File created: 26-Oct-2021 10:53:47 **{DSK}<home>larry>medley>library>LLCOLOR.;2**

changes to: (FNS \COLORDISPLAYBITS \DRAW8BPPCOLORLINE)

previous date: 10-Jul-92 14:57:14 {DSK}<home>larry>medley>library>LLCOLOR.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1982-1992 by Xerox Corporation.

(RPAQQ LLCOLORCOMS
       [(FNS COLORDISPLAY COLORMAPBITS \CreateColorScreenBitMap \CREATECOLORDISPLAYFDEV COLORMAP COLORMAPCOPY
             SCREENCOLORMAP SCREENCOLORMAPENTRY ROTATECOLORMAP RGBCOLORMAP CMYCOLORMAP GRAYCOLORMAP
             COLORSCREENBITMAP \COLORDISPLAYBITS COLORSCREEN SHOWCOLORTESTPATTERN)
        (INITVARS (COLORMONITORTYPE 'CONRAC))
        (FNS \STARTCOLOR \STOPCOLOR \SENDCOLORMAPENTRY)
        (FNS COLORMAPCREATE COLORLEVEL COLORNUMBERP COLORFROMRGB INTENSITIESFROMCOLORMAP SETCOLORINTENSITY)
        (FNS \FAST8BIT \MAP4 \MAP8)
        (FNS \GETCOLORBRUSH)
        (FNS \DRAWCOLORLINE1 \DRAW4BPPCOLORLINE \DRAW8BPPCOLORLINE \DRAW24BPPCOLORLINE)
        (DECLARE%: DONTCOPY DOEVAL@COMPILE (MACROS .DRAW4BPPLINEX. .DRAW8BPPLINEX .DRAW24BPPLINEX
                                                  .DRAW4BPPLINEY. .DRAW8BPPLINEY .DRAW24BPPLINEY)
               (FILES (LOADCOMP)
                      MAIKOCOLOR))
        (FNS \BWTOCOLORBLT \4BITLINEBLT \8BITLINEBLT \24BITLINEBLT \GETBASE24 \PUTBASE24 COLORTEXTUREFROMCOLOR#
             \BITMAPWORD)
        (FNS COLORIZEBITMAP UNCOLORIZEBITMAP)
        (INITVARS (\1COLORMENU NIL)
               (\4COLORMENU NIL)
               (\8COLORMENU NIL))
        (FNS COLORMENU CURSORCOLOR)
        (RECORDS RGB HLS)
        (DECLARE%: DONTCOPY (RECORDS NIBBLES ONEOFFSETBITACCESS TWOOFFSETBITACCESS THREEOFFSETBTACCESS
                                     2BITNIBBLES ODD2BITNIBBLES))
        (DECLARE%: EVAL@COMPILE DONTCOPY (FILES (LOADCOMP)
                                                MAIKOCOLOR))
        (CONSTANTS (BITSPERWORD 16))
        (INITVARS (\COLORDISPLAYFDEV)
               (\4COLORMAP (CMYCOLORMAP 2 1 1 4))
               (\8COLORMAP (CMYCOLORMAP 3 3 2 8))
               (\COLORDISPLAYBITS)
               (ColorScreenBitMap)
               (\COLORSCREEN))
        (FNS PSEUDOCOLOR \PSEUDOCOLOR.BITMAP \PSEUDOCOLOR.UFN)
        (GLOBALVARS \COLORDISPLAYFDEV \COLORDISPLAYBITS ColorScreenBitMap \4COLORMAP \8COLORMAP)
        (P ;; NOTE: This is very bad.  I shouldn't have to and don't really want to do the following, but since about March 86, someone did something
           ;; really nonstandard wrt Helvetica fonts so that the in core versions are not equal to what is stored on file.  The SETFONTDESCRIPTOR
           ;; and friends undoes this kludge which has never been explained to LISPCORE^ by the person who brain damaged Helvetica this way.
           ;; If I don't undo this kludge by someone else, then color menus come out wrong.  *
           (SETFONTDESCRIPTOR 'HELVETICA 10 'MRR 0 'DISPLAY NIL)
           (SETQ MENUFONT (FONTCREATE 'HELVETICA 10)))
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                            (NLAML)
                                                                            (LAMA])

(DEFINEQ

(COLORDISPLAY
  [LAMBDA (ONOFF TYPE)                                          ; Edited 28-Apr-89 21:23 by takeshi
                                                                (* Turn hardware TYPE color display on or off.
                                                                *)
    (PROG (OLDONOFF OLDTYPE DISPLAYSTATE DISPLAYINFO)
          [COND
             ((\COLORDISPLAYFDEV (SETQ DISPLAYSTATE (fetch (FDEV DEVICEINFO) of \COLORDISPLAYFDEV))
                    (SETQ DISPLAYINFO (fetch (FDEV WINDOWDATA) of \COLORDISPLAYFDEV))
                    (SETQ OLDONOFF (fetch (DISPLAYSTATE ONOFF) of DISPLAYSTATE))
                    (SETQ OLDTYPE (fetch (DISPLAYINFO DITYPE) of DISPLAYINFO))
                    (COND
                       ((NULL TYPE)
                        (SETQ TYPE OLDTYPE]
          [COND
             ((EQ ONOFF 'ON)
              (COND
                 ((EQ OLDONOFF 'ON)                             (* Color display already on. *)
                  (COLORDISPLAY 'OFF)
                  (COLORDISPLAY 'ON TYPE))
                 ((OR (NULL \COLORDISPLAYFDEV)
                      (NOT (EQ TYPE OLDTYPE)))
                  (SETQ \COLORDISPLAYFDEV (\CREATECOLORDISPLAYFDEV TYPE))
                                                                (* Color display is off, turn it on. *)
```

```
                (\STARTCOLOR \COLORDISPLAYFDEV))
               (T (\STARTCOLOR \COLORDISPLAYFDEV)))
         (SCREENCOLORMAP (SCREENCOLORMAP))
         (COND
            ((OR (NULL \COLORSCREEN)
                 (NOT (EQ TYPE OLDTYPE)))
             (SETQ \COLORSCREEN (CREATESCREEN (COLORSCREENBITMAP)))
             (WINDOWWORLD 'ON \COLORSCREEN)
```

(* Besides being a test pattern, SHOWCOLORTESTPATTERN changes a solid field of color into a striped pattern.
Some color cards have trouble holding a solid field of color without variation steady.
*)

```
             (SHOWCOLORTESTPATTERN 10)))
         (SETQ BACKGROUNDCURSOREXITFN 'CURSOREXIT))
        ((EQ ONOFF 'OFF)
         (COND
            ((NOT (EQ OLDONOFF 'OFF))                    (* Color display is on, turn it off. *)
             (SETQ BACKGROUNDCURSOREXITFN NIL)
             [COND
                ((NOT (EQ \CURSORSCREEN \MAINSCREEN))    (* Move cursor off \COLORSCREEN.
                                                         *)
                 (CURSORSCREEN \MAINSCREEN (IQUOTIENT (fetch (SCREEN SCWIDTH) of \MAINSCREEN)
                                                      2)
                       (IQUOTIENT (fetch (SCREEN SCHEIGHT) of \MAINSCREEN)
                             2]
             (\STOPCOLOR \COLORDISPLAYFDEV]
   (RETURN OLDONOFF])
```

(COLORMAPBITS

