File created: 3-Apr-86 18:16:05 **{ERIS}<LISPCORE>LIBRARY>C150STREAM.;15**

changes to: (FNS CREATEC150BUFFER)
(VARS C150COLORMAP C150FONTDIRECTORIES)

previous date: 3-Apr-86 16:05:11 {ERIS}<LISPCORE>LIBRARY>C150STREAM.;14

Read Table: OLD-INTERLISP-FILE

Package: INTERLISP

Format: XCCS


```
(RPAQQ C150STREAMCOMS
       ((CONSTANTS \C150PointsPerInch \C150RealBPP)
        (FNS C150.SEPARATOR C150.SETMARGINS \C150.ALLWHITESPACE \C150.BUFFER.DOT \C150.MICROLINEFEED
             \C150.SENDLINE \C150.SENDLINEINFO \C150INIT \CREATECHARSET.C150)
        (FNS CREATEC150BUFFER NEWLINE.C150 NEWPAGE.C150 OPENC150STREAM C150.RESET SEND.TO.C150 STARTPAGE.C150
             \BITBLT.C150 \BLTCHAR.C150 \BLTSHADE.C150 \C150.CRLF \CHANGECHARSET.C150 \CHARWIDTH.C150
             \CLOSEFN.C150 \CREATEC150FONT \READC150FONTFILE \DRAWCIRCLE.C150 \DRAWCURVE.C150 \DRAWELLIPSE.C150
             \DRAWLINE.C150 \DSPBACKCOLOR.C150 \DSPCLIPPINGREGION.C150 \DSPCOLOR.C150 \C150.ASSURE.COLOR
             \C150.LOOKUPRGB \DSPFONT.C150 \DSPLEFTMARGIN.C150 \DSPLINEFEED.C150 \DSPOPERATION.C150
             \DSPPRINTCHAR.C150 \DSPPRINTCR/LF.C150 \DSPRESET.C150 \DSPRIGHTMARGIN.C150 \DSPXPOSITION.C150
             \DSPYPOSITION.C150 \DUMPPAGEBUFFER.C150 \FILLCIRCLE.C150 \OUTCHARFN.C150 \SEARCHC150FONTFILES
             \STRINGWIDTH.C150)
        (VARS MISSINGC150FONTCOERCIONS (\C150COLORTABLE)
              (\C150.FRAMEBUFFER)
              (\C150STREAM)
              C150COLORMAP C150FONTCOERCIONS C150FONTDIRECTORIES C150FONTEXTENSIONS)
        (INITVARS (C150.CLIPBUFFER T)
                  (\C150DEFAULTDEVICE (QUOTE CENTRONICS)))
        (FNS COLORMAP.TO.C150TABLE)
        (FILES COLOR XXGEOM XXFILL)
        [P (IF (NOT (GETD (QUOTE POLYSHADE.BLT)))
               THEN
               (* A fix for KOTO, which is not necessary in <lc>n>)
               (MOVD (QUOTE POLYSHADE.DISPLAY)
                     (QUOTE POLYSHADE.BLT]
        (DECLARE: DONTEVAL@LOAD DOCOPY (P (\C150INIT))
               (FILES CENTRONICS))
        (DECLARE: EVAL@LOAD DONTCOPY (FILES (LOADFROM)
                                            ADISPLAY LLDISPLAY))
        (MACROS \C150BackingStream)))

(DECLARE: EVAL@COMPILE

(RPAQQ \C150PointsPerInch 120)

(RPAQQ \C150RealBPP 4)

(CONSTANTS \C150PointsPerInch \C150RealBPP)
)

(DEFINEQ
```

(C150.SEPARATOR

```
  [LAMBDA (BACKINGSTREAM)                                    (* hdj " 5-Sep-85 12:12")
    (LET ((SEPR.LENGTH 30))
         (for C instring (CONCAT "Ÿg0" SEPR.LENGTH " ") do (BOUT BACKINGSTREAM C))
         (for DASH from 1 to SEPR.LENGTH do (BOUT BACKINGSTREAM 255])
```

(C150.SETMARGINS

```
  [LAMBDA (BACKINGSTREAM C150LEFT C150RIGHT)                 (* hdj " 5-Sep-85 12:21")
```

(* * Set the left and right margins for the C150 printer)

```
    (LET [[LEFTCODE (CONCAT (FIX (TIMES 10 (if (OR (EQ C150LEFT NIL)
                                                   (LESSP C150LEFT 0.5)
                                                   (GEQ C150LEFT 9.0)
                                                   (GEQ C150LEFT C150RIGHT))
                                               then 0.5
                                               else C150LEFT]
          (RIGHTCODE (CONCAT (FIX (TIMES 10 (if (OR (EQ C150RIGHT NIL)
                                                    (GREATERP C150RIGHT 9)
                                                    (LEQ C150RIGHT 0.5)
                                                    (LEQ C150RIGHT C150LEFT))
                                                then 9
                                                else C150RIGHT]     (* send the left margin)
         (BOUT BACKINGSTREAM (CHARCODE ESC))
         (BOUT BACKINGSTREAM (CHARCODE l))
         (for CHAR instring LEFTCODE do (BOUT BACKINGSTREAM CHAR))
         (BOUT BACKINGSTREAM (CHARCODE CR))                     (* send the right margin)
         (BOUT BACKINGSTREAM (CHARCODE ESC))
         (BOUT BACKINGSTREAM (CHARCODE r))
```

```
        (for CHAR instring RIGHTCODE do (BOUT BACKINGSTREAM CHAR))
        (BOUT BACKINGSTREAM (CHARCODE CR])
```

(\C150.ALLWHITESPACE

```
[LAMBDA (BITMAP TABLES STARTINGSCAN)                    (* hdj " 6-Aug-85 15:50")
                                                        (* is there anything to print on the next 4 scanlines?)
  (LET* ((MaxX (SUB1 (BITMAPWIDTH BITMAP)))
         [MaxColor (SUB1 (EXPT 2 (BITSPERPIXEL BITMAP]
         (COLORUSED? (ARRAY (ADD1 MaxColor)
                            (QUOTE POINTER)
                           NIL 0))
         (BlackTable (ELT TABLES 0))
         (MagentaTable (ELT TABLES 1))
         (YellowTable (ELT TABLES 2))
         (CyanTable (ELT TABLES 3)))
        (for Scanline from STARTINGSCAN to (IDIFFERENCE STARTINGSCAN 3) by -1
           do (for X from 0 to MaxX do (SETA COLORUSED? (BITMAPBIT BITMAP X Scanline)
                                              T)))
        (for Value from 0 to MaxColor never (AND (ELT COLORUSED? Value)
                                                 (OR (EQ (ELT BlackTable Value)
                                                         1)
                                                     (EQ (ELT MagentaTable Value)
                                                         1)
                                                     (EQ (ELT YellowTable Value)
                                                         1)
                                                     (EQ (ELT CyanTable Value)
                                                         1])
```

(\C150.BUFFER.DOT

```
[LAMBDA (DOT X BUFFER)                                  (* hdj " 3-Aug-85 20:55")
  (SETA BUFFER X DOT])
```

(\C150.MICROLINEFEED

```
[LAMBDA (BACKINGSTREAM)                                 (* hdj " 5-Sep-85 12:12")
  (for CHAR instring "Ÿk1" do (BOUT BACKINGSTREAM CHAR])
```

(\C150.SENDLINE

```
[LAMBDA (BACKINGSTREAM LINE# COLOR BUFFER)              (* hdj " 5-Sep-85 12:13")
  (for CHAR instring (CONCAT "Ÿg" (CHARACTER (IPLUS (ITIMES 4 COLOR)
                                                    (IREMAINDER LINE# 4)
                                                    (CHARCODE 0)))
                             (FOLDHI (ARRAYSIZE BUFFER)
                                     8)
                             " ")
     do (BOUT BACKINGSTREAM CHAR))
  (bind (BYTE.TO.SEND _ 0) for BYTE from 0 to (SUB1 (ARRAYSIZE BUFFER)) by 8
     do [for BIT from 7 to 0 by -1 do (SETQ BYTE.TO.SEND (LOGOR BYTE.TO.SEND (LLSH (ELT BUFFER (IPLUS BYTE BIT)
                                                                                        )
                                                                                    BIT]
        (BOUT BACKINGSTREAM BYTE.TO.SEND])
```

(\C150.SENDLINEINFO

```
[LAMBDA (BACKINGSTREAM COLOR LENGTHINBYTES LINE#)       (* hdj " 5-Sep-85 12:13")
  (for CHAR instring (CONCAT "Ÿg" (CHARACTER (IPLUS (UNFOLD COLOR 4)
                                                    LINE#
                                                    (CHARCODE 0)))
                             LENGTHINBYTES " ")
     do (BOUT BACKINGSTREAM CHAR])
```

(\C150INIT

```
[LAMBDA NIL                                             (* gbn " 5-Nov-85 19:34")
                                                        (* Initializes global variables for the C150)
  (DECLARE (GLOBALVARS \C150IMAGEOPS))
  (SETQ \C150IMAGEOPS (create IMAGEOPS
                              IMAGETYPE _ (QUOTE C150)
                              IMFONT _ (FUNCTION \DSPFONT.C150)
                              IMLEFTMARGIN _ (FUNCTION \DSPLEFTMARGIN.C150)
                              IMRIGHTMARGIN _ (FUNCTION \DSPRIGHTMARGIN.C150)
                              IMLINEFEED _ (FUNCTION \DSPLINEFEED.C150)
                              IMXPOSITION _ (FUNCTION \DSPXPOSITION.C150)
                              IMYPOSITION _ (FUNCTION \DSPYPOSITION.C150)
                              IMCLOSEFN _ (FUNCTION \CLOSEFN.C150)
                              IMDRAWCURVE _ (FUNCTION \DRAWCURVE.C150)
                              IMFILLCIRCLE _ (QUOTE \FILLCIRCLE.C150)
                              IMDRAWLINE _ (FUNCTION \DRAWLINE.C150)
                              IMDRAWELLIPSE _ (FUNCTION \DRAWELLIPSE.C150)
                              IMDRAWCIRCLE _ (FUNCTION \DRAWCIRCLE.C150)
                              IMBITBLT _ (FUNCTION \BITBLT.C150)
                              IMBLTSHADE _ (FUNCTION \BLTSHADE.C150)
                              IMNEWPAGE _ (FUNCTION NEWPAGE.C150)
                              IMSCALE _ [FUNCTION (LAMBDA NIL
```

```
                                                    (FQUOTIENT 120 72]
                                IMSPACEFACTOR _ (FUNCTION NILL)
                                IMFONTCREATE _ (QUOTE C150)
                                IMCOLOR _ (FUNCTION \DSPCOLOR.C150)
                                IMBACKCOLOR _ (FUNCTION \DSPBACKCOLOR.C150)
                                IMOPERATION _ (FUNCTION \DSPOPERATION.C150)
                                IMSTRINGWIDTH _ (FUNCTION \STRINGWIDTH.C150)
                                IMCHARWIDTH _ (FUNCTION \CHARWIDTH.C150)
                                IMCLIPPINGREGION _ (FUNCTION \DSPCLIPPINGREGION.C150)
                                IMRESET _ (FUNCTION \DSPRESET.C150)
                                IMFILLPOLYGON _ (FUNCTION POLYSHADE.BLT)))
    [push IMAGESTREAMTYPES (LIST (QUOTE C150)
                                 (LIST (QUOTE OPENSTREAM)
                                       (FUNCTION OPENC150STREAM))
                                 (LIST (QUOTE FONTCREATE)
                                       (FUNCTION \CREATEC150FONT))
                                 (LIST (QUOTE FONTSAVAILABLE)
                                       (FUNCTION \SEARCHC150FONTFILES))
                                 (LIST (QUOTE CREATECHARSET)
                                       (FUNCTION \CREATECHARSET.C150]
    (push PRINTERTYPES (LIST (LIST (QUOTE C150))
                             (LIST (QUOTE CANPRINT)
                                   (LIST (QUOTE C150)))
                             (LIST (QUOTE STATUS)
                                   (FUNCTION TRUE))
                             (LIST (QUOTE PROPERTIES)
                                   (FUNCTION NILL))
                             (LIST (QUOTE SEND)
                                   (FUNCTION SEND.TO.C150))
                             (LIST (QUOTE BITMAPSCALE)
                                  NIL)
                             (LIST (QUOTE BITMAPFILE)
                                   NIL)))
    (ADDTOVAR DEFAULTPRINTINGHOST (C150 C150))
    (PUTPROP (QUOTE C150)
           (QUOTE PRINTERTYPE)
           (QUOTE C150))
    [push PRINTFILETYPES (LIST (QUOTE C150)
                               (LIST (QUOTE TEST)
                                     (FUNCTION NILL))
                               (LIST (QUOTE EXTENSION)
                                     (LIST (QUOTE C150]
    (DEFAULTFONT (QUOTE C150)
           (QUOTE (CLASSIC 10 MRR))
           (QUOTE NEW))
    T])
```

