fetch (\DISPLAYDATA DDClippingLeft) of DISPLAYDATA))
                       (SETQ BOTTOM (fetch (\DISPLAYDATA DDClippingBottom) of DISPLAYDATA))
                       (SETQ TOP (SUB1 (fetch (\DISPLAYDATA DDClippingTop) of DISPLAYDATA)))
                       (SETQ RIGHT (SUB1 (fetch (\DISPLAYDATA DDClippingRight) of DISPLAYDATA)))
                       (SETQ OPERATION (fetch (\DISPLAYDATA DDOPERATION) of DISPLAYDATA))
                       (SETQ DestinationBitMap (fetch (\DISPLAYDATA DDDestination) of DISPLAYDATA))
                       (SETQ NBITS (fetch (BITMAP BITMAPBITSPERPIXEL) of DestinationBitMap))
                       [SETQ TEXTUREBM (COND
                                          ((BITMAPP TEXTURE))
                                          [(NOT (EQ NBITS 1))    ; color case, default texture differently
                                           (COND
                                              ((BITMAPP (COLORTEXTUREFROMCOLOR# (COLORNUMBERP
                                                                                   (OR TEXTURE (DSPCOLOR NIL
                                                                                                         DISPLAYSTREAM
                                                                                                          ))
                                                                                   NBITS T)
                                                                                NBITS)))
                                              [(AND (LISTP TEXTURE)
                                                    (BITMAPP (COLORTEXTUREFROMCOLOR# (COLORNUMBERP (CADR
                                                                                                     TEXTURE
                                                                                                     )
                                                                                                   NBITS)
                                                                                     NBITS]
                                              (T (\ILLEGAL.ARG TEXTURE]
                                          ((LISTP TEXTURE)       ; either a color or a list of (texture color)
                                           (INSURE.B&W.TEXTURE TEXTURE))
                                          [(AND (NULL TEXTURE)
                                                (BITMAPP (fetch (\DISPLAYDATA DDTexture) of DISPLAYDATA]
                                          ([OR (FIXP TEXTURE)
                                               (AND (NULL TEXTURE)
                                                    (SETQ TEXTURE (fetch (\DISPLAYDATA DDTexture) of DISPLAYDATA]
                                                                  ; create bitmap for the texture.  Could reuse a bitmap but for now
                                                                  ; this is good enough.
                                           (SETQ TEXTUREBM (BITMAPCREATE 16 4))
                                           (SETQ GRAYBASE (fetch (BITMAP BITMAPBASE) of TEXTUREBM))
                                           (\PUTBASE GRAYBASE 0 (\SFReplicate (LOGAND (LRSH TEXTURE 12)
                                                                                      15)))
                                           (\PUTBASE GRAYBASE 1 (\SFReplicate (LOGAND (LRSH TEXTURE 8)
                                                                                      15)))
                                           (\PUTBASE GRAYBASE 2 (\SFReplicate (LOGAND (LRSH TEXTURE 4)
                                                                                      15)))
                                           (\PUTBASE GRAYBASE 3 (\SFReplicate (LOGAND TEXTURE 15)))
                                           TEXTUREBM)
                                          (T (\ILLEGAL.ARG TEXTURE]
                       (SETQ GRAYBASE (fetch (BITMAP BITMAPBASE) of TEXTUREBM))
                       (SETQ DESTINATIONBASE (fetch (BITMAP BITMAPBASE) of DestinationBitMap))
                       (SETQ RASTERWIDTH (fetch (BITMAP BITMAPRASTERWIDTH) of DestinationBitMap))
                                                                  ; update as many fields in the brush bitblt table as possible from
                                                                  ; DS.
                       (replace (PILOTBBT PBTFLAGS) of FCBBT with 0)
                       (replace (PILOTBBT PBTDESTBPL) of FCBBT with (UNFOLD RASTERWIDTH BITSPERWORD))
                                                                  ; clear gray information.  PBTSOURCEBPL is used for gray
                                                                  ; information too.
                       (replace (PILOTBBT PBTSOURCEBPL) of FCBBT with 0)
                       (replace (PILOTBBT PBTUSEGRAY) of FCBBT with T)
                       [replace (PILOTBBT PBTGRAYWIDTHLESSONE) of FCBBT with (SUB1 (SETQ GRAYWIDTH
                                                                                    (IMIN (fetch (BITMAP
```