```
[LAMBDA (COLORMAP)                                       (* kbr%: " 5-Jun-85 20:47")
  (INTEGERLENGTH (SUB1 (ARRAYSIZE COLORMAP])
```

(\CreateColorScreenBitMap

```
[LAMBDA (FDEV)                                           ; Edited 16-Jan-87 17:17 by gbn
                                                         (* Creates color display bitmap ColorScreenBitMap for FDEV)
  (DECLARE (GLOBALVARS ColorScreenBitMap))
  (PROG (DISPLAYINFO WIDTH HEIGHT BITSPERPIXEL)
        (SETQ DISPLAYINFO (fetch (FDEV WINDOWDATA) of FDEV))
        (SETQ WIDTH (fetch (DISPLAYINFO DIWIDTH) of DISPLAYINFO))
        (SETQ HEIGHT (fetch (DISPLAYINFO DIHEIGHT) of DISPLAYINFO))
        (SETQ BITSPERPIXEL (fetch (DISPLAYINFO DIBITSPERPIXEL) of DISPLAYINFO))
        (SETQ ColorScreenBitMap (create BITMAP
                                        BITMAPBASE _ (\COLORDISPLAYBITS WIDTH HEIGHT BITSPERPIXEL)
                                        BITMAPRASTERWIDTH _ (FOLDHI (ITIMES WIDTH BITSPERPIXEL)
                                                                    BITSPERWORD)
                                        BITMAPWIDTH _ WIDTH
                                        BITMAPHEIGHT _ HEIGHT
                                        BITMAPBITSPERPIXEL _ BITSPERPIXEL))
        (RETURN ColorScreenBitMap])
```

(\CREATECOLORDISPLAYFDEV

```
[LAMBDA (TYPE)                                           (* kbr%: "15-Feb-86 14:48")
  (PROG (DISPLAYINFO WSOPS)
        (SETQ DISPLAYINFO (ASSOC TYPE \DISPLAYINFOALIST))
        (SETQ WSOPS (fetch (DISPLAYINFO DIWSOPS) of DISPLAYINFO))
        (COND
           ((NULL DISPLAYINFO)
            (RETURN \COLORDISPLAYFDEV)))
        [COND
           ((NULL \COLORDISPLAYFDEV)
            (SETQ \COLORDISPLAYFDEV (\CREATEDISPLAY 'COLORDISPLAY]
        (replace (FDEV WINDOWDATA) of \COLORDISPLAYFDEV with DISPLAYINFO)
        (replace (FDEV EVENTFN) of \COLORDISPLAYFDEV with (fetch (WSOPS EVENTFN) of WSOPS))
        (replace (FDEV WINDOWOPS) of \COLORDISPLAYFDEV with WSOPS)
        (\CreateColorScreenBitMap \COLORDISPLAYFDEV)
        (RETURN \COLORDISPLAYFDEV])
```

(COLORMAP

```
[LAMBDA (BITSPERPIXEL NEWCOLORMAP)                       (* kbr%: "21-Aug-85 21:06")
                                                         (* Change system colormap to NEWCOLORMAP returning
                                                         OLDCOLORMAP *)
  (PROG (OLDCOLORMAP)
        (SETQ OLDCOLORMAP (SELECTQ BITSPERPIXEL
                                   (4 \4COLORMAP)
                                   (8 \8COLORMAP)
                                   NIL))
        [COND
           (NEWCOLORMAP (SELECTQ BITSPERPIXEL
                                 (4 (SETQ \4COLORMAP NEWCOLORMAP))
                                 (8 (SETQ \8COLORMAP NEWCOLORMAP))
                                 NIL)
                        (COND
```

```
                    ((AND \COLORDISPLAYFDEV (EQ (fetch (DISPLAYSTATE ONOFF) of (fetch (FDEV DEVICEINFO)
                                                                                  of \COLORDISPLAYFDEV))
                                                'ON)
                          (EQ (BITSPERPIXEL (COLORSCREENBITMAP))
                              BITSPERPIXEL))
                     (for I from 0 to (SUB1 (ARRAYSIZE NEWCOLORMAP)) do (\SENDCOLORMAPENTRY
                                                                           \COLORDISPLAYFDEV I
                                                                           (ELT NEWCOLORMAP I]
    (RETURN OLDCOLORMAP])
```

(COLORMAPCOPY

```
  [LAMBDA (COLORMAP BITSPERPIXEL)                                    (* rrb "21-OCT-82 18:32")
```

(* makes a copy of a color map If COLORMAP is not a color map, it returns a new color map with default values.
If the colormaps are different sizes, the first 16 entries will be the same and the rest will be black)

```
    (COLORMAPCREATE (AND (COLORMAPP COLORMAP BITSPERPIXEL)
                         (INTENSITIESFROMCOLORMAP COLORMAP))
     BITSPERPIXEL])
```

(SCREENCOLORMAP

```
  [LAMBDA (NEWCOLORMAP)                                              (* kbr%: "21-Aug-85 21:12")
    (COLORMAP (BITSPERPIXEL (COLORSCREENBITMAP))
     NEWCOLORMAP])
```

(SCREENCOLORMAPENTRY

```
  [LAMBDA (COLOR RGB)                                                (* kbr%: " 5-Jun-86 19:40")
    (SETA (SCREENCOLORMAP)
          COLOR RGB)
    (\SENDCOLORMAPENTRY \COLORDISPLAYFDEV COLOR RGB])
```

(ROTATECOLORMAP

```
  [LAMBDA (STARTCOLOR THRUCOLOR)                                     (* kbr%: " 5-Jun-86 23:20")
    (PROG (COLORMAP RGB)
          (SETQ COLORMAP (SCREENCOLORMAP))
          (COND
             ((NULL STARTCOLOR)
              (SETQ STARTCOLOR 0)))
          [COND
             ((NULL THRUCOLOR)
              (SETQ THRUCOLOR (SUB1 (ARRAYSIZE COLORMAP]
          (SETQ RGB (ELT COLORMAP THRUCOLOR))
          (for COLOR from STARTCOLOR to THRUCOLOR do (swap RGB (ELT COLORMAP COLOR))
                                                    (\SENDCOLORMAPENTRY \COLORDISPLAYFDEV COLOR
                                                           (ELT COLORMAP COLOR])
```

(RGBCOLORMAP

```
  [LAMBDA (REDBITS GREENBITS BLUEBITS BITSPERPIXEL)                  (* kbr%: "13-Aug-85 16:49")
```

(* creates a color map with the specified number of bits allocated per primary color.
Always has the RED bits on the left.)