(\CREATECHARSET.C150

```
  [LAMBDA (FAMILY SIZE FACE ROTATION DEVICE CHARSET FONTDESC NOSLUG?)
                                                                  (* gbn " 9-Jan-86 13:00")

          (* * tries to build the csinfo required for CHARSET. Does the necessary coercions.
          Returns NIL when unsuccessful (\CREATECHARSET will do the same))

          (* * NOSLUG? means don't create an empty (slug) csinfo if the charset is not found, just return NIL)

    (DECLARE (GLOBALVARS C150FONTCOERCIONS MISSINGC150FONTCOERCIONS))

          (* C150FONTCOERCIONS is a list of font coercions, in the form
          ((user-font real-font) (user-font real-font) ...)%. Each user-font is a list of FAMILY, and optionally SIZE and CHARSET,
          (e.g., (GACHA) or (GACHA 10) or (GACHA 10 143)), and each real-font is a similar list.)

    (COND
       ((PROG1 (for TRANSL in C150FONTCOERCIONS bind NEWCSINFO USERFONT REALFONT
                  when (AND (SETQ USERFONT (CAR TRANSL))
                            (EQ FAMILY (CAR USERFONT))
                            (OR (NOT (CADR USERFONT))
                                (EQ SIZE (CADR USERFONT)))
                            (OR (NOT (CADDR USERFONT))
                                (EQ CHARSET (CADDR USERFONT)))
                            (SETQ REALFONT (CADR TRANSL))
                            (SETQ NEWCSINFO (\CREATECHARSET.C150 (OR (CAR REALFONT)
                                                                     FAMILY)
                                                                 (OR (CADR REALFONT)
                                                                     SIZE)
                                                                 FACE ROTATION DEVICE (OR (CADDR REALFONT)
                                                                                          CHARSET)
                                                                 FONTDESC NOSLUG?)))
                  do (RETURN NEWCSINFO))                          (* Just recursively call ourselves to handle entries in
                                                                  C150FONTCOERCIONS)
             ))
       ((AND (EQ ROTATION 0)

          (* If it is available, this will force the appropriate file to be read to fill in the charset entry)

          (\READC150FONTFILE FAMILY SIZE FACE ROTATION (QUOTE C150)
```

```
                CHARSET)))
        (T
```

(* * if we get here, the font is not directly available, either it needs to be rotated, boldified, or italicised "by hand")

```
           (PROG (NEWFONT XFONT XLATEDFAM)
                 (RETURN (COND
                             [(NEQ ROTATION 0)
```

(* to make a rotated font (even if it is bold or whatnot), recursively call fontcreate to get the unrotated font (maybe bold, etc), then call \SFMAKEROTATEDFONT on the csinfo.)

```
                              (OR (MEMB ROTATION (QUOTE (90 270)))
                                  (ERROR "only implemented rotations are 0, 90 and 270." ROTATION))
                              (COND
                                 ((SETQ XFONT (FONTCREATE FAMILY SIZE FACE 0 (QUOTE C150)
                                                          T CHARSET))
```

(* actually call FONTCREATE here, rather than \CREATEC150FONT or \CREATECHARSET.C150 so that the vanilla font that is built in this process will be cached and not repeated.)

```
                                     (if (SETQ CSINFO (\GETCHARSETINFO CHARSET XFONT T))
                                         then (\SFROTATECSINFO CSINFO ROTATION)
                                       else NIL]
                                ((AND (EQ (fetch WEIGHT of FACE)
                                          (QUOTE BOLD))
                                      (SETQ XFONT (FONTCREATE FAMILY SIZE (create FONTFACE using FACE WEIGHT _
                                                                                                  (QUOTE MEDIUM))
                                                              0
                                                              (QUOTE C150)
                                                              T CHARSET)))
```

(* if we want a bold font, and the medium weight font is available, build the medium weight version then call \SFMAKEBOLD on the csinfo)

```
                                     (if (SETQ CSINFO (\GETCHARSETINFO CHARSET XFONT T))
                                         then (\SFMAKEBOLD CSINFO)
                                       else NIL))
                                ((AND (EQ (fetch SLOPE of FACE)
                                          (QUOTE ITALIC))
                                      (SETQ XFONT (FONTCREATE FAMILY SIZE (create FONTFACE using FACE SLOPE _
                                                                                                 (QUOTE REGULAR))
                                                              0
                                                              (QUOTE C150)
                                                              T CHARSET)))
                                     (if (SETQ CSINFO (\GETCHARSETINFO CHARSET XFONT T))
                                         then (\SFMAKEITALIC CSINFO)
                                       else NIL))
                                ((for TRANSL in MISSINGC150FONTCOERCIONS bind NEWCSINFO USERFONT REALFONT
                                   when (AND (SETQ USERFONT (CAR TRANSL))
                                             (EQ FAMILY (CAR USERFONT))
                                             (OR (NOT (CADR USERFONT))
                                                 (EQ SIZE (CADR USERFONT)))
                                             (OR (NOT (CADDR USERFONT))
                                                 (EQ CHARSET (CADDR USERFONT)))
                                             (SETQ REALFONT (CADR TRANSL))
                                             (SETQ NEWCSINFO (\CREATECHARSET.C150 (OR (CAR REALFONT)
                                                                                      FAMILY)
                                                                                  (OR (CADR REALFONT)
                                                                                      SIZE)
                                                                                  FACE ROTATION DEVICE (OR (CADDR REALFONT)
                                                                                                           CHARSET)
                                                                                  FONTDESC NOSLUG?)))
                                   do (RETURN NEWCSINFO)))
                                ((NOT NOSLUG?)
                                 (\BUILDSLUGCSINFO (fetch (FONTDESCRIPTOR FONTAVGCHARWIDTH) of FONTDESC)
                                        (FONTPROP FONTDESC (QUOTE ASCENT))
                                        (FONTPROP FONTDESC (QUOTE DESCENT))
                                        (FONTPROP FONTDESC (QUOTE DEVICE])
)

(DEFINEQ

(**CREATEC150BUFFER**
  [LAMBDA (WIDTH HEIGHT)                                    (* FS " 3-Apr-86 18:14")
    (LET* ((BITWIDTH (ITIMES WIDTH \C150RealBPP))
           (RASTERWIDTH (FOLDHI BITWIDTH BITSPERWORD))
           (PAGES (FOLDHI (ITIMES RASTERWIDTH HEIGHT)
                          WORDSPERPAGE)))
```

(* * (create BITMAP BITMAPBITSPERPIXEL _ \C150RealBPP BITMAPRASTERWIDTH _ RASTERWIDTH BITMAPWIDTH _ BITWIDTH BITMAPHEIGHT _ HEIGHT BITMAPBASE _ (OR (\ALLOCPAGEBLOCK PAGES) (HELP "Can't allocate C150 buffer - pages needed = " PAGES))))

(* * Don't think code above is correct, commented out and added below, changing BITMAPWIDTH, and ignoring \MaxBitMapWords (safe?????) * *)

```
          (create BITMAP
                  BITMAPBITSPERPIXEL _ \C150RealBPP
                  BITMAPRASTERWIDTH _ RASTERWIDTH
                  BITMAPWIDTH _ WIDTH
                  BITMAPHEIGHT _ HEIGHT
                  BITMAPBASE _ (OR (\ALLOCPAGEBLOCK PAGES)
                                   (HELP "Can't allocate C150 buffer - pages needed = " PAGES])
```

(NEWLINE.C150

```
  [LAMBDA (C150STREAM)                                                    (* hdj " 6-Jun-85 14:01")
                                                                          (* Go to next line (or next page if on last line))
    (LET* [(C150DATA (fetch IMAGEDATA of C150STREAM))
           (NEWYPOS (IPLUS (ffetch DDYPOSITION of C150DATA)
                           (ffetch DDLINEFEED of C150DATA]
          (COND
             ((ILESSP NEWYPOS (ffetch DDClippingBottom of C150DATA))
              (NEWPAGE.C150 C150STREAM))
             (T (\DSPXPOSITION.C150 C150STREAM (ffetch DDLeftMargin of C150DATA))
                (\DSPYPOSITION.C150 C150STREAM NEWYPOS])
```

(NEWPAGE.C150

```
  [LAMBDA (C150STREAM)                                                    (* hdj " 7-Aug-85 16:48")
    (LET ((DD (fetch (STREAM IMAGEDATA) of C150STREAM)))
         [\DUMPPAGEBUFFER.C150 (fetch DDDestination of DD)
                C150STREAM
                (OR \C150COLORTABLE (SETQ \C150COLORTABLE (COLORMAP.TO.C150TABLE C150COLORMAP]
         (STARTPAGE.C150 C150STREAM])
```

(OPENC150STREAM

```
  [LAMBDA (C150FILE OPTIONS)                                              (* gbn " 6-Nov-85 19:08")
                                                                          (* Opens a C150 stream)
```

(* open a C150 stream. keep a permanent pointer to the frame buffer, because it can never be gc'ed any way, and we want to recycle it -- only allow one of them to be open at a time, due to global frame buffer)

```
    (DECLARE (GLOBALVARS \C150IMAGEOPS C150BAUDRATE \C150STREAM))
    (if (AND (STREAMP \C150STREAM)
             (OPENP \C150STREAM))
        then (ERROR "Sorry - you can only have one C150 stream open at one time" \C150STREAM)
      else (if (EQ (FILENAMEFIELD C150FILE (QUOTE HOST))
                   (QUOTE LPT))
               then
```