```
                                                                                    BITMAPWIDTH
                                                                                    )
                                                                              of TEXTUREBM)
                                                                          16]
          [replace (PILOTBBT PBTGRAYHEIGHTLESSONE) of FCBBT
             with (SUB1 (SETQ GRAYHEIGHT (IMIN (fetch (BITMAP BITMAPHEIGHT) of TEXTUREBM)
                                               16]
          (replace (PILOTBBT PBTDISJOINT) of FCBBT with T)
          (\SETPBTFUNCTION FCBBT 'TEXTURE OPERATION)
          (replace (PILOTBBT PBTHEIGHT) of FCBBT with 1)
                                                          ; take into account the brush thickness.
          (SETQ CX (\DSPTRANSFORMX CENTERX DISPLAYDATA))
          (SETQ CY (\DSPTRANSFORMY CENTERY DISPLAYDATA))
                                                          ; change Y TOP and BOTTOM to be in bitmap coordinates
          (SETQ CY (SUB1 (\SFInvert DestinationBitMap CY)))
          (SETQ TOP (SUB1 (\SFInvert DestinationBitMap TOP)))
          (SETQ BOTTOM (SUB1 (\SFInvert DestinationBitMap BOTTOM)))
          (swap TOP BOTTOM)
          (\INSURETOPWDS DISPLAYSTREAM)
    ;; Move the window to top while interruptable, but verify that it is still there uninterruptably with drawing points
          (COND
             ((EQ RADIUS 0)                               ; put a single point down.  Use \LINEBLT to get proper texture.
                                                          ; NIL
              (.WHILE.TOP.DS. DISPLAYSTREAM
                     (\LINEBLT FCBBT CX CY CX DESTINATIONBASE RASTERWIDTH LEFT RIGHT BOTTOM TOP
                            GRAYWIDTH GRAYHEIGHT GRAYBASE NBITS))
              (RETURN)))
     LP                                                   ; (UNFOLD x 2) is used instead of (ITIMES x 2)
          [COND
             [(IGREATERP 0 D)
              (SETQ X (ADD1 X))
              (COND
                 ((IGREATERP (UNFOLD (IPLUS D Y)
                                     2)
                             1)
                  (SETQ D (IPLUS D (UNFOLD (IDIFFERENCE X Y)
                                           2)
                                 4)))
                 (T (SETQ D (IPLUS D (UNFOLD X 2)
                                   1))                    ; don't draw unless Y changes.
                    (GO LP]
             ((OR (EQ 0 D)
                  (IGREATERP X D))
              (SETQ X (ADD1 X))
              (SETQ D (IPLUS D (UNFOLD (IDIFFERENCE X Y)
                                       2)
                             4)))
             (T (SETQ D (IPLUS (IDIFFERENCE D (UNFOLD Y 2))
                               3]
          (COND
             ((EQ Y 0)                                    ; draw the middle line differently to avoid duplication.
              (.WHILE.TOP.DS. DISPLAYSTREAM
                     (\LINEBLT FCBBT (IDIFFERENCE CX X)
                            CY
                            (IPLUS CX X)
                            DESTINATIONBASE RASTERWIDTH LEFT RIGHT BOTTOM TOP GRAYWIDTH GRAYHEIGHT
                            GRAYBASE NBITS)))
             (T (.WHILE.TOP.DS. DISPLAYSTREAM (\FILLCIRCLEBLT CX CY X Y))
                (SETQ Y (SUB1 Y))
                (GO LP)))
          (MOVETO CENTERX CENTERY DISPLAYSTREAM)
          (RETURN NIL])
```

(**LINEBLT**

```
[LAMBDA (BBT X Y XRIGHT DESTINATIONBASE RASTERWIDTH LEFT RIGHT BOTTOM TOP GRAYWIDTH GRAYHEIGHT GRAYBASE NBITS)
                                                          (* kbr%: "15-Feb-86 22:08")
```

;; fills in the changing fields of a bit blt tablt to draw one line of aan area.

```
(PROG NIL
      (COND
         ((ILESSP X LEFT)
          (SETQ X LEFT)))
      (COND
         ((IGREATERP XRIGHT RIGHT)
          (SETQ XRIGHT RIGHT)))
      (COND
         ((OR (IGREATERP X XRIGHT)
              (IGREATERP Y TOP)
              (IGREATERP BOTTOM Y))
          (RETURN)))
      (replace (PILOTBBT PBTDEST) of BBT with (\ADDBASE DESTINATIONBASE (ITIMES RASTERWIDTH Y)))
      [freplace (PILOTBBT PBTSOURCE) of BBT with (\ADDBASE GRAYBASE (freplace (PILOTBBT PBTGRAYOFFSET)
                                                                        of BBT with (MOD Y GRAYHEIGHT]
      (SELECTQ NBITS
          (1 (freplace (PILOTBBT PBTDESTBIT) of BBT with X)
```