```
    (PROG (NRED NGREEN NBLUE REDS GREENS BLUES COLORMAP)
          (SETQ NRED (SUB1 (EXPT 2 REDBITS)))
          (SETQ NGREEN (SUB1 (EXPT 2 GREENBITS)))
          (SETQ NBLUE (SUB1 (EXPT 2 BLUEBITS)))
          [SETQ REDS (for I from 0 to NRED collect (FIXR (FQUOTIENT (ITIMES 255 I)
                                                                    NRED]
          [SETQ GREENS (for I from 0 to NGREEN collect (FIXR (FQUOTIENT (ITIMES 255 I)
                                                                        NGREEN]
          [SETQ BLUES (for I from 0 to NBLUE collect (FIXR (FQUOTIENT (ITIMES 255 I)
                                                                      NBLUE]
          (SETQ COLORMAP
           (COLORMAPCREATE [for I from 1 to (EXPT 2 (IDIFFERENCE BITSPERPIXEL (IPLUS REDBITS GREENBITS BLUEBITS
                                                                                  )))
                           join (for RED in REDS
                                   join (for GREEN in GREENS
                                           join (for BLUE in BLUES
                                                   collect (create RGB
                                                                   RED _ RED
                                                                   GREEN _ GREEN
                                                                   BLUE _ BLUE]
                  BITSPERPIXEL))
          (RETURN COLORMAP])
```

(CMYCOLORMAP

```
  [LAMBDA (CYANBITS MAGENTABITS YELLOWBITS BITSPERPIXEL)             (* kbr%: "13-Aug-85 16:46")
    (PROG (COLORMAP MAXCOLOR)
          (SETQ COLORMAP (RGBCOLORMAP CYANBITS MAGENTABITS YELLOWBITS BITSPERPIXEL))
          (SETQ MAXCOLOR (SUB1 (ARRAYSIZE COLORMAP)))
          [for I from 0 to (IQUOTIENT MAXCOLOR 2) do (swap (ELT COLORMAP I)
```

```
                                                    (ELT COLORMAP (IDIFFERENCE MAXCOLOR I]
        (RETURN COLORMAP])
```

(GRAYCOLORMAP

```
  [LAMBDA (BITSPERPIXEL)                                    (* kbr%: "11-Jul-85 19:20")
                                                            (* creates a gray color map *)
    (PROG (MAXCOLOR GRAYS COLORMAP)
          (SETQ MAXCOLOR (MAXIMUMCOLOR BITSPERPIXEL))
          [SETQ GRAYS (for I from MAXCOLOR to 0 by -1 collect (FIXR (FQUOTIENT (ITIMES 255 I)
                                                                               MAXCOLOR]
          (SETQ COLORMAP (COLORMAPCREATE (for GRAY in GRAYS
                                            collect (create RGB
                                                            RED _ GRAY
                                                            GREEN _ GRAY
                                                            BLUE _ GRAY))
                                 BITSPERPIXEL))
          (RETURN COLORMAP])
```

(COLORSCREENBITMAP

```
  [LAMBDA NIL                                               (* rrb "22-OCT-82 14:01")
                                                            (* returns the color screen bitmap)
    ColorScreenBitMap])
```

(\COLORDISPLAYBITS

```
  [LAMBDA (WIDTH HEIGHT BITSPERPIXEL)                       ; Edited 26-Oct-2021 10:24 by larry
                                                            ; Edited 31-Oct-89 10:25 by takeshi
                                                            (* returns a pointer to the bits that the color board needs.)
    (DECLARE (GLOBALVARS \COLORDISPLAYBITS))
    (COND
       [(AND (EQ (MACHINETYPE)
                 'MAIKO)
             (OR (\MAIKO.CGSIXP)
                 (\MAIKO.CGTHREEP)
                 (\MAIKO.CGFOURP)))
        (PROG ((DUMMY (\ALLOCPAGEBLOCK 1))
               (ADDROFFSET (SUBRCALL COLOR-BASE)))
              (WHILE (NEQ (LOGAND \MAIKO.COLORBUF.ALIGN (IPLUS (\LOLOC DUMMY)
                                                               ADDROFFSET))
                          0)
                 DO (SETQ DUMMY (\ALLOCPAGEBLOCK 1)))
              (RETURN (OR (SETQ \COLORDISPLAYBITS (\ALLOCPAGEBLOCK \MAIKO.COLORPAGES))
                          (ERROR "No room for color screen of size" \MAIKO.COLORPAGES]
       (T (PROG (NPAGES)
```

(* TBW%: If you come through this function a second time with different screen params won't you get screwed half the time?
*)

```
                [COND
                   ((NULL \COLORDISPLAYBITS)                (* 2 extra pages needed for DORADOCOLOR microcode bug.
                                                            *)
                    (SETQ NPAGES (IPLUS (FOLDHI (ITIMES (FOLDHI (ITIMES WIDTH BITSPERPIXEL)
                                                                        BITSPERWORD)
                                                                HEIGHT)
                                                        WORDSPERPAGE)
                                        2))                 (* \ALLOCBLOCK can't hack bitmaps of the size of the 1132
                                                            color screen)
                    (SETQ \COLORDISPLAYBITS (COND
                                               ((IGREATERP (UNFOLD NPAGES CELLSPERPAGE)
                                                           \MaxArrayNCells)
                                                (OR (\ALLOCPAGEBLOCK NPAGES)
                                                    (ERROR "No room for color screen of size" NPAGES)))
                                               (T (\ALLOCBLOCK (UNFOLD NPAGES CELLSPERPAGE)
                                                               NIL NIL CELLSPERPAGE]
                (RETURN \COLORDISPLAYBITS])
```

(COLORSCREEN

```
  [LAMBDA NIL                                               (* kbr%: " 2-Feb-86 15:02")
    \COLORSCREEN])
```

(SHOWCOLORTESTPATTERN

```
  [LAMBDA (SIZE)                                            (* kbr%: "15-Feb-86 15:16")
```

(* Put a color test pattern on the color display. SIZE is the size of the stripes that will be put up.
*)

```
    (PROG (DESTINATION WIDTH HEIGHT BITSPERPIXEL COLORS NCOLORS)
          (OR (NUMBERP SIZE)
              (SETQ SIZE 10))
          (SETQ DESTINATION (COLORSCREENBITMAP))
          (SETQ WIDTH (BITMAPWIDTH DESTINATION))
          (SETQ HEIGHT (BITMAPHEIGHT DESTINATION))
          (SETQ BITSPERPIXEL (BITSPERPIXEL DESTINATION))
```

```
          (BLTSHADE MINIMUMSHADE DESTINATION)
          (SETQ COLORS (for BUCKET in COLORNAMES collect (CAR BUCKET)))
          (SETQ NCOLORS (LENGTH COLORS))
          (for COLOR from 0 as LEFT from 10 by 80 to WIDTH
             do (BLTSHADE [CAR (NTH COLORS (ADD1 (IMOD COLOR NCOLORS]
                       DESTINATION LEFT 410 60 60))
          (for COLOR from 1 as LEFT from 10 by 80 to WIDTH
             do (BLTSHADE [CAR (NTH COLORS (ADD1 (IMOD COLOR NCOLORS]
                       DESTINATION LEFT 330 60 60))
          (for HORIZCOLOR from 0 as BOTTOM from 0 to 300 by SIZE
             do (BLTSHADE [CAR (NTH COLORS (ADD1 (IMOD HORIZCOLOR NCOLORS]
                       DESTINATION 0 BOTTOM WIDTH SIZE 'REPLACE)
             finally (for VERTCOLOR from 0 as LEFT from 0 to WIDTH by (ITIMES SIZE 2)
                      do (BLTSHADE [CAR (NTH COLORS (ADD1 (IMOD VERTCOLOR NCOLORS]
                                DESTINATION LEFT 0 SIZE BOTTOM 'REPLACE])

)

(RPAQ? COLORMONITORTYPE 'CONRAC)

(DEFINEQ

(\STARTCOLOR
  [LAMBDA (FDEV)                                          (* kbr%: " 1-Jul-85 13:41")
    (WSOP 'STARTCOLOR FDEV])