(* if the hardcopy interface is opening to the LPT pseudodevice, change it to be the device that the printer is actually connected to.)

```
                 (SETQ C150FILE (PACKFILENAME (QUOTE HOST)
                                              \C150DEFAULTDEVICE
                                              (QUOTE BODY)
                                              C150FILE)))
          (LET* [(WIDTH (FIX (TIMES 8.5 \C150PointsPerInch)))
                 (HEIGHT (FIX (TIMES 11 \C150PointsPerInch)))
                 (BACKINGSTREAM (OPENSTREAM C150FILE (QUOTE OUTPUT)))
                 (C150STREAM (SETQ \C150STREAM (DSPCREATE (OR \C150.FRAMEBUFFER (SETQ \C150.FRAMEBUFFER
                                                                                      (CREATEC150BUFFER WIDTH
                                                                                                        HEIGHT]
                (replace (STREAM F1) of C150STREAM with BACKINGSTREAM)
                (replace (STREAM OUTCHARFN) of C150STREAM with (FUNCTION \OUTCHARFN.C150))
                (replace (STREAM STRMBOUTFN) of C150STREAM with (FUNCTION \DSPPRINTCHAR.C150))
                (replace (STREAM USERCLOSEABLE) of C150STREAM with T)
                (replace (STREAM IMAGEOPS) of C150STREAM with \C150IMAGEOPS)
                (replace (\DISPLAYDATA DDClippingRegion) of (\GETDISPLAYDATA C150STREAM)
                   with (CREATEREGION 0 0 WIDTH HEIGHT))
                (STREAMPROP C150STREAM (QUOTE COLORMAPCACHE)
                       (LIST NIL))
                (DSPLEFTMARGIN 0 C150STREAM)
                (DSPRIGHTMARGIN WIDTH C150STREAM)
                (DSPCOLOR 0 C150STREAM)
                (DSPBACKCOLOR 7 C150STREAM)
                (STARTPAGE.C150 C150STREAM)
                C150STREAM])
```

(C150.RESET

```
  [LAMBDA NIL                                                             (* gbn " 7-Nov-85 22:42")
```

(* * just does things that the user prob doesn't know about.)

```
    (SETQ \C150STREAM)
    (CLOSEF? (QUOTE {CENTRONICS}))
    (CENTRONICS.RESET])
```

(SEND.TO.C150

```
  [LAMBDA (HOST FILE PRINTOPTIONS)                                    (* hdj " 6-Jun-85 15:37")
    (COPYFILE FILE (PACKFILENAME (QUOTE HOST)
                          (QUOTE LPT)
                          (QUOTE NAME)
                         HOST
                          (QUOTE EXTENSION)
                          (QUOTE C150])
```

(**STARTPAGE.C150**

```
  [LAMBDA (C150STREAM)                                                (* hdj " 6-Aug-85 11:20")
    (LET* ((DD (\GETDISPLAYDATA C150STREAM))
           (CREG (fetch DDClippingRegion of DD))
           (FONTASCENT (FONTASCENT (fetch DDFONT of DD)))
           (PAGEBUFFER (fetch DDDestination of DD)))
          (BLTSHADE (DSPBACKCOLOR NIL C150STREAM)
                 PAGEBUFFER)
          (\DSPXPOSITION.C150 C150STREAM (fetch DDLeftMargin of DD))
          (\DSPYPOSITION.C150 C150STREAM (ADD1 (IDIFFERENCE (fetch TOP of CREG)
                                                            FONTASCENT])
```

(**\BITBLT.C150**

```
  [LAMBDA (SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTSTRM DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT
                 SOURCETYPE OPERATION TEXTURE CLIPPINGREGION CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM)
                                                                      (* hdj " 6-Jun-85 16:17")
    (DECLARE (LOCALVARS . T))
    (PROG (stodx stody left top bottom right DESTBITMAP DESTINATIONNBITS (SOURCENBITS (fetch (BITMAP
                                                                                          BITMAPBITSPERPIXEL
                                                                                          )
                                                                                 of SOURCEBITMAP))
           (DESTDD (fetch IMAGEDATA of DESTSTRM)))
          (SETQ DESTBITMAP (fetch DDDestination of DESTDD))
          [PROGN                                                      (* compute limits based on clipping regions.)
                 (SETQ left (fetch DDClippingLeft of DESTDD))
                 (SETQ bottom (fetch DDClippingBottom of DESTDD))
                 (SETQ right (fetch DDClippingRight of DESTDD))
                 (SETQ top (fetch DDClippingTop of DESTDD))
                 (COND
                    (CLIPPINGREGION                                   (* hard case, two destination clipping regions: do calculations to
                                                                         merge them.)
                           (PROG (CRLEFT CRBOTTOM)
                                 [SETQ left (IMAX left (SETQ CRLEFT (fetch LEFT of CLIPPINGREGION]
                                 [SETQ bottom (IMAX bottom (SETQ CRBOTTOM (fetch BOTTOM of CLIPPINGREGION]
                                 [SETQ right (IMIN right (IPLUS CRLEFT (fetch WIDTH of CLIPPINGREGION]
                                 (SETQ top (IMIN top (IPLUS CRBOTTOM (fetch HEIGHT of CLIPPINGREGION]
          (SETQ DESTINATIONNBITS (fetch (BITMAP BITMAPBITSPERPIXEL) of DESTBITMAP))

          (* left, right top and bottom are the limits in destination taking into account Clipping Regions.
          Clip to region in the arguments of this call.)

          [PROGN (SETQ left (IMAX DESTINATIONLEFT left))
                 (SETQ bottom (IMAX DESTINATIONBOTTOM bottom))
                 [COND
                    (WIDTH                                            (* WIDTH is optional)
                           (SETQ right (IMIN (IPLUS DESTINATIONLEFT WIDTH)
                                             right]
                 (COND
                    (HEIGHT                                           (* HEIGHT is optional)
                           (SETQ top (IMIN (IPLUS DESTINATIONBOTTOM HEIGHT)
                                           top]                       (* Clip and translate coordinates.)
          (SETQ stodx (IDIFFERENCE DESTINATIONLEFT SOURCELEFT))
          (SETQ stody (IDIFFERENCE DESTINATIONBOTTOM SOURCEBOTTOM))

          (* compute the source dimensions (left right bottom top) by intersecting the source bit map, the source area to be moved
          with the limits of the region to be moved in the destination coordinates.)

          [PROGN                                                      (* compute left margin)
                 (SETQ left (IMAX CLIPPEDSOURCELEFT (IDIFFERENCE left stodx)
                                  0))                                 (* compute bottom margin)
                 (SETQ bottom (IMAX CLIPPEDSOURCEBOTTOM (IDIFFERENCE bottom stody)
                                    0))                               (* compute right margin)
                 (SETQ right (IMIN (\PIXELOFBITADDRESS SOURCENBITS (ffetch BITMAPWIDTH of SOURCEBITMAP))
                                   (IDIFFERENCE right stodx)
                                   (IPLUS CLIPPEDSOURCELEFT WIDTH)))
                                                                      (* compute top margin)
                 (SETQ top (IMIN (ffetch BITMAPHEIGHT of SOURCEBITMAP)
                                 (IDIFFERENCE top stody)
                                 (IPLUS CLIPPEDSOURCEBOTTOM HEIGHT]
          (COND
             ((AND (IGREATERP right left)
                   (IGREATERP top bottom)))
             (T                                                       (* there is nothing to move.)
                (RETURN)))
          (OR OPERATION (SETQ OPERATION (ffetch (\DISPLAYDATA DDOPERATION) of DESTDD)))

          (* We'd rather handle the slow case when we are interruptable, so we do it here as a heuristic.
          But we might get interrupted before we go interruptable, so we do it there too.)
```

```
        (COND
           [(EQ SOURCENBITS DESTINATIONNBITS)                    (* going from one to another of the same size.)
                                                                 (* use LLSH with constant value rather than multiple because it
            compiles into opcodes.)
            [COND
               ((EQ DESTINATIONNBITS 4)
                (SETQ left (LLSH left 2))
                (SETQ right (LLSH right 2))
                (SETQ stodx (LLSH stodx 2)))
               (T (SETQ left (LLSH left 3))
                  (SETQ right (LLSH right 3))
                  (SETQ stodx (LLSH stodx 3]                     (* set texture if it will ever get looked at.)
            (AND (EQ SOURCETYPE (QUOTE MERGE))
                 (SETQ TEXTURE (COLORTEXTUREFROMCOLOR# TEXTURE DESTINATIONNBITS)))
                                                                 (* easy case of color to color)
            (PROG ([PILOTBBT (COND
                                ((type? PILOTBBT \SYSPILOTBBT)
                                 \SYSPILOTBBT)
                                (T (SETQ \SYSPILOTBBT (create PILOTBBT]
                   (HEIGHT (IDIFFERENCE top bottom))
                   (WIDTH (IDIFFERENCE right left))
                   (DTY (\SFInvert DESTBITMAP (IPLUS top stody)))
                   (DLX (IPLUS left stodx))
                   (STY (\SFInvert SOURCEBITMAP top))
                   (SLX left))
                  (replace PBTWIDTH of PILOTBBT with WIDTH)
                  (replace PBTHEIGHT of PILOTBBT with HEIGHT)
                  (COND
                     ((EQ SOURCETYPE (QUOTE MERGE))
                      (\BITBLT.MERGE PILOTBBT SOURCEBITMAP SLX STY DESTBITMAP DLX DTY WIDTH HEIGHT OPERATION
                            TEXTURE))
                     (T (\BITBLTSUB PILOTBBT SOURCEBITMAP SLX STY DESTBITMAP DLX DTY HEIGHT SOURCETYPE
                               OPERATION TEXTURE]
           [(EQ SOURCENBITS 1)                                   (* going from a black and white bitmap to a color map)
            (AND SOURCETYPE (NEQ SOURCETYPE (QUOTE INPUT))
                 (ERROR "SourceType not implemented from B&W to color bitmaps." SOURCETYPE))
            (PROG ((HEIGHT (IDIFFERENCE top bottom))
                   (WIDTH (IDIFFERENCE right left))
                   (DBOT (IPLUS bottom stody))
                   (DLFT (IPLUS left stodx)))
                  (SELECTQ OPERATION
                       ((NIL REPLACE)
                            (\BWTOCOLORBLT SOURCEBITMAP left bottom DESTBITMAP DLFT DBOT WIDTH HEIGHT
                                   (COLORNUMBERP (fetch (\DISPLAYDATA DDBACKGROUNDCOLOR) of DESTDD))
                                   (COLORNUMBERP (fetch (\DISPLAYDATA DDFOREGROUNDCOLOR) of DESTDD))
                                   DESTINATIONNBITS))
                       (PAINT)
                       (INVERT)
                       (ERASE)
                       (SHOULDNT]
           (T                                                    (* going from color map into black and white map.)
              (ERROR "not implemented to blt between bitmaps of different pixel size.")))
        (RETURN T])
```