```
                (freplace (PILOTBBT PBTSOURCEBIT) of BBT with (MOD X GRAYWIDTH))
                (freplace (PILOTBBT PBTWIDTH) of BBT with (ADD1 (IDIFFERENCE XRIGHT X))))
            (4                                                  ; color case, shift x values {which are in pixels} into bit values.
                (freplace (PILOTBBT PBTDESTBIT) of BBT with (SETQ X (LLSH X 2)))
                                                                ; if TEXTURE is not a multiple of nbits wide this is probably
                                                                ; garbage.
                (freplace (PILOTBBT PBTSOURCEBIT) of BBT with (MOD X GRAYWIDTH))
                (freplace (PILOTBBT PBTWIDTH) of BBT with (IDIFFERENCE (LLSH (ADD1 XRIGHT)
                                                                           2)
                                                                       X)))
            (8                                                  ; color case, shift x values {which are in pixels} into bit values.
                (freplace (PILOTBBT PBTDESTBIT) of BBT with (SETQ X (LLSH X 3)))
                (freplace (PILOTBBT PBTSOURCEBIT) of BBT with (MOD X GRAYWIDTH))
                (freplace (PILOTBBT PBTWIDTH) of BBT with (IDIFFERENCE (LLSH (ADD1 XRIGHT)
                                                                           3)
                                                                       X)))
            (24                                                 ; color case, shift x values {which are in pixels} into bit values.
                (freplace (PILOTBBT PBTDESTBIT) of BBT with (SETQ X (ITIMES 24 X)))
                (freplace (PILOTBBT PBTSOURCEBIT) of BBT with (MOD X GRAYWIDTH))
                (freplace (PILOTBBT PBTWIDTH) of BBT with (IDIFFERENCE (ITIMES 24 (ADD1 XRIGHT))
                                                                       X)))
            (SHOULDNT))
        (\PILOTBITBLT BBT 0])

)

;; making and copying bitmaps

(DEFINEQ

(SCREENBITMAP
  [LAMBDA (SCREEN)                                              ; Edited 20-Feb-87 14:57 by rrb
    ;; Return bitmap destination of SCREEN.
    (COND
       ((NULL SCREEN)
        ScreenBitMap)
       ((type? SCREEN SCREEN)
        (fetch (SCREEN SCDESTINATION) of SCREEN))
       ((WINDOWP SCREEN)
        (fetch (SCREEN SCDESTINATION) of (fetch (WINDOW SCREEN) of SCREEN)))
       (T (\ILLEGAL.ARG SCREEN])

(BITMAPP
  [LAMBDA (X)                                                   (* rrb "25-JUN-82 15:21")
                                                                ; is x a bitmap?
    (AND (type? BITMAP X)
         X])

(BITMAPHEIGHT
  [LAMBDA (BITMAP)                                              (* kbr%: " 8-Jul-85 16:01")
    ;; returns the height in pixels of a bitmap.
    (COND
       ((type? BITMAP BITMAP)
        (fetch (BITMAP BITMAPHEIGHT) of BITMAP))
       ((type? WINDOW BITMAP)
        (WINDOWPROP BITMAP 'HEIGHT))
       (T (\ILLEGAL.ARG BITMAP])

(BITSPERPIXEL
  [LAMBDA (BITMAP)                                              ; Edited 15-Feb-94 16:10 by nilsson
    ;; returns the height in pixels of a bitmap.
    (COND
       ((type? BITMAP BITMAP)
        (fetch (BITMAP BITMAPBITSPERPIXEL) of BITMAP))
       ((type? SCREEN BITMAP)
        ;; Read the propper slots, not the implicit bitmap.
        (OR (fetch (SCREEN SCDEPTH) of BITMAP)
            (fetch (SCREEN SCBITSPERPIXEL) of BITMAP)))
       ((type? WINDOW BITMAP)
        (BITSPERPIXEL (fetch (WINDOW SCREEN) of BITMAP)))
       ((ARRAYP BITMAP)                                         ; Consider array to be a colormap.
        (SELECTQ (ARRAYSIZE BITMAP)
             (256 8)
             (16 4)
             (LISPERROR "ILLEGAL ARG" BITMAP)))
       (T (LISPERROR "ILLEGAL ARG" BITMAP])

)
```

;; FOLLOWING DEFINITIONS EXPORTED

```
[PUTDEF 'BITMAPS 'FILEPKGCOMS '((COM MACRO (X (VARS . X]

[PUTDEF 'CURSORS 'FILEPKGCOMS '((COM MACRO (X (E (MAPC 'X 'PRINTCURSOR]
```

;; END EXPORTED DEFINITIONS

```
(DECLARE%: EVAL@COMPILE

;; FOLLOWING DEFINITIONS EXPORTED

(ADDTOVAR GLOBALVARS SCREENHEIGHT SCREENWIDTH ScreenBitMap)
)

;; END EXPORTED DEFINITIONS
```

;; Display stream functions that are not needed in the primitive system