(\STOPCOLOR
  [LAMBDA (FDEV)                                          (* kbr%: " 1-Jul-85 13:40")
    (WSOP 'STOPCOLOR FDEV])

(\SENDCOLORMAPENTRY
  [LAMBDA (FDEV COLOR# RGB)                               (* kbr%: " 1-Jul-85 19:43")
                                                          (* changes the window world background to SHADE)
    (WSOP 'SENDCOLORMAPENTRY FDEV COLOR# RGB])

)

(DEFINEQ

(COLORMAPCREATE
  [LAMBDA (INTENSITIES BITSPERPIXEL)                      ; Edited 16-Jan-87 17:36 by gbn
    (PROG (COLORMAP)
          (SELECTQ BITSPERPIXEL
              (4 [COND
                    ((NULL INTENSITIES)
                     (SETQ COLORMAP (CMYCOLORMAP 2 1 1 BITSPERPIXEL)))
                    (T (SETQ COLORMAP (ARRAY 16 NIL NIL 0))
                       (for COLOR from 0 to 15 as RGB in INTENSITIES do (SETA COLORMAP COLOR RGB])
              (8 [COND
                    ((NULL INTENSITIES)
                     (SETQ COLORMAP (CMYCOLORMAP 3 3 2 BITSPERPIXEL)))
                    (T (SETQ COLORMAP (ARRAY 256 NIL NIL 0))
                       (for COLOR from 0 to 255 as RGB in INTENSITIES do (SETA COLORMAP COLOR RGB])
              (24 (SETQ COLORMAP NIL))
              (\ILLEGAL.ARG BITSPERPIXEL))
          (RETURN COLORMAP])

(COLORLEVEL
  [LAMBDA (COLOR PRIMARY NEWLEVEL)                        (* kbr%: " 5-Jun-86 19:58")
    (PROG (RGB OLDVALUE)
          (SETQ RGB (ELT (SCREENCOLORMAP)
                         COLOR))
          (SETQ OLDVALUE (SELECTQ PRIMARY
                              (RED (fetch (RGB RED) of RGB))
                              (GREEN (fetch (RGB GREEN) of RGB))
                              (BLUE (fetch (RGB BLUE) of RGB))
                              (\ILLEGAL.ARG PRIMARY)))
          (COND
             (NEWLEVEL (SELECTQ PRIMARY
                             (RED (replace (RGB RED) of RGB with NEWLEVEL))
                             (GREEN (replace (RGB GREEN) of RGB with NEWLEVEL))
                             (BLUE (replace (RGB BLUE) of RGB with NEWLEVEL))
                             (SHOULDNT))
                       (\SENDCOLORMAPENTRY \COLORDISPLAYFDEV COLOR RGB)))
          (RETURN OLDVALUE])

(COLORNUMBERP
  [LAMBDA (COLOR# BITSPERPIXEL NOERRFLG)                  (* kbr%: "21-Aug-85 21:22")
                                                          (* returns the color number from a color.)
    (PROG (RGB)
          (COND
             [(FIXP COLOR#)
              (RETURN (COND
```

```
                        ((AND (IGEQ COLOR# 0)
                              (ILEQ COLOR# (MAXIMUMCOLOR BITSPERPIXEL))
                              COLOR#))
                        (NOERRFLG NIL)
                        (T (\ILLEGAL.ARG COLOR#]
               [(LITATOM COLOR#)
                (RETURN (COND
                        ((SETQ RGB (\LOOKUPCOLORNAME COLOR#))   (* recursively look up color number)
                         (COLORNUMBERP (CDR RGB)
                                BITSPERPIXEL NOERRFLG))
                        (NOERRFLG NIL)
                        (T (ERROR "Unknown color name" COLOR#]
               ((HLSP COLOR#)                                   (* HLS form convert to RGB)
                (SETQ RGB (HLSTORGB COLOR#)))
               ((RGBP COLOR#)                                   (* check for RGB or HLS)
                (SETQ RGB COLOR#))
               (NOERRFLG (RETURN NIL))
               (T (\ILLEGAL.ARG COLOR#)))
            (RETURN (COND
                      ((COLORFROMRGB RGB BITSPERPIXEL))
                      (NOERRFLG NIL)
                      (T (ERROR COLOR# "not available in color map"])
```

(COLORFROMRGB

```
  [LAMBDA (RGB BITSPERPIXEL)                                    (* kbr%: "15-Feb-86 11:16")
                                                                (* looks in the colormap for a color that has RGB levels)
    (PROG (COLOR COLORMAP)
          (COND
             ((EQ BITSPERPIXEL 24)                              (* Assuming subtractive system in which white=0.
                                                                *)
              [SETQ COLOR (LOGOR (LLSH (IDIFFERENCE 255 (fetch (RGB RED) of RGB))
                                       16)
                                 (LLSH (IDIFFERENCE 255 (fetch (RGB GREEN) of RGB))
                                       8)
                                 (IDIFFERENCE 255 (fetch (RGB BLUE) of RGB]
              (RETURN COLOR)))
          (SETQ COLORMAP (COLORMAP BITSPERPIXEL))
          (SETQ COLOR (for COLOR from 0 to (SUB1 (ARRAYSIZE COLORMAP)) thereis (EQUAL (ELT COLORMAP COLOR)
                                                                                      RGB)))
          (RETURN COLOR])
```

(INTENSITIESFROMCOLORMAP

```
  [LAMBDA (COLORMAP)                                            (* kbr%: "21-Aug-85 21:17")

          (* returns the intensity levels of the primary colors from a colormap.
          This list can be passed into COLORMAPCREATE to get an equivalent colormap.)

    (for I from 0 to (SUB1 (ARRAYSIZE COLORMAP)) collect (ELT COLORMAP I])
```

(SETCOLORINTENSITY

```
  [LAMBDA (COLORMAP COLOR# INTENSITIES)                         (* rrb "13-DEC-82 13:15")

          (* sets the intensity levels of a color number in a color map. Does not return the previous setting.)

    (PROG (RGB)
          (SETQ RGB INTENSITIES)
      LP  (COND
             [(NULL RGB)
              (SETQ RGB '(0 0 0]
             ((RGBP RGB))
             ((HLSP RGB)
              (SETQ RGB (HLSTORGB RGB)))
             ((SETQ RGB (CDR (\LOOKUPCOLORNAME RGB)))
              (GO LP))
             (T (\ILLEGAL.ARG RGB)))
          (COLORLEVEL COLORMAP COLOR# 'RED (fetch (RGB RED) of RGB))
          (COLORLEVEL COLORMAP COLOR# 'GREEN (fetch (RGB GREEN) of RGB))
          (COLORLEVEL COLORMAP COLOR# 'BLUE (fetch (RGB BLUE) of RGB])

)

(DEFINEQ
```

(\FAST8BIT

```
  [LAMBDA (A B N MAP)                                           (* edited%: "10-SEP-82 16:14")
    (bind AW (I _ 0) for J from 0 do (SETQ AW (\ADDBASE A J))
                                     (OR (IGREATERP N I)
                                         (RETURN))
                                     (\PUTBASE B I (ELT MAP (fetch (2BITNIBBLES EN1) of AW)))
                                     (OR (IGREATERP N (add I 1))
                                         (RETURN))
                                     (\PUTBASE B I (ELT MAP (fetch (2BITNIBBLES EN2) of AW)))
                                     (OR (IGREATERP N (add I 1))
                                         (RETURN))
```