(**BLTCHAR.C150**

```
  [LAMBDA (CHARCODE C150STREAM C150DATA)                         (* hdj "19-Jul-85 13:32")

        (* * puts a character on a C150STREAM. Since a C150STREAM is based on a color bitmap stream, we can use
        \SLOWBLTCHAR)

    [COND
       ((NEQ (ffetch DDCHARSET of C150DATA)
             (\CHARSET CHARCODE))                                (* The charset has changed.)
        (\CHANGECHARSET.C150 C150DATA (\CHARSET CHARCODE]
    (LET [(CHAR8CODE (\CHAR8CODE CHARCODE))
          (ROTATION (ffetch (FONTDESCRIPTOR ROTATION) of (ffetch DDFONT of C150DATA]
         (COND
            [(EQ 0 ROTATION)
             (PROG (NEWX LEFT RIGHT (CURX (ffetch DDXPOSITION of C150DATA)))
                   [COND
                      ((IGREATERP (SETQ NEWX (IPLUS CURX (\DSPGETCHARWIDTH CHAR8CODE C150DATA)))
                                  (ffetch DDRightMargin of C150DATA))  (* past RIGHT margin, force eol)
                       (\DSPPRINTCR/LF.C150 (CHARCODE EOL)
                              C150STREAM)
                       (SETQ CURX (ffetch DDXPOSITION of C150DATA))
                       (SETQ NEWX (IPLUS CURX (\DSPGETCHARWIDTH CHAR8CODE C150DATA]
                                                                 (* update the x position.)
                   (freplace DDXPOSITION of C150DATA with NEWX)
                   (SETQ LEFT (IMAX (ffetch DDClippingLeft of C150DATA)
                                    CURX))
                   (SETQ RIGHT (IMIN (ffetch DDClippingRight of C150DATA)
                                     NEWX))
                   (COND
                      ((AND (ILESSP LEFT RIGHT)
                            (NEQ (ffetch PBTHEIGHT of (SETQ NEWX (ffetch DDPILOTBBT of C150DATA)))
```

```
                        0))
                  (SELECTQ (ffetch (BITMAP BITMAPBITSPERPIXEL) of (ffetch (\DISPLAYDATA DDDestination)
                                                                          of C150DATA))
                      (1 (freplace PBTDESTBIT of NEWX with LEFT)
                         (freplace PBTWIDTH of NEWX with (IDIFFERENCE RIGHT LEFT))
                         (freplace PBTSOURCEBIT of NEWX with (IDIFFERENCE (IPLUS (\DSPGETCHAROFFSET CHAR8CODE
                                                                                       C150DATA)
                                                                                 LEFT)
                                                                          CURX))
                         (\PILOTBITBLT NEWX 0))
                      (4 (freplace PBTDESTBIT of NEWX with (SETQ LEFT (LLSH LEFT 2)))
                         (freplace PBTWIDTH of NEWX with (IDIFFERENCE (LLSH RIGHT 2)
                                                                      LEFT))
                         (freplace PBTSOURCEBIT of NEWX with (IDIFFERENCE (IPLUS (LLSH (\DSPGETCHAROFFSET
                                                                                        CHAR8CODE C150DATA)
                                                                                       2)
                                                                                 LEFT)
                                                                          (LLSH CURX 2)))
                         (\PILOTBITBLT NEWX 0))
                      (8 (freplace PBTDESTBIT of NEWX with (SETQ LEFT (LLSH LEFT 3)))
                         (freplace PBTWIDTH of NEWX with (IDIFFERENCE (LLSH RIGHT 3)
                                                                      LEFT))
                         (freplace PBTSOURCEBIT of NEWX with (IDIFFERENCE (IPLUS (LLSH (\DSPGETCHAROFFSET
                                                                                        CHAR8CODE C150DATA)
                                                                                       3)
                                                                                 LEFT)
                                                                          (LLSH CURX 3)))
                         (\PILOTBITBLT NEWX 0))
                      (SHOULDNT))
                  T]
            (T                                                    (* handle rotated fonts)
               (LET [(YPOS (ffetch DDYPOSITION of C150DATA))
                     (HEIGHTMOVED (\DSPGETCHARWIDTH CHAR8CODE C150DATA))
                     (CSINFO (\GETCHARSETINFO (\CHARSET CHARCODE)
                                    (ffetch DDFONT of C150DATA]
                    (COND
                       ((EQ ROTATION 90)                          (* don't force CR for rotated fonts.)
                        (\DSPYPOSITION.C150 C150STREAM (IPLUS YPOS HEIGHTMOVED))
                                                                  (* update the display stream x position.)
                        (BITBLT (ffetch (CHARSETINFO CHARSETBITMAP) of CSINFO)
                               0
                               (\DSPGETCHAROFFSET CHAR8CODE C150DATA)
                               C150STREAM
                               (ADD1 (IDIFFERENCE (ffetch DDXPOSITION of C150DATA)
                                                  (ffetch (CHARSETINFO CHARSETASCENT) of CSINFO)))
                               YPOS
                               (IPLUS (ffetch (CHARSETINFO CHARSETASCENT) of CSINFO)
                                      (ffetch (CHARSETINFO CHARSETDESCENT) of CSINFO))
                               HEIGHTMOVED))
                       ((EQ ROTATION 270)
                        (\DSPYPOSITION.C150 C150STREAM (IDIFFERENCE YPOS HEIGHTMOVED))
                        (BITBLT (ffetch (CHARSETINFO CHARSETBITMAP) of CSINFO)
                               0
                               (\DSPGETCHAROFFSET CHAR8CODE C150DATA)
                               C150STREAM
                               (IDIFFERENCE (ffetch DDXPOSITION of C150DATA)
                                      (ffetch (CHARSETINFO CHARSETDESCENT) of CSINFO))
                               (ffetch DDYPOSITION of C150STREAM)
                               (IPLUS (ffetch (CHARSETINFO CHARSETASCENT) of CSINFO)
                                      (ffetch (CHARSETINFO CHARSETDESCENT) of CSINFO))
                               HEIGHTMOVED))
                       (T (ERROR "Not implemented to rotate by other than 0, 90 or 270"])
```

(**BLTSHADE.C150**

```
  [LAMBDA (TEXTURE STREAM DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT OPERATION CLIPPINGREGION)
                                                                  (* gbn " 5-Nov-85 18:42")
                                                                  (* BLTSHADE to C150 color printer)
    (DECLARE (LOCALVARS . T))
    (PROG (left top bottom right DESTINATIONNBITS DESTINATIONBITMAP (DESTDD (fetch IMAGEDATA of STREAM)))
          (SETQ DESTINATIONLEFT DESTINATIONLEFT)
          (SETQ DESTINATIONBOTTOM DESTINATIONBOTTOM)
          [PROGN                                                  (* compute limits based on clipping regions.)
                 (SETQ left (ffetch DDClippingLeft of DESTDD))
                 (SETQ bottom (ffetch DDClippingBottom of DESTDD))
                 (SETQ right (ffetch DDClippingRight of DESTDD))
                 (SETQ top (ffetch DDClippingTop of DESTDD))
                 (COND
                    (CLIPPINGREGION                               (* hard case, two destination clipping regions: do calculations to
                                                                  merge them.)
                           (PROG (CRLEFT CRBOTTOM)
                                 [SETQ left (IMAX left (SETQ CRLEFT (fetch LEFT of CLIPPINGREGION]
                                 [SETQ bottom (IMAX bottom (SETQ CRBOTTOM (fetch BOTTOM of CLIPPINGREGION]
                                 [SETQ right (IMIN right (IPLUS CRLEFT (fetch WIDTH of CLIPPINGREGION]
                                 (SETQ top (IMIN top (IPLUS CRBOTTOM (fetch HEIGHT of CLIPPINGREGION]
          [SETQ DESTINATIONNBITS (fetch (BITMAP BITMAPBITSPERPIXEL) of (SETQ DESTINATIONBITMAP
                                                                              (fetch DDDestination of DESTDD]
```

```
                                                                (* SETQ right (\PIXELOFBITADDRESS DESTINATIONNBITS
                                                                right))

        (* left, right top and bottom are the limits in destination taking into account Clipping Regions.
        Clip to region in the arguments of this call.)

        [PROGN (SETQ left (IMAX DESTINATIONLEFT left))
               (SETQ bottom (IMAX DESTINATIONBOTTOM bottom))
               [COND
                  (WIDTH                                        (* WIDTH is optional)
                         (SETQ right (IMIN (IPLUS DESTINATIONLEFT WIDTH)
                                           right]
               (COND
                  (HEIGHT                                       (* HEIGHT is optional)
                         (SETQ top (IMIN (IPLUS DESTINATIONBOTTOM HEIGHT)
                                         top]
        (COND
           ((OR (ILEQ right left)
                (ILEQ top bottom))                              (* there is nothing to move.)
            (RETURN)))
        [SETQ TEXTURE (COND
                         ((NULL TEXTURE)
                          (DSPBACKCOLOR NIL STREAM))
                         [(FIXP TEXTURE)

        (* if fixp use the low order bits as a color number. This picks up the case of BLACKSHADE being used to INVERT.)

                          (OR (COLORNUMBERP TEXTURE DESTINATIONNBITS T)
                              (LOGAND TEXTURE (COND
                                                 ((EQ DESTINATIONNBITS 4)
                                                  15)
                                                 (T 255]
                         (T (\C150.ASSURE.COLOR TEXTURE STREAM]
                                                                (* filling an area with a texture.)
        (SETQ left (ITIMES DESTINATIONNBITS left))
        (SETQ right (ITIMES DESTINATIONNBITS right))
        (SETQ TEXTURE (COLORTEXTUREFROMCOLOR# TEXTURE DESTINATIONNBITS))

        (* easy case of black and white bitmap into black and white or color to color or texture filling.)

        (* We'd rather handle the slow case when we are interruptable, so we do it here as a heuristic.
        But we might get interrupted before we go interruptable, so we do it there too.)

        (PROG ([PILOTBBT (COND
                            ((type? PILOTBBT \SYSPILOTBBT)
                             \SYSPILOTBBT)
                            (T (SETQ \SYSPILOTBBT (create PILOTBBT]
               (HEIGHT (IDIFFERENCE top bottom)))
              (replace PBTWIDTH of PILOTBBT with (IDIFFERENCE right left))
              (replace PBTHEIGHT of PILOTBBT with HEIGHT)
              (\BITBLTSUB PILOTBBT NIL left NIL DESTINATIONBITMAP left (\SFInvert DESTINATIONBITMAP top)
                     HEIGHT
                     (QUOTE TEXTURE)
                     (OR OPERATION (ffetch (\DISPLAYDATA DDOPERATION) of DESTDD))
                     TEXTURE))
        (RETURN T])
```

(**C150.CRLF**

```
  [LAMBDA (STREAM)                                              (* hdj "25-Jan-85 17:11")
                                                                (* Send a CRLF to the printer)

    (BOUT STREAM (CHARCODE CR))
    (BOUT STREAM (CHARCODE LF])
```