```
(DEFINEQ

(DSPFILL
  [LAMBDA (REGION TEXTURE OPERATION STREAM)                    (* kbr%: " 8-Jul-85 15:40")
    ;; wipes a region of an imagestream with texture.
    ;; TEXTURE and OPERATION default to those of STREAM
    (PROG (STRM)
          (SETQ STRM (\OUTSTREAMARG STREAM))
          (OR REGION (SETQ REGION (DSPCLIPPINGREGION NIL STRM)))
          (RETURN (BLTSHADE TEXTURE STRM (fetch (REGION LEFT) of REGION)
                        (fetch (REGION BOTTOM) of REGION)
                        (fetch (REGION WIDTH) of REGION)
                        (fetch (REGION HEIGHT) of REGION)
                        OPERATION])

(INVERTW
  [LAMBDA (WIN SHADE)                                          (* rrb "18-May-84 21:52")
    ;; inverts a window and returns the window.  Used in RESETFORMS.
    (DSPFILL (DSPCLIPPINGREGION NIL WIN)
           (OR SHADE BLACKSHADE)
           'INVERT WIN)
    WIN])

)

(DEFINEQ

(\DSPCOLOR.DISPLAY
  [LAMBDA (STREAM COLOR)                                       ; Edited 29-Jan-91 11:33 by matsuda
    ;; sets and returns a display stream's background color.
    (PROG (DD COLORCELL DESTINATION BITSPERPIXEL)
          (SETQ DD (\GETDISPLAYDATA STREAM))
          (SETQ COLORCELL (fetch (\DISPLAYDATA DDCOLOR) of DD))
          (SETQ DESTINATION (fetch (\DISPLAYDATA DDDestination) of DD))
          (SETQ BITSPERPIXEL (BITSPERPIXEL DESTINATION))
          (RETURN (COND
                     (COLOR (SETQ COLOR (COLORNUMBERP COLOR BITSPERPIXEL))
                            (PROG1 (COND
                                      (COLORCELL (PROG1 (CAR COLORCELL)
                                                        (RPLACA COLORCELL COLOR)))
                                      (T                          ; no color cell yet, make one.
                                         (replace (\DISPLAYDATA DDCOLOR) of DD with (CONS COLOR 0))
                                         (MAXIMUMCOLOR BITSPERPIXEL)))
                                   (\SFFixFont STREAM DD)))
                     (T (OR (CAR COLORCELL)
                            (MAXIMUMCOLOR BITSPERPIXEL])

(\DSPBACKCOLOR.DISPLAY
  [LAMBDA (STREAM COLOR)                                       (* kbr%: "25-Aug-85 18:15")
    ;; sets and returns a display stream's foreground color.
    (PROG (DD COLORCELL DESTINATION BITSPERPIXEL)
          (SETQ DD (\GETDISPLAYDATA STREAM))
          (SETQ COLORCELL (fetch (\DISPLAYDATA DDCOLOR) of DD))
          (RETURN (COND
                     (COLOR (SETQ DESTINATION (fetch (\DISPLAYDATA DDDestination) of DD))
                            (SETQ BITSPERPIXEL (fetch (BITMAP BITMAPBITSPERPIXEL) of DESTINATION))
                            (SETQ COLOR (COLORNUMBERP COLOR BITSPERPIXEL))
                            (PROG1 (COND
                                      (COLORCELL (PROG1 (CDR COLORCELL)
                                                        (RPLACD COLORCELL COLOR)))
                                      (T                          ; no color cell yet, make one.
```

```
                                        (replace (\DISPLAYDATA DDCOLOR) of DD with (CONS (MAXIMUMCOLOR
                                                                                         BITSPERPIXEL)
                                                                                 COLOR))
                                        0))
                                  (\SFFixFont STREAM DD)))
                        (T (OR (CDR COLORCELL)
                               0])
```

(DSPEOLFN

```
  [LAMBDA (EOLFN DISPLAYSTREAM)                                    (* rrb "18-May-84 21:44")
```

;; sets the end of line function for a displaystream. EOLFN will be called every EOL with the argument of the display stream. If EOLFN is 'OFF, the ;; eolfn is cleared.

```
    (PROG ((DD (\GETDISPLAYDATA DISPLAYSTREAM)))
          (RETURN (PROG1 (COND
                              ((fetch (\DISPLAYDATA DDEOLFN) of DD))
                              (T 'OFF))
                         [AND EOLFN (COND
                                        [(LITATOM EOLFN)
                                         (replace (\DISPLAYDATA DDEOLFN) of DD with (COND
                                                                                        ((EQ EOLFN 'OFF)
                                                                                         NIL)
                                                                                        (T EOLFN]
                                        (T (\ILLEGAL.ARG EOLFN])])
```

)

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

(RPAQQ BLACKSHADE 65535)

(RPAQQ WHITESHADE 0)

(CONSTANTS (BLACKSHADE 65535)
       (WHITESHADE 0))
)

(RPAQQ GRAYSHADE 43605)

(ADDTOVAR GLOBALVARS GRAYSHADE)
```