```
                                    (\PUTBASE B I (ELT MAP (fetch (2BITNIBBLES EN3) of AW)))
                                    (OR (IGREATERP N (add I 1))
                                        (RETURN))
                                    (\PUTBASE B I (ELT MAP (fetch (2BITNIBBLES EN4) of AW)))
                                    (OR (IGREATERP N (add I 1))
                                        (RETURN))
                                    (\PUTBASE B I (ELT MAP (fetch (2BITNIBBLES EN5) of AW)))
                                    (OR (IGREATERP N (add I 1))
                                        (RETURN))
                                    (\PUTBASE B I (ELT MAP (fetch (2BITNIBBLES EN6) of AW)))
                                    (OR (IGREATERP N (add I 1))
                                        (RETURN))
                                    (\PUTBASE B I (ELT MAP (fetch (2BITNIBBLES EN7) of AW)))
                                    (OR (IGREATERP N (add I 1))
                                        (RETURN))
                                    (\PUTBASE B I (ELT MAP (fetch (2BITNIBBLES EN8) of AW)))
                                    (add I 1])
```

(\MAP4

```
  [LAMBDA (0C 1C)                                                    (* edited%: "10-SEP-82 15:50")
    (SETQ 0C (COND
                (0C (COLORNUMBERP 0C 4))
                (T 0)))                                              (* Mask out but 4 bits)
    (SETQ 1C (COND
                (1C (COLORNUMBERP 1C 4))
                (T 15)))
    (PROG (MAP)
          (SETQ MAP (ARRAY 16 'SMALLPOSP 0 0))
          [for I from 0 to 15 do (SETA MAP I (for J from 0 to 3
                                                sum (LLSH (COND
                                                             ((ZEROP (LOGAND I (LLSH 1 J)))
                                                              0C)
                                                             (T 1C))
                                                          (ITIMES J 4]
          (RETURN MAP])
```

(\MAP8

```
  [LAMBDA (0C 1C)                                                    (* edited%: "10-SEP-82 15:50")

          (* returns an array of words that contain the destination bitmap should contain if a black and white bitmap is blown up to an
          8 bit per pixel bitmap.)

    (SETQ 0C (COND
                (0C (COLORNUMBERP 0C 8))
                (T 0)))                                              (* make sure color numbers are given.)
    (SETQ 1C (COND
                (1C (COLORNUMBERP 1C 8))
                (T 255)))
    (PROG (MAP)
          (SETQ MAP (ARRAY 4 'SMALLPOSP 0 0))
          [for I from 0 to 3 do (SETA MAP I (LOGOR (COND
                                                      ((ZEROP (LOGAND I 1))
                                                       0C)
                                                      (T 1C))
                                                   (LLSH (COND
                                                            ((ZEROP (LOGAND I 2))
                                                             0C)
                                                            (T 1C))
                                                         8]
          (RETURN MAP])

)

(DEFINEQ
```

(\GETCOLORBRUSH

```
  [LAMBDA (BRUSH COLOR NBITS)                                        (* rrb "21-DEC-82 20:46")
                                                                     (* produces a colorbitmap that is 1's where ever the brush
                                                                     bitmap would be 1)
    (COND
       ((AND (BITMAPP BRUSH)
             (EQ (fetch (BITMAP BITMAPBITSPERPIXEL) of BRUSH)
                 NBITS))
        BRUSH)
       (T (COLORIZEBITMAP [COND
                             ((LISTP BRUSH)
                              (\BRUSHBITMAP (fetch (BRUSH BRUSHSHAPE) of BRUSH)
                                     (fetch (BRUSH BRUSHSIZE) of BRUSH)))
                             (T (\BRUSHBITMAP 'ROUND (OR BRUSH 1]
                0 COLOR NBITS])

)

(DEFINEQ
```

(\DRAWCOLORLINE1

```
  [LAMBDA (X0 Y0 XLIMIT YLIMIT DX DY CDL YINC MODE BITMAPBASE RASTERWIDTH NBITS COLOR)
                                                                    ; Edited 21-Aug-91 12:15 by jds
    (DECLARE (LOCALVARS . T))
    [COND
       ((EQ MODE 'ERASE)                                            ; treat erase as AND of background
        (SETQ COLOR (OPPOSITECOLOR COLOR NBITS]
    (COND
       ((EQ NBITS 4)
        (\DRAW4BPPCOLORLINE X0 Y0 XLIMIT YLIMIT DX DY CDL YINC MODE BITMAPBASE RASTERWIDTH COLOR))
       (T (\DRAW8BPPCOLORLINE X0 Y0 XLIMIT YLIMIT DX DY CDL YINC MODE BITMAPBASE RASTERWIDTH COLOR])
```

(\DRAW4BPPCOLORLINE

```
  [LAMBDA (X0 Y0 XLIMIT YLIMIT DX DY CDL YINC MODE BITMAPBASE RASTERWIDTH COLOR)
                                                                    ; Edited 21-Aug-91 12:12 by jds
    (DECLARE (LOCALVARS . T))

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

    (PROG (MAPPTR MASK COLORMASK COLORMASKORG WORDOFFSET)
          (SETQ COLORMASKORG (LLSH COLOR 12))

     ;; keep word offset from bitmapbase so that the YINC can be negative or positive.  Used to use \ADDBASE directly but negative case was not in
     ;; micro code and ran much slower.

          [SETQ WORDOFFSET (IPLUS (ITIMES Y0 RASTERWIDTH)
                                  (FOLDLO X0 (CONSTANT (LRSH BITSPERWORD 2]
          (SETQ MAPPTR (\ADDBASE BITMAPBASE WORDOFFSET))
          (SETQ MASK (\4BITMASK X0))
          (SETQ COLORMASK (LLSH COLOR (LLSH (IDIFFERENCE 3 (LOGAND X0 3))
                                            2)))
          (SETQ X0 0)
          (SETQ Y0 0)
          (COND
             ((IGEQ DX DY)                                          ; X is the fastest mover.
              (.DRAW4BPPLINEX. MODE))
             (T                                                     ; Y is the fastest mover.
                (.DRAW4BPPLINEY. MODE])
```

(\DRAW8BPPCOLORLINE

```
  [LAMBDA (X0 Y0 XLIMIT YLIMIT DX DY CDL YINC MODE BITMAPBASE RASTERWIDTH COLOR)
                                                                    ; Edited 26-Oct-2021 10:25 by larry
                                                                    ; Edited 19-Mar-91 12:46 by matsuda
    (SUBRCALL COLOR-8BPPDRAWLINE X0 Y0 XLIMIT YLIMIT DX DY CDL YINC MODE BITMAPBASE RASTERWIDTH COLOR])
```

(\DRAW24BPPCOLORLINE

```
  [LAMBDA (X0 Y0 XLIMIT YLIMIT DX DY CDL YINC MODE BITMAPBASE RASTERWIDTH COLOR)
                                                                    (* kbr%: "15-Feb-86 23:00")
    (DECLARE (LOCALVARS . T))

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

    (PROG (MAPPTR STARTBYTE WORDOFFSET)

          (* keep word offset from bitmapbase so that the YINC can be negative or positive.
          Used to use \ADDBASE directly but negative case was not in micro code and ran much slower.)