(**CHANGECHARSET.C150**

```
  [LAMBDA (DISPLAYDATA CHARSET)                                 (* hdj "19-Jul-85 13:48")

        (* Called when the character set information cached in a display stream doesn't correspond to CHARSET)

    (PROG [BM (PBT (ffetch DDPILOTBBT of DISPLAYDATA))
           (CSINFO (COND
                      ((IEQP 1 (fetch (BITMAP BITMAPBITSPERPIXEL) of (fetch (\DISPLAYDATA DDDestination)
                                                                            of DISPLAYDATA)))
                       (\GETCHARSETINFO CHARSET (ffetch DDFONT of DISPLAYDATA)))
                      (T (\GETCOLORCSINFO (fetch (\DISPLAYDATA DDFONT) of DISPLAYDATA)
                                 (fetch DDFOREGROUNDCOLOR of DISPLAYDATA)
                                 (fetch DDBACKGROUNDCOLOR of DISPLAYDATA)
                                 (fetch (BITMAP BITMAPBITSPERPIXEL) of (fetch (\DISPLAYDATA DDDestination)
                                                                              of DISPLAYDATA))
                                 CHARSET]
        (UNINTERRUPTABLY
           (freplace DDWIDTHSCACHE of DISPLAYDATA with (ffetch (CHARSETINFO WIDTHS) of CSINFO))
           (freplace DDOFFSETSCACHE of DISPLAYDATA with (ffetch (CHARSETINFO OFFSETS) of CSINFO))
           (freplace DDCHARIMAGEWIDTHS of DISPLAYDATA with (ffetch (CHARSETINFO IMAGEWIDTHS) of CSINFO))
           (freplace DDCHARSET of DISPLAYDATA with CHARSET)
           (SETQ BM (ffetch CHARSETBITMAP of CSINFO))
```

```
            (freplace PBTSOURCEBPL of PBT with (UNFOLD (ffetch BITMAPRASTERWIDTH of BM)
                                                       BITSPERWORD))
            [if (OR (NEQ (ffetch DDCHARSETASCENT of DISPLAYDATA)
                         (ffetch CHARSETASCENT of CSINFO))
                    (NEQ (ffetch DDCHARSETDESCENT of DISPLAYDATA)
                         (ffetch CHARSETDESCENT of CSINFO)))
                then (\SFFixY DISPLAYDATA CSINFO)
              else (freplace PBTSOURCE of PBT with (\ADDBASE (ffetch BITMAPBASE of BM)
                                                             (ITIMES (ffetch BITMAPRASTERWIDTH of BM)
                                                                     (ffetch DDCHARHEIGHTDELTA of DISPLAYDATA])])
```

(\CHARWIDTH.C150

```
[LAMBDA (C150STREAM CHARCODE)                                   (* hdj " 5-Jun-85 12:56")
```

(* gets the width of a character code in a display stream. Need to fix up for spacefactor.)

```
  (\FGETWIDTH (ffetch (\DISPLAYDATA DDWIDTHSCACHE) of (ffetch IMAGEDATA of C150STREAM))
         CHARCODE])
```

(\CLOSEFN.C150

```
[LAMBDA (C150STREAM)                                            (* hdj " 4-Oct-85 12:31")
```

(* * do cleanup prefatory to closing. dump last buffer and close the backing stream)

```
  (LET ((DD (fetch (STREAM IMAGEDATA) of C150STREAM)))
       [\DUMPPAGEBUFFER.C150 (fetch DDDestination of DD)
              C150STREAM
              (OR \C150COLORTABLE (SETQ \C150COLORTABLE (COLORMAP.TO.C150TABLE C150COLORMAP]
       (CLOSEF (\C150BackingStream C150STREAM])
```

(\CREATEC150FONT

```
[LAMBDA (FAMILY SIZE FACE ROTATION DEVICE CHARSET)              (* gbn " 8-Jan-86 17:09")
```

(* * create a font for the C150, synthesizing it if we must)

```
  (PROG [(FONTDESC (create FONTDESCRIPTOR
                           FONTDEVICE _ (QUOTE C150)
                           FONTFAMILY _ FAMILY
                           FONTSIZE _ SIZE
                           FONTFACE _ FACE
                           \SFAscent _ 0
                           \SFDescent _ 0
                           \SFHeight _ 0
                           ROTATION _ ROTATION
                           FONTDEVICESPEC _ (LIST FAMILY SIZE FACE ROTATION (QUOTE C150]
       (if (\GETCHARSETINFO CHARSET FONTDESC T)
           then (RETURN FONTDESC)
         else (RETURN NIL])
```

(\READC150FONTFILE

```
[LAMBDA (FAMILY SIZE FACE ROTATION DEVICE CHARSET)              (* hdj "26-Sep-85 21:49")
  (DECLARE (GLOBALVARS C150FONTEXTENSIONS C150FONTDIRECTORIES))
  (bind FONTFILE CSINFO STRM for EXT inside C150FONTEXTENSIONS
     when (SETQ FONTFILE (FINDFILE (\FONTFILENAME FAMILY SIZE FACE EXT CHARSET)
                                   T C150FONTDIRECTORIES))
     do (SETQ STRM (OPENSTREAM FONTFILE (QUOTE INPUT)))
        (RESETLST
            (SETQ CSINFO (\READACFONTFILE STRM FAMILY SIZE FACE)))
```

(* If not a recognizable format, I guess we should keep looking for another possible extension, altho it would also be nice to tell the user that he has a bogus file.)

```
        (RETURN CSINFO])
```

(\DRAWCIRCLE.C150

```
[LAMBDA (C150STREAM CENTERX CENTERY RADIUS BRUSH DASHING)        (* gbn " 9-Jan-86 13:36")
```

(* \DRAWCIRCLE.C150 extended for color. Color is specified by either BRUSH or the DSPCOLOR of DS.)

(* * how is a litatom passed in as brush?)

```
  (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)
     (T (GLOBALRESOURCE \BRUSHBBT
               (PROG ((BRUSH (create BRUSH using BRUSH BRUSHCOLOR _ (\C150.ASSURE.COLOR (fetch BRUSHCOLOR
                                                                                            of BRUSH)
```

```
                                                                C150STREAM)))
            (X 0)
            (Y RADIUS)
            (D (ITIMES 2 (IDIFFERENCE 1 RADIUS)))
            DestinationBitMap LEFT RIGHTPLUS1 TOP BOTTOM BRUSHWIDTH BRUSHHEIGHT LEFTMINUSBRUSH
            BOTTOMMINUSBRUSH TOPMINUSBRUSH BRUSHBM DESTINATIONBASE BRUSHBASE RASTERWIDTH
            BRUSHRASTERWIDTH NBITSRIGHTPLUS1 OPERATION HEIGHTMINUS1 CX CY (BBT \BRUSHBBT)
            COLOR COLORBRUSHBASE NBITS (DISPLAYDATA (fetch IMAGEDATA of C150STREAM))
            (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 (IDIFFERENCE CENTERX (FOLDLO BRUSHWIDTH 2))))
                     (4 (SETQ CX (IDIFFERENCE CENTERX (FOLDLO (LRSH BRUSHWIDTH 2)
                                                              2))))
                     (8 (SETQ CX (IDIFFERENCE CENTERX (FOLDLO (LRSH BRUSHWIDTH 3)
                                                              2))))
                     (SHOULDNT))                    (* take into account the brush thickness.)
                (SETQ CY (IDIFFERENCE CENTERY (FOLDLO BRUSHHEIGHT 2)))
```

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

```
                ))
          [COND
             ((EQ RADIUS 1)                         (* put a single brush down.)
                                                    (* draw the top and bottom most points.)
              (COND
                 (USERFN (APPLY* USERFN CX CY C150STREAM))
                 (T (\CURVEPT CX CY)))
              (RETURN))
             (T                                     (* draw the top and bottom most points.)
                (COND
                   (USERFN (APPLY* USERFN CX (IPLUS CY RADIUS)
                                   C150STREAM)
                           (APPLY* USERFN CX (IDIFFERENCE CY RADIUS)
                                   C150STREAM))
                   (T (\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 C150STREAM)
                         (APPLY* USERFN (IDIFFERENCE CX X)
                                 CY C150STREAM))
                 (T (\CURVEPT (IPLUS CX X)
                              CY)
                    (\CURVEPT (IDIFFERENCE CX X)
```

```
                                CY]
                          (T (COND
                                (USERFN (APPLY* USERFN (IPLUS CX X)
                                                (IPLUS CY Y)
                                                C150STREAM)
                                        (APPLY* USERFN (IDIFFERENCE CX X)
                                                (IPLUS CY Y)
                                                C150STREAM)
                                        (APPLY* USERFN (IPLUS CX X)
                                                (IDIFFERENCE CY Y)
                                                C150STREAM)
                                        (APPLY* USERFN (IDIFFERENCE CX X)
                                                (IDIFFERENCE CY Y)
                                                C150STREAM))
                                   (T (\CIRCLEPTS CX CY X Y)))
                               (GO LP)))
                       (MOVETO CENTERX CENTERY C150STREAM)
                       (RETURN NIL])
```

(**DRAWCURVE.C150**

```
[LAMBDA (C150STREAM KNOTS CLOSED BRUSH DASHING)                    (* gbn "12-Jan-86 15:03")
                                                                   (* draws a spline curve with a given brush.)
  (GLOBALRESOURCE \BRUSHBBT (PROG ([DASHLST (AND DASHING (OR (AND (LISTP DASHING)
                                                                  (EVERY DASHING (FUNCTION FIXP))
                                                                  DASHING)
                                                             (\ILLEGAL.ARG DASHING]
                                   (BBT \BRUSHBBT)
                                   (CBRUSH (CREATE BRUSH USING BRUSH BRUSHCOLOR _
                                                                (\C150.ASSURE.COLOR (FETCH BRUSHCOLOR
                                                                                           OF BRUSH)
                                                                                    C150STREAM)))
                                   LKNOT)
                                  (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 (fetch XCOORD of (CAR KNOTS))
                                                 (fetch YCOORD of (CAR KNOTS))
                                                 BRUSH C150STREAM))
                                       (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 C150STREAM BBT))
                                       (\CURVE2 (PARAMETRICSPLINE KNOTS CLOSED)
                                              CBRUSH DASHLST BBT C150STREAM))
                                  (RETURN C150STREAM])
```

(**DRAWELLIPSE.C150**

```
[LAMBDA (DISPLAYSTREAM CENTERX CENTERY SEMIMINORRADIUS SEMIMAJORRADIUS ORIENTATION BRUSH DASHING)
                                                                   (* hdj " 6-Jun-85 16:17")
  (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)))
```

(* 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 (IDIFFERENCE CENTERX (FOLDLO BRUSHWIDTH 2))))
                            (4 (SETQ CX (IDIFFERENCE CENTERX (FOLDLO (LRSH BRUSHWIDTH 2)
                                                                     2))))
                            (8 (SETQ CX (IDIFFERENCE CENTERX (FOLDLO (LRSH BRUSHWIDTH 3)
                                                                     2))))
                            (SHOULDNT))
                       (SETQ CY (IDIFFERENCE CENTERY (FOLDLO BRUSHHEIGHT 2)))
```

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

```
                     ))
              (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 (\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])
```

(**DRAWLINE.C150**

```
[LAMBDA (C150STREAM X1 Y1 X2 Y2 WIDTH OPERATION COLOR)          (* gbn " 5-Nov-85 13:39")
```

(* C150STREAM is guaranteed to be a C150STREAM Draws a line from x1,y1 to x2,y2 leaving the position at x2,y2)

```
   (PROG ((DD (fetch IMAGEDATA of C150STREAM)))
         (\CLIPANDDRAWLINE (OR (FIXP X1)
                               (FIXR X1))
                (OR (FIXP Y1)
                    (FIXR Y1))
                (OR (FIXP X2)
                    (FIXR X2))
                (OR (FIXP Y2)
                    (FIXR Y2))
                [COND
                   ((NULL WIDTH)
                    1)
                   ((OR (FIXP WIDTH)
                        (FIXR WIDTH]
                (SELECTQ OPERATION
                     (NIL (ffetch DDOPERATION of DD))
                     ((REPLACE PAINT INVERT ERASE)
                         OPERATION)
                     (\ILLEGAL.ARG OPERATION))
                (ffetch DDDestination of DD)
                (ffetch DDClippingLeft of DD)
                (SUB1 (ffetch DDClippingRight of DD))
                (ffetch DDClippingBottom of DD)
                (SUB1 (ffetch DDClippingTop of DD))
                C150STREAM
                (\C150.ASSURE.COLOR COLOR C150STREAM)))
```

(* the generic case of MOVETO is used so that the hardcopy streams get handled as well.)