;; END EXPORTED DEFINITIONS

```
(DECLARE%: EVAL@COMPILE

(PUTPROPS DSPRUBOUTCHAR MACRO ((DS CHAR X Y TTBL)
                               (\DSPMOVELR DS CHAR X Y TTBL NIL T)))
)

(DEFINEQ
```

(DSPCLEOL

```
  [LAMBDA (DISPLAYSTREAM XPOS YPOS HEIGHT)                         (* lmm " 3-May-84 10:31")
    (\CHECKCARET DISPLAYSTREAM)
    (PROG ((DD (\GETDISPLAYDATA DISPLAYSTREAM DISPLAYSTREAM)))
          (RETURN (BITBLT NIL NIL NIL DISPLAYSTREAM (OR (FIXP XPOS)
                                                        (SETQ XPOS (ffetch DDLeftMargin of DD)))
                          [OR (FIXP YPOS)
                              (IDIFFERENCE (ffetch DDYPOSITION of DD)
                                     (FONTPROP DISPLAYSTREAM 'DESCENT]
                          (IMAX 0 (IDIFFERENCE (ffetch DDRightMargin of DD)
                                       XPOS))
                          (OR (FIXP HEIGHT)
                              (IMINUS (ffetch DDLINEFEED of DD)))
                          'TEXTURE
                          'REPLACE])
```

(DSPRUBOUTCHAR

```
  [LAMBDA (STREAM CHAR X Y TTBL)                                   (* Pavel " 6-Oct-86 22:44")
    (if (DISPLAYSTREAMP CHAR)
        then ;; Some older code may use the CHAR argument first.
             (swap STREAM CHAR)
             (SETQ TTBL X)
             (SETQ X)
             (SETQ Y))
    (\GETDISPLAYDATA STREAM STREAM)
    (\DSPMOVELR STREAM CHAR X Y TTBL NIL T])
```

(\DSPMOVELR

```
  [LAMBDA (DS CHAR X Y TTBL RIGHTWARDSFLG ERASEFLG)                (* JonL " 7-May-84 02:47")
```

```
    ;; Moves the cursor 'leftwards' (or 'rightwards' if RIGHTWARDSFLG is non-null) over any main character and control or meta indicators.  Returns
    ;; NIL if the move can't be determined, such as trying to move left when already at the left margin.  Effaces (or 'Rubs out') any bits moved over if
    ;; ERASEFLG is non-null.
    ([LAMBDA (DD)
        ;; Must do the \GETDISPLAYDATA first, since it may reset DS when it coerces to a DISPLAYSTREAM
        (PROG [(WIDTH (\STREAMCHARWIDTH (COND
                                        ((CHARCODEP CHAR)
                                         CHAR)
                                        (T (CHARCODE M)))
                                DS TTBL))
               (DEFAULTPOS? (AND (NULL X)
                                 (NULL Y]
              (OR ERASEFLG DEFAULTPOS? (SHOULDNT))                ; CURSORLEFT and CURSORRIGHT commands aren't allowed
                                                                  ; to start from anywhere except current spot
         ;; Note that if CHAR is not specified and DS has a variable-pitch font, then the results may be somewhat random.  Smart terminal drivers thus
         ;; can work well only on fixed-pitch fonts.
              (COND
                 ((NULL WIDTH)
                  (RETURN))
                 ((EQ 0 WIDTH)                                    ; Ha, what an easy case
                  (RETURN T)))
              (OR (FIXP X)
                  (SETQ X (ffetch DDXPOSITION of DD)))
              (OR (FIXP Y)
                  (SETQ Y (ffetch DDYPOSITION of DD)))
              (COND
                 ([COND
                     (RIGHTWARDSFLG (IGREATERP (add X WIDTH)
                                               (ffetch DDRightMargin of DD)))
                     (T (ILESSP (add X (IMINUS WIDTH))
                                (ffetch DDLeftMargin of DD]          ; If we can't do the full backup, then return NIL to signal this fact
                  (RETURN)))
              (\CHECKCARET DS)                                    ; Take down the caret, if there is one, just in case we are moving
                                                                  ; over it.
              [COND
                 (ERASEFLG                                        ; And do the erasure if requested
                        ([LAMBDA (FONT)
                           (PROG ((YPRIME (IDIFFERENCE Y (FONTDESCENT FONT)))
                                  (HEIGHT (FONTHEIGHT FONT)))
                                 (COND
                                    ((NOT DEFAULTPOS?)
                                     (MOVETO X Y DS)              ; Backup over the bits, and 'wipe' them out.

                                     ))
                                 (BITBLT NIL 0 0 DS X YPRIME WIDTH HEIGHT 'TEXTURE 'REPLACE)
                                                                  ; wipe out some bits

                           ]
                         (ffetch DDFONT of DD]
              (DSPXPOSITION X DS)                                 ; Now do the move.
              (RETURN T]
       (\GETDISPLAYDATA DS DS])

)

;; for cursor

(RPAQQ \DefaultCursor )

(DEFINEQ

(\CURSOR.DEFPRINT
  [LAMBDA (CURSOR STREAM)                                         ; Edited 15-Sep-94 16:13 by sybalsky
    (COND
       (*PRINT-ARRAY* (PRIN1 "#,(LET(image) (CURSORCREATE (SETQ image '" STREAM)
              (PRIN4 (fetch (CURSOR CUIMAGE) of CURSOR)
                     STREAM)
              (PRIN1 ") " STREAM)
              (COND
                 ((EQ (fetch (CURSOR CUIMAGE) of CURSOR)
                      (fetch (CURSOR CUMASK) of CURSOR))
                  (PRIN1 " image " STREAM))
                 (T (PRIN1 " '" STREAM)
                    (PRIN4 (fetch (CURSOR CUMASK) of CURSOR)
                           STREAM)))
              (PRIN1 " " STREAM)
              (PRIN1 (fetch (CURSOR CUHOTSPOTX) of CURSOR)
                     STREAM)
              (PRIN1 " " STREAM)
              (PRIN1 (fetch (CURSOR CUHOTSPOTY) of CURSOR)
                     STREAM)
              (PRIN1 " " STREAM)
              (PRIN1 (fetch (CURSOR CUDATA) of CURSOR)
```

```
                    STREAM)
              (PRIN1 "))" STREAM])

)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(RPAQ? DEFAULTCURSOR (CURSORCREATE \DefaultCursor NIL 0 15))

(COND
   ((NULL \CURRENTCURSOR)
    (SETQ \CURRENTCURSOR DEFAULTCURSOR)))

(DEFPRINT 'CURSOR '\CURSOR.DEFPRINT)
)

(DECLARE%: DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS DEFAULTCURSOR)
)
)
```