          [SETQ WORDOFFSET (IPLUS (ITIMES Y0 RASTERWIDTH)
                                  (FOLDLO X0 (CONSTANT (LRSH BITSPERWORD 3]
          (SETQ MAPPTR (\ADDBASE BITMAPBASE WORDOFFSET))
          (SETQ STARTBYTE (LOGAND X0 1))
          (SETQ X0 0)
          (SETQ Y0 0)
          (COND
             ((IGEQ DX DY)                                          (* X is the fastest mover.)
              (.DRAW24BPPLINEX MODE))
             (T                                                     (* Y is the fastest mover.)
                (.DRAW24BPPLINEY MODE])

)

(DECLARE%: DONTCOPY DOEVAL@COMPILE

(DECLARE%: EVAL@COMPILE

(PUTPROPS .DRAW4BPPLINEX. MACRO [(MODE)
                                 (PROG (INSIDEBITS OUTSIDEBITS)
                                       (until (IGREATERP X0 XLIMIT)
                                          do                        (* main loop)
                                              (SETQ INSIDEBITS (LOGAND MASK (fetch (BITMAPWORD BITS)
                                                                               of MAPPTR)))
```

```
                                        (SETQ OUTSIDEBITS (LOGAND (LOGNOT MASK)
                                                                  (fetch (BITMAPWORD BITS) of MAPPTR)))
                                        [replace (BITMAPWORD BITS) of MAPPTR
                                           with (SELECTQ MODE
                                                     (ERASE (LOGOR (LOGAND COLORMASK INSIDEBITS)
                                                                   OUTSIDEBITS))
                                                     (INVERT (LOGOR (LOGXOR COLORMASK INSIDEBITS)
                                                                    OUTSIDEBITS))
                                                     (PAINT (LOGOR (LOGOR COLORMASK INSIDEBITS)
                                                                   OUTSIDEBITS))
                                                     (PROGN
                                                           (* case is REPLACE. Legality of OPERATION has been
                                                           checked by \CLIPANDDRAWLINE1)
                                                             (LOGOR COLORMASK OUTSIDEBITS]
                                        [COND
                                           ([NOT (IGREATERP DX (SETQ CDL (IPLUS CDL DY]
                                            (COND
                                               ((IGREATERP (SETQ Y0 (ADD1 Y0))
                                                           YLIMIT)
                                                (RETURN)))
                                            (SETQ CDL (IDIFFERENCE CDL DX))
                                            (SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET
                                                                                    (IPLUS WORDOFFSET YINC]
                                        [COND
                                           [(ZEROP (SETQ MASK (LRSH MASK 4)))
                                                           (* crossed word boundary)
                                            [SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET
                                                                                    (ADD1 WORDOFFSET]
                                            (SETQ COLORMASK COLORMASKORG)
                                            (SETQ MASK (CONSTANT (\4BITMASK 0]
                                           (T (SETQ COLORMASK (LRSH COLORMASK 4]
                                        (SETQ X0 (ADD1 X0])

(PUTPROPS .DRAW8BPPLINEX MACRO ((MODE)
                                (PROG NIL
                                      (COND
                                         ((EQ STARTBYTE 1)
                                          (GO 1LP)))
                                  0LP                           (* main loop)
                                      (\PUTBASEBYTE MAPPTR 0 (SELECTQ MODE
                                                                 (ERASE (LOGAND COLOR (\GETBASEBYTE MAPPTR 0)
                                                                                ))
                                                                 (INVERT (LOGXOR COLOR (\GETBASEBYTE MAPPTR 0
                                                                                             )))
                                                                 (PAINT (LOGOR COLOR (\GETBASEBYTE MAPPTR 0))
)
                                                                 (PROGN
                                                           (* case is REPLACE. Legality of OPERATION has been
                                                           checked by \CLIPANDDRAWLINE1)
                                                                        COLOR)))
                                      [COND
                                         ([NOT (IGREATERP DX (SETQ CDL (IPLUS CDL DY]
                                          (COND
                                             ((IGREATERP (SETQ Y0 (ADD1 Y0))
                                                         YLIMIT)
                                              (RETURN)))
                                          (SETQ CDL (IDIFFERENCE CDL DX))
                                          (SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET (IPLUS WORDOFFSET
                                                                                                    YINC]
                                      (COND
                                         ((IGREATERP (SETQ X0 (ADD1 X0))
                                                     XLIMIT)
                                          (RETURN)))
                                  1LP (\PUTBASEBYTE MAPPTR 1 (SELECTQ MODE
                                                                 (ERASE (LOGAND COLOR (\GETBASEBYTE MAPPTR 1)
                                                                                ))
                                                                 (INVERT (LOGXOR COLOR (\GETBASEBYTE MAPPTR 1
                                                                                             )))
                                                                 (PAINT (LOGOR COLOR (\GETBASEBYTE MAPPTR 1))
)
                                                                 (PROGN
                                                           (* case is REPLACE. Legality of OPERATION has been
                                                           checked by \CLIPANDDRAWLINE1)
                                                                        COLOR)))
                                      [COND
                                         ([NOT (IGREATERP DX (SETQ CDL (IPLUS CDL DY]
                                          (COND
                                             ((IGREATERP (SETQ Y0 (ADD1 Y0))
                                                         YLIMIT)
                                              (RETURN)))
                                          (SETQ CDL (IDIFFERENCE CDL DX))
                                          (SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET (IPLUS WORDOFFSET
                                                                                                    YINC]
                                      (COND
                                         ((IGREATERP (SETQ X0 (ADD1 X0))
                                                     XLIMIT)
                                          (RETURN)))
```

```
                                    [SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET (ADD1 WORDOFFSET]
                                    (GO 0LP))))

(PUTPROPS .DRAW24BPPLINEX MACRO ((MODE)
                                  (PROG NIL                          (* main loop)
                                    LP  (\PUTBASE24 MAPPTR 0 (SELECTQ MODE
                                                                 (ERASE (LOGAND COLOR (\GETBASE24 MAPPTR 0)))
                                                                 (INVERT (LOGXOR COLOR (\GETBASE24 MAPPTR 0)))
                                                                 (PAINT (LOGOR COLOR (\GETBASE24 MAPPTR 0)))
                                                                 (PROGN
                                                                 (* case is REPLACE. Legality of OPERATION has been
                                                                 checked by \CLIPANDDRAWLINE1)
                                                                        COLOR)))
                                    [COND
                                       ([NOT (IGREATERP DX (SETQ CDL (IPLUS CDL DY]
                                        (COND
                                           ((IGREATERP (SETQ Y0 (ADD1 Y0))
                                                   YLIMIT)
                                            (RETURN)))
                                        (SETQ CDL (IDIFFERENCE CDL DX))
                                        (SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET (IPLUS
                                                                                                WORDOFFSET
                                                                                                YINC]
                                    (COND
                                       ((IGREATERP (SETQ X0 (ADD1 X0))
                                              XLIMIT)
                                        (RETURN)))
                                    [SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET (ADD1 WORDOFFSET]
                                    (GO LP))))