```
   (MOVETO X2 Y2 C150STREAM])
```

(**DSPBACKCOLOR.C150**

```
[LAMBDA (STREAM COLOR)                                          (* rmk: "12-Sep-84 09:54")
                                                                (* sets and returns a display stream's background color.)
   (PROG (COLORCELL (DD (\GETDISPLAYDATA STREAM)))
         (SETQ COLORCELL (fetch DDCOLOR of DD))
         (RETURN (COND
                    (COLOR (OR (\POSSIBLECOLOR COLOR)
                               (\ILLEGAL.ARG COLOR))
                           (PROG1 (COND
                                     (COLORCELL (PROG1 (CDR COLORCELL)
                                                       (RPLACD COLORCELL COLOR)))
                                     (T                         (* no color cell yet, make one.)
                                        (replace DDCOLOR of DD with (CONS WHITECOLOR COLOR))
                                        BLACKCOLOR))
                                  (\SFFixFont STREAM DD)))
                    (T (OR (CDR COLORCELL)
                           BLACKCOLOR])
```

(\DSPCLIPPINGREGION.C150

```
[LAMBDA (C150STREAM REGION)                                    (* hdj " 5-Jun-85 12:56")
                                                               (* sets the clipping region of a display stream.)
  (PROG ((DD (\GETDISPLAYDATA C150STREAM)))
        (RETURN (PROG1 (ffetch DDClippingRegion of DD)
                       [COND
                          (REGION (OR (type? REGION REGION)
                                      (ERROR REGION " is not a REGION."))
                                  (UNINTERRUPTABLY
                                      (freplace DDClippingRegion of DD with REGION)
                                      (\SFFixClippingRegion DD)
                                      (\SFFixY DD))])])
```

(\DSPCOLOR.C150

```
[LAMBDA (STREAM COLOR)                                         (* gbn "13-Jan-86 12:08")
                                                               (* sets and returns a display stream's foreground color.)
  (LET (CURRENTCOLOR NEWCOLOR (DD (\GETDISPLAYDATA STREAM)))
       (SETQ CURRENTCOLOR (fetch DDCOLOR of DD))
       (COND
          (COLOR (SETQ NEWCOLOR (\C150.ASSURE.COLOR COLOR STREAM))
                 (PROG1 (COND
                           (CURRENTCOLOR (PROG1 (CAR CURRENTCOLOR)
                                                (RPLACA CURRENTCOLOR NEWCOLOR)))
                           (T                                  (* no color cell yet, make one.)
                              (replace DDCOLOR of DD with (CONS NEWCOLOR BLACKCOLOR))
                              WHITECOLOR))
                        (\SFFixFont STREAM DD)))
          (T (OR (CAR CURRENTCOLOR)
                 WHITECOLOR])
```

(\C150.ASSURE.COLOR

```
[LAMBDA (COLOR# C150STREAM)                                    (* gbn " 7-Jan-86 17:44")
  (PROG (LEVELS)
        (AND (COND
                ((NULL COLOR)
                 (RETURN (DSPCOLOR NIL C150STREAM)))
                [(FIXP COLOR#)
                 (RETURN (COND
                            ((AND (IGEQ COLOR# 0)
                                  (ILESSP COLOR# 8)
                                 COLOR#))
                            (T (\ILLEGAL.ARG COLOR#]
                [(LITATOM COLOR#)
                 (RETURN (COND
                            ((SETQ LEVELS (\LOOKUPCOLORNAME COLOR#))
                                                               (* recursively look up color number)
                             (\C150.ASSURE.COLOR (CDR LEVELS)
                                    C150STREAM))
                            (T (ERROR "Unknown color name" COLOR#]
                ((EQ (LENGTH COLOR#)
                     2)

        (* temporarily, handle the case of being given a texture and a color, by using the color)

                 (RETURN (\C150.ASSURE.COLOR (CADR COLOR#)
                                C150STREAM)))
                ((HLSP COLOR#)                                 (* HLS form convert to RGB)
                 (SETQ LEVELS (HLSTORGB COLOR#)))
                ((RGBP COLOR#)                                 (* check for RGB or HLS)
                 (SETQ LEVELS COLOR#))
                ((TYPENAMEP COLOR# (QUOTE BITMAP))             (* just a hack to not blow up)
                 (RETURN (IMOD (for I from 1 to (BITMAPWIDTH COLOR#) sum (BITMAPBIT COLOR# I 1))
                               8)))
                (T (\ILLEGAL.ARG COLOR#)))
           (RETURN (COND
                      ((\C150.LOOKUPRGB LEVELS C150STREAM))
                      (T (ERROR COLOR# "not available in color map"])
```

(\C150.LOOKUPRGB

```
[LAMBDA (RGB C150STREAM)                                       (* gbn " 5-Nov-85 15:47")
```

(* * returns the colormap index whose value is RGB. Looks first in the cache, then runs through the colormap. Returns NIL if RGB NOT found)

```
  (DECLARE (GLOBALVARS C150COLORMAP))
  (PROG [INDEX (CACHE (STREAMPROP C150STREAM (QUOTE COLORMAPCACHE]
        (RETURN (if (SETQ INDEX (SASSOC RGB CACHE))
                    then (CDR INDEX)
                  else [SETQ INDEX (bind (CM _ C150COLORMAP) for I from 0 to (SUB1 (EXPT 2 3))
                                      thereis (AND (EQ (\GENERIC.COLORLEVEL CM I (QUOTE RED))
                                                       (fetch (RGB RED) of LEVELS))
                                                   (EQ (\GENERIC.COLORLEVEL CM I (QUOTE GREEN))
                                                       (fetch (RGB GREEN) of LEVELS))
```

```
                                        (EQ (\GENERIC.COLORLEVEL CM I (QUOTE BLUE))
                                            (fetch (RGB BLUE) of LEVELS]
                        (if INDEX
                            then (PUTASSOC RGB INDEX CACHE))
                        INDEX])
```

(\DSPFONT.C150

```
[LAMBDA (C150STREAM FONT)                                   (* hdj " 4-Oct-85 11:55")
```

(* sets the font that a display stream uses to print characters. C150STREAM is guaranteed to be a stream of type C150)

```
  (PROG (XFONT OLDFONT (DD (fetch IMAGEDATA of C150STREAM)))
```

(* save old value to return, smash new value and update the bitchar portion of the record.)

```
        (RETURN (PROG1 (SETQ OLDFONT (fetch DDFONT of DD))
                       [COND
                          (FONT (SETQ XFONT (OR (\GETFONTDESC FONT (QUOTE C150)
                                                              T)
                                                (FONTCOPY (ffetch DDFONT of DD)
                                                          FONT)))   (* color case, create a font with the current foreground and
                                                                    background colors.)
                                                                    (* (SETQ XFONT (\GETCOLORFONT XFONT
                                                                    (DSPCOLOR NIL C150STREAM)
                                                                    (DSPBACKCOLOR NIL C150STREAM)
                                                                    (ffetch (BITMAP BITMAPBITSPERPIXEL) of
                                                                    (ffetch (\DISPLAYDATA DDDestination) of DD)))))
                                                                    (* updating font information is fairly expensive operation.
                                                                    Don't bother unless font has changed.)
                                (OR (EQ XFONT OLDFONT)
                                    (UNINTERRUPTABLY
                                        (freplace DDFONT of DD with XFONT)
                                        (freplace DDLINEFEED of DD with (IMINUS (fetch \SFHeight of XFONT)))
                                        (\SFFixFont C150STREAM DD))])])
```

(\DSPLEFTMARGIN.C150

```
[LAMBDA (C150STREAM XPOSITION)                              (* hdj " 5-Jun-85 12:56")
                                                            (* sets the xposition that a carriage return returns to.)
  (PROG ((DD (fetch IMAGEDATA of C150STREAM)))
        (RETURN (PROG1 (ffetch DDLeftMargin of DD)
                       [AND XPOSITION (COND
                                          ((AND (SMALLP XPOSITION)
                                                (IGREATERP XPOSITION -1))
                                           (UNINTERRUPTABLY
                                               (freplace DDLeftMargin of DD with XPOSITION)
                                               (\SFFIXLINELENGTH C150STREAM)))
                                          (T (\ILLEGAL.ARG XPOSITION])])
```

(\DSPLINEFEED.C150

```
[LAMBDA (C150STREAM DELTAY)                                 (* hdj " 5-Jun-85 12:56")
                                                            (* sets the amount that a line feed increases the y coordinate
                                                            by.)
  (PROG ((DD (fetch IMAGEDATA of C150STREAM)))
        (RETURN (PROG1 (ffetch DDLINEFEED of DD)
                       [AND DELTAY (COND
                                       ((NUMBERP DELTAY)
                                        (freplace DDLINEFEED of DD with DELTAY))
                                       (T (\ILLEGAL.ARG DELTAY])])
```

(\DSPOPERATION.C150

```
[LAMBDA (C150STREAM OPERATION)                              (* hdj " 5-Jun-85 12:56")
                                                            (* sets the operation field of a display stream)
  (PROG ((DD (\GETDISPLAYDATA C150STREAM)))
        (RETURN (PROG1 (fetch DDOPERATION of DD)
                       [COND
                          (OPERATION (OR (FMEMB OPERATION (QUOTE (PAINT REPLACE INVERT ERASE)))
                                         (LISPERROR "ILLEGAL ARG" OPERATION))
                                 (UNINTERRUPTABLY
                                     (freplace DDOPERATION of DD with OPERATION)
                                                            (* update other fields that depend on operation.)
                                     (\SETPBTFUNCTION (fetch DDPILOTBBT of DD)
                                            (fetch DDSOURCETYPE of DD)
                                            OPERATION))])])
```

(\DSPPRINTCHAR.C150

```
[LAMBDA (STREAM CHARCODE)                                   (* hdj " 5-Jun-85 12:56")
```

(* Displays the character and increments the Xposition. STREAM is guaranteed to be of type display.)