;; stuff to interpret colors as textures which is needed even in system that don't have color.

```
(DEFINEQ
```

(**TEXTUREOFCOLOR**

```
  [LAMBDA (COLOR NOERRORFLG)                                    (* rrb "30-Oct-85 19:43")
```

;; returns a texture to represent a color on a black and white display

```
    (PROG ((RGB (INSURE.RGB.COLOR COLOR NOERRORFLG)))
          (RETURN (COND
                      ((NULL RGB)
                       NIL)
                      ((AND (IGREATERP (fetch (RGB RED) of RGB)
                                       245)
                            (IGREATERP (fetch (RGB GREEN) of RGB)
                                       245)
                            (IGREATERP (fetch (RGB BLUE) of RGB)
                                       245))                    ; special case white
                       BLACKSHADE16)
                      (T (PROG [(TEX (\PRIMARYTEXTURE 'RED (fetch (RGB RED) of RGB]
                               (BITBLT NIL NIL NIL TEX 0 0 16 16 'TEXTURE 'PAINT (\PRIMARYTEXTURE
                                                                                  'BLUE
                                                                                  (fetch (RGB BLUE) of RGB)))
                               (BITBLT NIL NIL NIL TEX 0 0 16 16 'TEXTURE 'PAINT (\PRIMARYTEXTURE
                                                                                  'GREEN
                                                                                  (fetch (RGB GREEN) of RGB)))
                               (RETURN TEX])
```

(**\PRIMARYTEXTURE**

```
  [LAMBDA (PRIMARY LEVEL)                                       (* rrb "30-Oct-85 19:25")
```

;; returns the 16x16 texture for a primary color level.

```
    (PROG [(TEXTURE (BITMAPCOPY (SELECTQ PRIMARY
                                         (RED REDTEXTURE)
                                         (BLUE BLUETEXTURE)
                                         (GREEN GREENTEXTURE)
                                         (\ILLEGAL.ARG PRIMARY]
          (BITBLT (\LEVELTEXTURE LEVEL)
                 0 0 TEXTURE 0 0 16 16 'INPUT 'ERASE)
          (RETURN TEXTURE])
```

(**\LEVELTEXTURE**

```
  [LAMBDA (LEVEL)                                               (* rrb "20-Aug-85 16:42")
```

;; returns a 16x16 texture which is merged so that only light bits on both go to light with a primary color pattern to get a level primary pattern.

```
    (COND
       ((ILESSP LEVEL 100)
        BLACKSHADE16)
       ((ILESSP LEVEL 150)
        DARKGRAY16)
       ((ILESSP LEVEL 200)
        MEDIUMGRAY16)
       ((ILESSP LEVEL 245)
        LIGHTGRAY16)
       (T WHITESHADE16])
```

(**INSURE.B&W.TEXTURE**

```
  [LAMBDA (TEXTURE NOERRORFLG)                                  (* rrb "30-Oct-85 19:47")
```