(PUTPROPS .DRAW4BPPLINEY. MACRO [(MODE)
                                  (PROG (INSIDEBITS OUTSIDEBITS)
                                        (until (IGREATERP Y0 YLIMIT)
                                           do                        (* main loop)
                                              (SETQ INSIDEBITS (LOGAND MASK (fetch (BITMAPWORD BITS)
                                                                                   of MAPPTR)))
                                              (SETQ OUTSIDEBITS (LOGAND (LOGNOT MASK)
                                                                        (fetch (BITMAPWORD BITS) of MAPPTR)))
                                              [replace (BITMAPWORD BITS) of MAPPTR
                                                 with (SELECTQ MODE
                                                              (ERASE (LOGOR (LOGAND COLORMASK INSIDEBITS)
                                                                            OUTSIDEBITS))
                                                              (INVERT (LOGOR (LOGXOR COLORMASK INSIDEBITS)
                                                                             OUTSIDEBITS))
                                                              (PAINT (LOGOR (LOGOR COLORMASK INSIDEBITS)
                                                                            OUTSIDEBITS))
                                                              (PROGN
                                                                    (* case is REPLACE. Legality of OPERATION has been
                                                                    checked by \CLIPANDDRAWLINE1)
                                                                      (LOGOR COLORMASK OUTSIDEBITS]
                                              [COND
                                                 ([NOT (IGREATERP DY (SETQ CDL (IPLUS CDL DX]
                                                  (COND
                                                     ((IGREATERP (SETQ X0 (ADD1 X0))
                                                             XLIMIT)
                                                      (RETURN)))
                                                  (SETQ CDL (IDIFFERENCE CDL DY))
                                                  (COND
                                                     [(ZEROP (SETQ MASK (LRSH MASK 4)))
                                                                    (* crossed word boundary)
                                                      [SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET
                                                                                              (ADD1 WORDOFFSET]
                                                      (SETQ COLORMASK COLORMASKORG)
                                                      (SETQ MASK (CONSTANT (\4BITMASK 0]
                                                     (T (SETQ COLORMASK (LRSH COLORMASK 4]
                                              [SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET
                                                                                      (IPLUS WORDOFFSET YINC]
                                              (SETQ Y0 (ADD1 Y0)])

(PUTPROPS .DRAW8BPPLINEY MACRO ((MODE)
                                 (PROG NIL
                                       (COND
                                          ((EQ STARTBYTE 1)
                                           (GO 1LP)))
                                   0LP                               (* main loop)
                                       (\PUTBASEBYTE MAPPTR 0 (SELECTQ MODE
                                                                    (ERASE (LOGAND COLOR (\GETBASEBYTE MAPPTR 0)
                                                                                   ))
                                                                    (INVERT (LOGXOR COLOR (\GETBASEBYTE MAPPTR 0
                                                                                                        )))
                                                                    (PAINT (LOGOR COLOR (\GETBASEBYTE MAPPTR 0))
)
                                                                    (PROGN
                                                                    (* case is REPLACE. Legality of OPERATION has been
                                                                    checked by \CLIPANDDRAWLINE1)
                                                                           COLOR)))
                                       (COND
```

```
                                        ((IGREATERP (SETQ Y0 (ADD1 Y0))
                                                YLIMIT)
                                         (RETURN)))
                                    [SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET (IPLUS WORDOFFSET
                                                                                              YINC]
                                    (COND
                                       ([NOT (IGREATERP DY (SETQ CDL (IPLUS CDL DX]
                                                                   (* moved enough in Y to move a point in X)
                                        (COND
                                           ((IGREATERP (SETQ X0 (ADD1 X0))
                                                   XLIMIT)
                                            (RETURN)))
                                        (SETQ CDL (IDIFFERENCE CDL DY))
                                        (GO 1LP)))
                                    (GO 0LP)
                                  1LP (\PUTBASEBYTE MAPPTR 1 (SELECTQ MODE
                                                                  (ERASE (LOGAND COLOR (\GETBASEBYTE MAPPTR 1)
                                                                                 ))
                                                                  (INVERT (LOGXOR COLOR (\GETBASEBYTE MAPPTR 1
                                                                                                      )))
                                                                  (PAINT (LOGOR COLOR (\GETBASEBYTE MAPPTR 1))
)
                                                                  (PROGN
                                                                  (* case is REPLACE. Legality of OPERATION has been
                                                                  checked by \CLIPANDDRAWLINE1)
                                                                         COLOR)))
                                    (COND
                                       ((IGREATERP (SETQ Y0 (ADD1 Y0))
                                                YLIMIT)
                                        (RETURN)))
                                    [SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET (IPLUS WORDOFFSET
                                                                                              YINC]
                                    (COND
                                       ([NOT (IGREATERP DY (SETQ CDL (IPLUS CDL DX]
                                                                   (* moved enough in Y to move a point in X)
                                        (COND
                                           ((IGREATERP (SETQ X0 (ADD1 X0))
                                                   XLIMIT)
                                            (RETURN)))
                                        (SETQ CDL (IDIFFERENCE CDL DY))
                                        [SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET (ADD1 WORDOFFSET]
                                        (GO 0LP)))
                                    (GO 1LP))))

(PUTPROPS .DRAW24BPPLINEY MACRO ((MODE)
                                 (PROG NIL
                                       (COND
                                          ((EQ STARTBYTE 1)
                                           (GO 1LP)))
                                    0LP                            (* main loop)
                                        (\PUTBASEBYTE MAPPTR 0 (SELECTQ MODE
                                                                  (ERASE (LOGAND COLOR (\GETBASEBYTE MAPPTR 0
                                                                                                     )))
                                                                  (INVERT (LOGXOR COLOR (\GETBASEBYTE MAPPTR
                                                                                                      0)))
                                                                  (PAINT (LOGOR COLOR (\GETBASEBYTE MAPPTR 0)
                                                                                ))
                                                                  (PROGN
                                                                  (* case is REPLACE. Legality of OPERATION has been
                                                                  checked by \CLIPANDDRAWLINE1)
                                                                         COLOR)))
                                       (COND
                                          ((IGREATERP (SETQ Y0 (ADD1 Y0))
                                                  YLIMIT)
                                           (RETURN)))
                                       [SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET (IPLUS WORDOFFSET
                                                                                                 YINC]
                                       (COND
                                          ([NOT (IGREATERP DY (SETQ CDL (IPLUS CDL DX]
                                                                      (* moved enough in Y to move a point in X)
                                           (COND
                                              ((IGREATERP (SETQ X0 (ADD1 X0))
                                                      XLIMIT)
                                               (RETURN)))
                                           (SETQ CDL (IDIFFERENCE CDL DY))
                                           (GO 1LP)))
                                       (GO 0LP)
                                     1LP (\PUTBASEBYTE MAPPTR 1 (SELECTQ MODE
                                                                  (ERASE (LOGAND COLOR (\GETBASEBYTE MAPPTR 1
                                                                                                     )))
                                                                  (INVERT (LOGXOR COLOR (\GETBASEBYTE MAPPTR
                                                                                                      1)))
                                                                  (PAINT (LOGOR COLOR (\GETBASEBYTE MAPPTR 1)
                                                                                ))
                                                                  (PROGN
                                                                  (* case is REPLACE. Legality of OPERATION has been
                                                                  checked by \CLIPANDDRAWLINE1)
```