```
  (PROG ((DD (fetch IMAGEDATA of STREAM)))
        (SELCHARQ CHARCODE
             ((EOL CR LF)
```

```
                    (\DSPPRINTCR/LF.C150 CHARCODE STREAM)
                    (replace CHARPOSITION of STREAM with 0))
               (LF (\DSPPRINTCR/LF.C150 CHARCODE STREAM))
               (TAB (PROG (TABWIDTH (SPACEWIDTH (CHARWIDTH (CHARCODE SPACE)
                                                           STREAM)))
                          (SETQ TABWIDTH (UNFOLD SPACEWIDTH 8))
                          (if (IGREATERP (\DISPLAYSTREAMINCRXPOSITION (SETQ TABWIDTH
                                                                      (IDIFFERENCE TABWIDTH
                                                                             (MOD (IDIFFERENCE (fetch DDXPOSITION
                                                                                                  of DD)
                                                                                               (ffetch DDLeftMargin
                                                                                                  of DD))
                                                                                  TABWIDTH)))
                                                                      DD)
                                                 (ffetch DDRightMargin of DD))
                              then                                          (* tab was past rightmargin, force cr.)
                                   (\DSPPRINTCR/LF.C150 (CHARCODE EOL)
                                                        STREAM))            (* return the number of spaces taken.)
                          (add (fetch CHARPOSITION of STREAM)
                               (IQUOTIENT TABWIDTH SPACEWIDTH))))
               (add (fetch CHARPOSITION of STREAM)
                    (IPLUS (if (ILESSP CHARCODE 32)
                               then                                         (* CONTROL character)
                                    (\BLTCHAR.C150 CHARCODE STREAM DD)
                                    0
                               else (\BLTCHAR.C150 CHARCODE STREAM DD)
                                    1])
```

(\DSPPRINTCR/LF.C150

```
[LAMBDA (CHARCODE DS)                                                   (* hdj " 6-Jun-85 14:08")
                                                                        (* CHARCODE is EOL, CR, or LF Assumes that DS has been
                                                                        checked by \DSPPRINTCHAR)
  (PROG (BTM AMOUNT/BELOW Y ROTATION FONT (DD (fetch IMAGEDATA of DS)))
        (COND
           ((AND (fetch DDSlowPrintingCase of DD)
                 (NEQ (SETQ ROTATION (fetch (FONTDESCRIPTOR ROTATION) of (fetch DDFONT of DD)))
                      0))
            (PROG ((CLIPREG (ffetch DDClippingRegion of DD))
                   X)
                  [COND
                     ((EQ CHARCODE (CHARCODE EOL))                      (* on LF, no change in X)
                      (COND
                         ((SETQ Y (fetch DDEOLFN of DD))                (* call the eol function for ds.)
                          (APPLY* Y DS)))
                      (\DSPYPOSITION.C150 DS (SELECTQ ROTATION
                                                 (90 (fetch (REGION BOTTOM) of CLIPREG))
                                                 (270 (fetch (REGION TOP) of CLIPREG))
                                                 (ERROR "Only rotations supported are 0, 90 and 270"]
                  [SETQ X (IPLUS (fetch DDXPOSITION of DD)
                                 (SELECTQ ROTATION
                                      (90 (IMINUS (ffetch DDLINEFEED of DD)))
                                      (270 (ffetch DDLINEFEED of DD))
                                      (ERROR "Only rotations supported are 0, 90 and 270"]
                  (DSPXPOSITION X DS)))
           (T (COND
                 ((EQ CHARCODE (CHARCODE EOL))                          (* on LF, no change in X)
                  (COND
                     ((SETQ Y (fetch DDEOLFN of DD))                    (* call the eol function for ds.)
                      (APPLY* Y DS)))
                  (DSPXPOSITION (ffetch DDLeftMargin of DD)
                         DS)))
              (SETQ Y (IPLUS (ffetch DDYPOSITION of DD)
                             (ffetch DDLINEFEED of DD)))
              (DSPYPOSITION Y DS])
```

(\DSPRESET.C150

```
[LAMBDA (C150STREAM)                                                    (* hdj " 5-Aug-85 18:57")
  (DECLARE (GLOBALVARS \CURRENTDISPLAYLINE))                            (* resets a display stream)
  (PROG (CREG FONT FONTASCENT (DD (\GETDISPLAYDATA C150STREAM)))
        (SETQ CREG (ffetch DDClippingRegion of DD))
        (SETQ FONT (fetch DDFONT of DD))
        (SETQ FONTASCENT (FONTASCENT FONT))
        (SELECTQ (fetch (FONTDESCRIPTOR ROTATION) of FONT)
            (0 (\DSPXPOSITION.C150 C150STREAM (ffetch DDLeftMargin of DD))
               (\DSPYPOSITION.C150 C150STREAM (ADD1 (IDIFFERENCE (fetch TOP of CREG)
                                                                 FONTASCENT))))
            (90 (\DSPXPOSITION.C150 C150STREAM (IPLUS (fetch LEFT of CREG)
                                                      FONTASCENT))
                (\DSPYPOSITION.C150 C150STREAM (fetch BOTTOM of CREG)))
            (270 (\DSPXPOSITION.C150 C150STREAM (IDIFFERENCE (fetch RIGHT of CREG)
                                                             FONTASCENT))
                 (\DSPYPOSITION.C150 C150STREAM (fetch TOP of CREG)))
            (ERROR "only supported rotations are 0, 90 and 270"))
        (\CLEARBM (ffetch (\DISPLAYDATA DDDestination) of DD)
               (DSPBACKCOLOR NIL C150STREAM)
```

```
                CREG])
```

(\DSPRIGHTMARGIN.C150

```
[LAMBDA (C150STREAM XPOSITION)                                  (* hdj " 5-Jun-85 12:56")
                                                                (* Sets the right margin that determines when a cr is inserted by
  print.)
  (PROG (OLDRM (DD (fetch IMAGEDATA of C150STREAM)))
        (SETQ OLDRM (ffetch DDRightMargin of DD))
        (COND
           ((NULL XPOSITION))
           [(AND (SMALLP XPOSITION)
                 (IGREATERP XPOSITION -1))                      (* Avoid fixing linelength if right margin hasn't changed.)
            (OR (EQ XPOSITION OLDRM)
                (UNINTERRUPTABLY
                    (freplace DDRightMargin of DD with XPOSITION)
                    (\SFFIXLINELENGTH C150STREAM))]
           (T (\ILLEGAL.ARG XPOSITION)))
        (RETURN OLDRM])
```

(\DSPXPOSITION.C150

```
[LAMBDA (C150STREAM XPOSITION)                                  (* hdj " 5-Jun-85 12:56")
                                                                (* coordinate position is stored in 15 bits in the range -2^15 to
                                                                +2^15.)
  (PROG ((DD (fetch IMAGEDATA of C150STREAM)))
        (RETURN (PROG1 (fetch DDXPOSITION of DD)
                       (COND
                          ((NULL XPOSITION))
                          ((NUMBERP XPOSITION)
                           (freplace DDXPOSITION of DD with XPOSITION)
                                                                (* reset the charposition field so that PRINT etc.
                                                                won't put out eols.)
                           (freplace (STREAM CHARPOSITION) of C150STREAM with 0))
                          (T (\ILLEGAL.ARG XPOSITION))))])
```

(\DSPYPOSITION.C150

```
[LAMBDA (DISPLAYSTREAM YPOSITION)                               (* hdj " 3-Oct-85 17:57")
  (LET ((DD (fetch IMAGEDATA of DISPLAYSTREAM)))
       (PROG1 (ffetch DDYPOSITION of DD)
           (COND
              ((NULL YPOSITION))
              ((NUMBERP YPOSITION)
               (UNINTERRUPTABLY
                    (freplace DDYPOSITION of DD with YPOSITION)
                    (\INVALIDATEDISPLAYCACHE DD)))
              (T (\ILLEGAL.ARG YPOSITION))))])
```

(\DUMPPAGEBUFFER.C150

```
[LAMBDA (BITMAP C150STREAM COLOR.TABLES)                        (* gbn "13-Jan-86 21:37")
  (CENTRONICS.RESET C150STREAM)
  (LET*
   [(BACKINGSTREAM (\C150BackingStream C150STREAM))
    (MAXX (SUB1 (BITMAPWIDTH BITMAP)))
    (MAXY (SUB1 (BITMAPHEIGHT BITMAP)))
    (LINEBYTES (FOLDHI (BITMAPWIDTH BITMAP)
                       BITSPERBYTE))
    (PrintingTimeInSeconds 1)
    (PrintingTimer (SETUPTIMER PrintingTimeInSeconds NIL (QUOTE SECONDS]
   (C150.SETMARGINS BACKINGSTREAM)
   (C150.SEPARATOR BACKINGSTREAM)
   (bind (BLANKLINES _ 0)
         (FIRSTLINE _ T) for SCANLINE from MAXY to 0 by -4
      do
      (if (\C150.ALLWHITESPACE BITMAP COLOR.TABLES SCANLINE)
          then (add BLANKLINES 1)
               (BLOCK)
        else

        (* * First dump the buffered microlinefeeds)

        (if (AND FIRSTLINE C150.CLIPBUFFER)
            then

        (* don't bother printing these microlinefeeds, since they are just the blanks at the top of the buffer)

                 (SETQ FIRSTLINE NIL)
          else (for I to BLANKLINES do (\C150.MICROLINEFEED BACKINGSTREAM)))
        (SETQ BLANKLINES 0)
        [for SUBSCAN from 0 to 3
           do (if (TIMEREXPIRED? PrintingTimer (QUOTE SECONDS))
                  then (BLOCK)
                       (SETUPTIMER PrintingTimeInSeconds PrintingTimer (QUOTE SECONDS)))
              (for COLOR from 0 to 3
                 do                                             (* loop over (black magenta yellow cyan))
```

```
                    (LET [(COLOR.ARRAY.BASE (fetch (ARRAYP BASE) of (ELT COLOR.TABLES COLOR]
                         (\C150.SENDLINEINFO BACKINGSTREAM COLOR LINEBYTES SUBSCAN)
                         (for XPOSITION from 0 to MAXX by 8
                            do (BOUT BACKINGSTREAM (for BIT from 0 to 7
                                                      sum (LLSH (\GETBASE COLOR.ARRAY.BASE
                                                                   (BITMAPBIT BITMAP (IPLUS XPOSITION BIT
                                                                                            )
                                                                         (IDIFFERENCE SCANLINE SUBSCAN))
                                                                   )
                                                                (IDIFFERENCE 7 BIT]
       (\C150.MICROLINEFEED BACKINGSTREAM))
     finally (if (NOT C150.CLIPBUFFER)
                 then                                          (* print out the remaining microlinefeeds)
                      (for I from 1 to BLANKLINES do (\C150.MICROLINEFEED BACKINGSTREAM])
```

(\FILLCIRCLE.C150