;; coerces a TEXTURE argument to a 1 bit per pixel bitmap or small number

```
    (SELECTQ (TYPENAME TEXTURE)
        (LITATOM                                                    ; includes NIL case
                 (COND
                    (TEXTURE                                        ; should be a color name
                           (TEXTUREOFCOLOR (INSURE.RGB.COLOR TEXTURE NOERRORFLG)))
                    (T WHITESHADE)))
        ((SMALLP FIXP)
            (LOGAND TEXTURE BLACKSHADE))
        (BITMAP TEXTURE)
        (LISTP                                                      ; can be a list of (TEXTURE COLOR) or a list of levels rgb or hls.
               (COND
                  ((TEXTUREOFCOLOR TEXTURE T))
                  ((CAR TEXTURE)
                   (INSURE.B&W.TEXTURE (CAR TEXTURE)
                          NOERRORFLG))
                  ((CAR (LISTP (CDR TEXTURE)))
                   (TEXTUREOFCOLOR (CADR TEXTURE)
                          NOERRORFLG))
                  (T                                                ; list of form (NIL NIL)
                     WHITESHADE)))
        (COND
           ((NULL NOERRORFLG)
            (\ILLEGAL.ARG TEXTURE])
```

(INSURE.RGB.COLOR

```
  [LAMBDA (COLOR NOERRFLG)                                          (* rrb "30-Oct-85 19:34")
                                                                    ; returns the RGB triple for a color.
    (PROG (LEVELS)
          (RETURN (COND
                     [(FIXP COLOR)                                  ; don't know what to do with color numbers so error
                      (COND
                         (NOERRFLG NIL)
                         (T (\ILLEGAL.ARG COLOR]
                     [(LITATOM COLOR)
                      (COND
                         ((SETQ LEVELS (\LOOKUPCOLORNAME COLOR))
                                                                    ; recursively look up color number
                          (INSURE.RGB.COLOR (CDR LEVELS)
                                 NOERRFLG))
                         (NOERRFLG NIL)
                         (T (ERROR "Unknown color name" COLOR]
                     ((HLSP COLOR)                                  ; HLS form convert to RGB
                      (HLSTORGB COLOR))
                     ((RGBP COLOR)                                  ; check for RGB or HLS
                      COLOR)
                     (NOERRFLG NIL)
                     (T (\ILLEGAL.ARG COLOR])
```

(\LOOKUPCOLORNAME

```
  [LAMBDA (COLORNAME)                                               (* rrb "13-DEC-82 13:14")
    ;; looks up a prospective color name.  Returns a list whose CAR is the name and whose CDR is a color spec.
    (FASSOC COLORNAME COLORNAMES])
```

(RGBP

```
  [LAMBDA (X)                                                       (* rrb "27-OCT-82 10:15")
                                                                    ; return X if it is a red green blue triple.
    (PROG (TMP)
          (RETURN (AND (LISTP X)
                       (SMALLP (SETQ TMP (CAR X)))
                       (IGREATERP TMP -1)
                       (IGREATERP 256 TMP)
                       (SMALLP (SETQ TMP (CADR X)))
                       (IGREATERP TMP -1)
                       (IGREATERP 256 TMP)
                       (SMALLP (SETQ TMP (CADDR X)))
                       (IGREATERP TMP -1)
                       (IGREATERP 256 TMP)
                      X])
```

(HLSP

```
  [LAMBDA (X)                                                       (* rrb "31-Oct-85 10:51")
    ;; return T if X is a hue lightness saturation triple.
    (AND (NUMBERP (CAR (LISTP X)))
         (IGREATERP (CAR X)
                -1)
         (IGREATERP 361 (CAR X))
         [FLOATP (CAR (LISTP (CDR X]
         [FLOATP (CAR (LISTP (CDDR X]
         X])
```

(HLSTORGB

```
  [LAMBDA (HLS)                                                    (* rrb "30-Oct-85 19:59")
```

;; converts from a hue saturation lightness triple into red green blue triple. HUE is in range 0 to 360, lightness and saturation are in the range 0 to 1.0

;; this algorithm was taken from siggraph vol 13 number 3 August 1979: Status report on graphics standards planning committee.

```
    (PROG ((H (fetch (HLS HUE) of HLS))
           (L (fetch (HLS LIGHTNESS) of HLS))
           (S (fetch (HLS SATURATION) of HLS))
           Max Min)
          [SETQ Max (COND
                       ((FGREATERP 0.5 L)
                        (FTIMES L (FPLUS 1.0 S)))
                       (T (FDIFFERENCE (FPLUS L S)
                                 (FTIMES L S]
          (SETQ Min (FDIFFERENCE (FTIMES L 2)
                           Max))
          (RETURN (create RGB
                          RED _ (\HLSVALUEFN Min Max H)
                          GREEN _ (\HLSVALUEFN Min Max (IDIFFERENCE H 120))
                          BLUE _ (\HLSVALUEFN Min Max (IDIFFERENCE H 240])
```