```
[LAMBDA (C150STREAM CENTERX CENTERY RADIUS TEXTURE)          (* hdj " 6-Jun-85 16:17")
  (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 (fetch IMAGEDATA
                                                                                        of C150STREAM))
                           (X 0)
                           (Y RADIUS)
                           (D (ITIMES 2 (IDIFFERENCE 1 RADIUS)))
                           DESTINATIONBASE RASTERWIDTH CX CY TEXTUREBM GRAYHEIGHT GRAYWIDTH GRAYBASE NBITS
                           (FCBBT \BRUSHBBT))
                     (SETQ TOP (SUB1 (fetch DDClippingTop of DISPLAYDATA)))
                     (SETQ BOTTOM (fetch DDClippingBottom of DISPLAYDATA))
                     (SETQ LEFT (fetch DDClippingLeft of DISPLAYDATA))
                     (SETQ RIGHT (SUB1 (fetch DDClippingRight of DISPLAYDATA)))
                     (SETQ OPERATION (ffetch DDOPERATION of DISPLAYDATA))
                     (SETQ DestinationBitMap (fetch DDDestination of DISPLAYDATA))
                     (SETQ NBITS (fetch (BITMAP BITMAPBITSPERPIXEL) of DestinationBitMap))
                     [SETQ TEXTUREBM (COND
                                        ((BITMAPP TEXTURE))
                                        [(AND (NEQ NBITS 1)
                                              (BITMAPP (COLORTEXTUREFROMCOLOR# (COLORNUMBERP
                                                                                  (OR TEXTURE (DSPCOLOR NIL
                                                                                                        C150STREAM]
                                        [(AND (NULL TEXTURE)
                                              (BITMAPP (ffetch DDTexture of DISPLAYDATA]
                                        ([OR (FIXP TEXTURE)
                                             (AND (NULL TEXTURE)
                                                  (SETQ TEXTURE (ffetch 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 BITMAPBASE of DestinationBitMap))
                     (SETQ RASTERWIDTH (fetch BITMAPRASTERWIDTH of DestinationBitMap))
                                                              (* update as many fields in the brush bitblt table as possible
                                                              from DS.)
                     (replace PBTFLAGS of FCBBT with 0)
                     (replace PBTDESTBPL of FCBBT with (UNFOLD RASTERWIDTH BITSPERWORD))
                                                              (* clear gray information. PBTSOURCEBPL is used for gray
                                                              information too.)
                     (replace PBTSOURCEBPL of FCBBT with 0)
                     (replace PBTUSEGRAY of FCBBT with T)
                     [replace PBTGRAYWIDTHLESSONE of FCBBT with (SUB1 (SETQ GRAYWIDTH (IMIN (fetch (BITMAP
                                                                                                  BITMAPWIDTH
                                                                                                        )
                                                                                              of TEXTUREBM)
                                                                                           16]
                     [replace PBTGRAYHEIGHTLESSONE of FCBBT with (SUB1 (SETQ GRAYHEIGHT
                                                                          (IMIN (fetch (BITMAP BITMAPHEIGHT)
                                                                                   of TEXTUREBM)
                                                                                16]
                     (replace PBTDISJOINT of FCBBT with T)
                     (\SETPBTFUNCTION FCBBT (QUOTE TEXTURE)
                           OPERATION)
                     (replace PBTHEIGHT of FCBBT with 1)      (* take into account the brush thickness.)
```

```
                    (SETQ CX CENTERX)
                    (SETQ CY CENTERY)                          (* change Y TOP and BOTTOM to be in bitmap coordinates)
                    (SETQ CY (\SFInvert DestinationBitMap CY))
                    [SETQ BOTTOM (PROG1 (SUB1 (\SFInvert DestinationBitMap TOP))
                                        (SETQ TOP (SUB1 (\SFInvert DestinationBitMap BOTTOM))))]
                    (COND
                       ((EQ RADIUS 0)                          (* put a single point down. Use \LINEBLT to get proper texture.
                                                               NIL)
                        (\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.)
                        (\LINEBLT FCBBT (IDIFFERENCE CX X)
                                  CY
                                  (IPLUS CX X)
                                  DESTINATIONBASE RASTERWIDTH LEFT RIGHT BOTTOM TOP GRAYWIDTH GRAYHEIGHT
                                  GRAYBASE NBITS))
                       (T (\FILLCIRCLEBLT CX CY X Y)
                          (SETQ Y (SUB1 Y))
                          (GO LP)))
                    (MOVETO CENTERX CENTERY C150STREAM)
                    (RETURN NIL])
```

(**OUTCHARFN.C150**

```
  [LAMBDA (C150STREAM CHARCODE)                                (* hdj "10-Jun-85 15:14")
    (SELCHARQ CHARCODE
         (EOL                                                  (* New Line)
              (NEWLINE.C150 C150STREAM)
              (replace (STREAM CHARPOSITION) of C150STREAM with 0))
         (LF                                                   (* Line feed--move down, but not over)
             (\DSPXPOSITION.C150 C150STREAM (PROG1 (\DSPXPOSITION.C150 C150STREAM)
                                                   (NEWLINE.C150 C150STREAM))))
         (^L                                                   (* Form Feed)
             (replace (STREAM CHARPOSITION) of C150STREAM with 0)
             (NEWPAGE.C150 C150STREAM))
         (\BOUT C150STREAM CHARCODE])
```

(**SEARCHC150FONTFILES**

```
  [LAMBDA (FAMILY SIZE FACE ROTATION)                          (* hdj " 5-Jun-85 14:19")

          (* * returns a list of the fonts that can be read in for the C150 device.
          Rotation is ignored because it is assumed that all devices support 0 90 and 270)

    (DECLARE (GLOBALVARS DISPLAYFONTEXTENSIONS DISPLAYFONTDIRECTORIES))
    (SELECTQ (SYSTEMTYPE)
         (D (for E FILENAMEPATTERN FONTSFOUND THISFONT inside DISPLAYFONTEXTENSIONS
               do (SETQ FILENAMEPATTERN (\FONTFILENAME FAMILY SIZE FACE E))
                  [for DIR inside DISPLAYFONTDIRECTORIES
                     do (for FONTFILE in (DIRECTORY (PACKFILENAME (QUOTE DIRECTORY)
                                                                  DIR
                                                                  (QUOTE BODY)
                                                                  FILENAMEPATTERN))
                           do (OR (MEMBER (SETQ THISFONT (\FONTINFOFROMFILENAME FONTFILE (QUOTE DISPLAY)))
                                          FONTSFOUND)
                                  (SETQ FONTSFOUND (CONS THISFONT FONTSFOUND]
               finally (RETURN FONTSFOUND)))
         (SHOULDNT])
```

(**STRINGWIDTH.C150**

```
  [LAMBDA (C150STREAM STR RDTBL)                               (* hdj " 5-Jun-85 12:56")
```

```
                                                                    (* Returns the width of for the current font/spacefactor in
                                                                    STREAM.)
    (PROG (WIDTHSBASE)
          (RETURN (\STRINGWIDTH.GENERIC STR (SETQ WIDTHSBASE (ffetch (\DISPLAYDATA DDWIDTHSCACHE)
                                                                    of (ffetch IMAGEDATA of C150STREAM)))
                        RDTBL
                        (\FGETWIDTH WIDTHSBASE (CHARCODE SPACE])

)

(RPAQQ MISSINGC150FONTCOERCIONS (((GACHA)
                                  (MODERN))
                                 ((TIMESROMAN)
                                  (MODERN))
                                 ((HELVETICA)
                                  (MODERN))))

(RPAQQ \C150COLORTABLE NIL)

(RPAQQ \C150.FRAMEBUFFER NIL)

(RPAQQ \C150STREAM NIL)

(RPAQ C150COLORMAP (READARRAY 16 (QUOTE POINTER)
                          0))


((0 0 0)
 (0 0 255)
 (0 255 0)
 (255 0 0)
 (255 255 0)
 (255 0 255)
 (0 255 255)
 (255 255 255)
 (0 0 0)
 (0 0 255)
 (0 255 0)
 (255 0 0)
 (255 255 0)
 (255 0 255)
 (0 255 255)
 (255 255 255)
 NIL)

(RPAQQ C150FONTCOERCIONS
       (((CLASSIC 8)
         (CLASSIC 10))
        ((MODERN 8)
         (MODERN 10))
        ((MODERN 24)
         (MODERN 18))
        ((MODERN 18)
         (CLASSIC 18))
        ((CLASSIC 24)
         (CLASSIC 18))
        ((CLASSIC 12)
         (CLASSIC 14))))

(RPAQQ C150FONTDIRECTORIES ({ERIS}<LISPCORE>LIBRARY>))

(RPAQQ C150FONTEXTENSIONS (C150FONT))

(RPAQ? C150.CLIPBUFFER T)

(RPAQ? \C150DEFAULTDEVICE (QUOTE CENTRONICS))

(DEFINEQ

(COLORMAP.TO.C150TABLE
  [LAMBDA (COLORMAP)                                                (* hdj " 3-Aug-85 21:36")
    (LET* ((SIZE (ARRAYSIZE COLORMAP))
           (TABLETABLE (ARRAY 4 (QUOTE POINTER)
                              NIL 0))
           (BLACKTABLE (ARRAY SIZE (QUOTE SMALLP)
                              0 0))
           (CYANTABLE (ARRAY SIZE (QUOTE SMALLP)
                             0 0))
           (MAGENTATABLE (ARRAY SIZE (QUOTE SMALLP)
                                0 0))
           (YELLOWTABLE (ARRAY SIZE (QUOTE SMALLP)
                               0 0)))
          (bind CYAN MAGENTA YELLOW for PIXELVAL from 0 to (SUB1 SIZE)
             do [SETQ CYAN (SETA CYANTABLE PIXELVAL (IDIFFERENCE 1 (IQUOTIENT (fetch (RGB RED)
                                                                                 of (COLORMAPENTRY COLORMAP
                                                                                                   PIXELVAL))
```

```
                [SETQ MAGENTA (SETA MAGENTATABLE PIXELVAL (IDIFFERENCE 1 (IQUOTIENT (fetch (RGB GREEN)
                                                                                       of (COLORMAPENTRY
                                                                                                 COLORMAP
                                                                                                 PIXELVAL))
                                                                                    128]
                [SETQ YELLOW (SETA YELLOWTABLE PIXELVAL (IDIFFERENCE 1 (IQUOTIENT (fetch (RGB BLUE)
                                                                                         of (COLORMAPENTRY COLORMAP
                                                                                                   PIXELVAL))
                                                                                  128]
                (if (AND (EQ CYAN 1)
                         (EQ MAGENTA 1)
                         (EQ YELLOW 1))
                    then (SETA CYANTABLE PIXELVAL 0)
                         (SETA MAGENTATABLE PIXELVAL 0)
                         (SETA YELLOWTABLE PIXELVAL 0)
                         (SETA BLACKTABLE PIXELVAL 1)))
          (SETA TABLETABLE 0 BLACKTABLE)
          (SETA TABLETABLE 1 MAGENTATABLE)
          (SETA TABLETABLE 2 YELLOWTABLE)
          (SETA TABLETABLE 3 CYANTABLE)
          TABLETABLE])

)

(FILESLOAD COLOR XXGEOM XXFILL)

(IF (NOT (GETD (QUOTE POLYSHADE.BLT)))
    THEN                                                    (* A fix for KOTO, which is not necessary in <lc>n>)
         (MOVD (QUOTE POLYSHADE.DISPLAY)
               (QUOTE POLYSHADE.BLT)))

(DECLARE: DONTEVAL@LOAD DOCOPY

(\C150INIT)

(FILESLOAD CENTRONICS)
)

(DECLARE: EVAL@LOAD DONTCOPY

(FILESLOAD (LOADFROM)
       ADISPLAY LLDISPLAY)
)

(DECLARE: EVAL@COMPILE


(DEFMACRO \C150BackingStream (C150STREAM)
   (BQUOTE (fetch (STREAM F1) of , C150STREAM)))
)

(PUTPROPS C150STREAM COPYRIGHT ("Xerox Corporation" 1985 1986))
```

FUNCTION INDEX

VARIABLE INDEX

CONSTANT INDEX

MACRO INDEX