(\HLSVALUEFN

```
  [LAMBDA (MIN MAX HUE)                                            (* rrb "25-OCT-82 10:47")
```

;; internal value function for converting from HLS to RGB.

```
    [COND
       ((ILESSP HUE 0)
        (SETQ HUE (IPLUS HUE 360]
    (FIX (FTIMES (COND
                    ((ILESSP HUE 60)
                     (FPLUS MIN (FQUOTIENT (FTIMES (FDIFFERENCE MAX MIN)
                                                   HUE)
                                           60)))
                    ((ILESSP HUE 180)
                     MAX)
                    ((ILESSP HUE 240)
                     (FPLUS MIN (FQUOTIENT (FTIMES (FDIFFERENCE MAX MIN)
                                                   (FDIFFERENCE 240 HUE))
                                           60)))
                    (T MIN))
                 255])

)

(RPAQQ COLORNAMES
       ((WHITE 255 255 255)
        (CYAN 0 255 255)
        (MAGENTA 255 0 255)
        (YELLOW 255 255 0)
        (RED 255 0 0)
        (GREEN 0 255 0)
        (BLUE 0 0 255)
        (BLACK 0 0 0)))

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS COLORNAMES)
)

(DECLARE%: DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS BLACKSHADE16 DARKGRAY16 MEDIUMGRAY16 LIGHTGRAY16 WHITESHADE16 REDTEXTURE GREENTEXTURE BLUETEXTURE)
)
)

(READVARS-FROM-STRINGS '(BLACKSHADE16 DARKGRAY16 MEDIUMGRAY16 LIGHTGRAY16 WHITESHADE16 REDTEXTURE GREENTEXTURE
                         BLUETEXTURE)
       "({(READBITMAP)(16 16
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
       %"OOOO%"
```

```
        %"OOOO%")}  {(READBITMAP)(16 16
        %"NMGG%"
        %"KGMM%"
        %"MNKK%"
        %"GKNN%"
        %"MNKK%"
        %"GKNM%"
        %"NMGN%"
        %"KGMG%"
        %"NKKM%"
        %"KNNK%"
        %"GGMN%"
        %"MMGG%"
        %"GGKM%"
        %"MJOG%"
        %"NOEK%"
        %"KMNN%")}  {(READBITMAP)(16 16
        %"JJJJ%"
        %"EEEE%"
        %"JJJJ%"
        %"EEEE%"
        %"JJJJ%"
        %"EEEE%"
        %"JJJJ%"
        %"EEEE%"
        %"JJJJ%"
        %"EEEE%"
        %"JJJJ%"
        %"EEEE%"
        %"JJJJ%"
        %"EEEE%"
        %"JJJJ%"
        %"EEEE%")}  {(READBITMAP)(16 16
        %"HBDB%"
        %"BHAA%"
        %"DDHD%"
        %"AABH%"
        %"HHDA%"
        %"BBAD%"
        %"DDHB%"
        %"AABH%"
        %"HDAD%"
        %"AADA%"
        %"DHBH%"
        %"BBHB%"
        %"HHAD%"
        %"ABDA%"
        %"DDHH%"
        %"BABB%")}  {(READBITMAP)(16 16
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%"
        %"@@@@%")}  {(READBITMAP)(16 16
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%"
        %"LLLL%")}  {(READBITMAP)(16 16
        %"CLCL%"
        %"O@O@%"
        %"LCLC%"
        %"@O@O%"
        %"CLCL%"
        %"O@O@%"
```

```
        %"LCLC%"
        %"@O@O%"
        %"CLCL%"
        %"O@O@%"
        %"LCLC%"
        %"@O@O%"
        %"CLCL%"
        %"O@O@%"
        %"LCLC%"
        %"@O@O%")}  {(READBITMAP)(16 16
        %"LFGA%"
        %"NCCH%"
        %"GAIL%"
        %"CHLN%"
        %"ALFG%"
        %"HNCC%"
        %"LGAI%"
        %"NCHL%"
        %"GALF%"
        %"CHNC%"
        %"ILGA%"
        %"LNCH%"
        %"FGAL%"
        %"CCHN%"
        %"AILG%"
        %"HLNC%")})
        ")

(DECLARE%: DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

(RECORD HLS (HUE LIGHTNESS SATURATION))

(RECORD RGB (RED GREEN BLUE))
)
)
```

;; END EXPORTED DEFINITIONS

```
(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA UNIONREGIONS INTERSECTREGIONS)
)
```

FUNCTION INDEX

VARIABLE INDEX

RECORD INDEX

MACRO INDEX

CONSTANT INDEX