```
                                                                    COLOR)))
                                          (COND
                                             ((IGREATERP (SETQ Y0 (ADD1 Y0))
                                                         YLIMIT)
                                              (RETURN)))
                                          [SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET (IPLUS WORDOFFSET
                                                                                                    YINC]
                                          (COND
                                             ([NOT (IGREATERP DY (SETQ CDL (IPLUS CDL DX]
                                                                    (* moved enough in Y to move a point in X)
                                              (COND
                                                 ((IGREATERP (SETQ X0 (ADD1 X0))
                                                             XLIMIT)
                                                  (RETURN)))
                                              (SETQ CDL (IDIFFERENCE CDL DY))
                                              [SETQ MAPPTR (\ADDBASE BITMAPBASE (SETQ WORDOFFSET (ADD1 WORDOFFSET
                                                                                                        ]
                                              (GO 0LP)))
                                          (GO 1LP))))
)

(FILESLOAD (LOADCOMP)
       MAIKOCOLOR)
)

(DEFINEQ
```

(**BWTOCOLORBLT**

```
  [LAMBDA (SOURCEBWBM SLEFT SBOTTOM DESTCOLORBM DLEFT DBOTTOM WIDTH HEIGHT 0COLOR 1COLOR DESTNBITS)
                                                                    (* kbr%: "15-Feb-86 11:06")

          (* blits from a black and white bitmap into a color bitmap which has DESTNBITS bits per pixel.
          DESTCOLORBM is a pointer to the color bitmap.)
                                                                    (* assumes all datatypes and bounds have been checked)
    (SELECTQ DESTNBITS
        (4 [PROG (MAP SRCBASE SRCHEIGHT SRCRW SRCWRD SRCOFFSET DESBASE DESHEIGHT DESRW DESWRD DESOFF NBITS
                      DESALIGNLEFT SCR)
                 (SETQ MAP (fetch (ARRAYP BASE) of (\MAP4 0COLOR 1COLOR)))
                 (SETQ SRCBASE (fetch (BITMAP BITMAPBASE) of SOURCEBWBM))
                 (SETQ SRCHEIGHT (fetch (BITMAP BITMAPHEIGHT) of SOURCEBWBM))
                 (SETQ SRCRW (fetch (BITMAP BITMAPRASTERWIDTH) of SOURCEBWBM))
                 (SETQ SRCWRD (FOLDLO SLEFT BITSPERWORD))
                 (SETQ SRCOFFSET (MOD SLEFT BITSPERWORD))
                 (SETQ DESBASE (fetch (BITMAP BITMAPBASE) of DESTCOLORBM))
                 (SETQ DESHEIGHT (fetch (BITMAP BITMAPHEIGHT) of DESTCOLORBM))
                 (SETQ DESRW (fetch (BITMAP BITMAPRASTERWIDTH) of DESTCOLORBM))
                 (SETQ DESWRD (FOLDLO DLEFT 4))
                 (SETQ DESOFF (MOD DLEFT 4))
                 (SETQ NBITS 4)                                     (* DESTCOLORBM is used to allow one bit per pixel bitblt
                                                                    operations on the bitmap.)
                 [COND
                    ((NOT (EQ 0 DESOFF))                            (* save the left bits of the destination bitmap so it can be word
                                                                    aligned.)
                     (SETQ SCR (BITMAPCREATE 4 HEIGHT 4))
                     (BITBLT DESTCOLORBM (SETQ DESALIGNLEFT (LLSH DESWRD 2))
                             DBOTTOM SCR 0 0 DESOFF HEIGHT 'INPUT 'REPLACE]
                 (for LINECOUNTER from 1 to HEIGHT
                    do

          (* linecounter goes from 1 to height because bitmaps are stored internally with top first so subtracting height is necessary to
          get offset of line and the 1 corrects for height difference.)

                      (\4BITLINEBLT (\ADDBASE SRCBASE (IPLUS (ITIMES (IDIFFERENCE SRCHEIGHT (IPLUS LINECOUNTER
                                                                                                   SBOTTOM))
                                                                                    SRCRW)
                                                                             SRCWRD))
                                    SRCOFFSET
                                    (\ADDBASE DESBASE (IPLUS (ITIMES (IDIFFERENCE DESHEIGHT (IPLUS LINECOUNTER
                                                                                                   DBOTTOM))
                                                                                     DESRW)
                                                                              DESWRD))
                                    WIDTH MAP 0COLOR 1COLOR))
                 (COND
                    (DESALIGNLEFT                                   (* move the color bits to the right and restore the saved color
                                                                    bits.)
                           (BITBLT DESTCOLORBM DESALIGNLEFT DBOTTOM DESTCOLORBM (IPLUS DESALIGNLEFT DESOFF)
                                  DBOTTOM WIDTH HEIGHT 'INPUT 'REPLACE)
                           (BITBLT SCR 0 0 DESTCOLORBM DESALIGNLEFT DBOTTOM DESOFF HEIGHT 'INPUT 'REPLACE]))
        (8 (PROG (MAP SRCBASE SRCHEIGHT SRCRW SRCWRD SRCOFFSET DESBASE DESHEIGHT DESRW DESWRD DESOFF)
                 (SETQ MAP (fetch (ARRAYP BASE) of (\MAP8 0COLOR 1COLOR)))
                 (SETQ SRCBASE (fetch (BITMAP BITMAPBASE) of SOURCEBWBM))
                 (SETQ SRCHEIGHT (fetch (BITMAP BITMAPHEIGHT) of SOURCEBWBM))
                 (SETQ SRCRW (fetch (BITMAP BITMAPRASTERWIDTH) of SOURCEBWBM))
                 (SETQ SRCWRD (FOLDLO SLEFT BITSPERWORD))
                 (SETQ SRCOFFSET (MOD SLEFT BITSPERWORD))
                 (SETQ DESBASE (fetch (BITMAP BITMAPBASE) of DESTCOLORBM))
                 (SETQ DESHEIGHT (fetch (BITMAP BITMAPHEIGHT) of DESTCOLORBM))
```

```
                (SETQ DESRW (fetch (BITMAP BITMAPRASTERWIDTH) of DESTCOLORBM))
                (SETQ DESWRD (FOLDLO DLEFT 2))
                (SETQ DESOFF (MOD DLEFT 2))
                (for LINECOUNTER from 1 to HEIGHT
                   do
```

(* linecounter goes from 1 to height because bitmaps are stored internally with top first so subtracting height is necessary to get offset of line and the 1 corrects for height difference.)

```
                      (\8BITLINEBLT (\ADDBASE SRCBASE (IPLUS (ITIMES (IDIFFERENCE SRCHEIGHT (IPLUS LINECOUNTER
                                                                                               SBOTTOM))
                                                                     SRCRW)
                                                              SRCWRD))
                                    SRCOFFSET
                                    (\ADDBASE DESBASE (IPLUS (ITIMES (IDIFFERENCE DESHEIGHT (IPLUS LINECOUNTER
                                                                                                  DBOTTOM))
                                                                     DESRW)
                                                              DESWRD))
                                    DESOFF WIDTH MAP 0COLOR 1COLOR))))
          (24 (PROG (SRCBASE SRCHEIGHT SRCRW DESBASE DESHEIGHT DESRW)
                    (SETQ SRCBASE (fetch (BITMAP BITMAPBASE) of SOURCEBWBM))
                    (SETQ SRCHEIGHT (fetch (BITMAP BITMAPHEIGHT) of SOURCEBWBM))
                    (SETQ SRCRW (fetch (BITMAP BITMAPRASTERWIDTH) of SOURCEBWBM))
                    (SETQ DESBASE (fetch (BITMAP BITMAPBASE) of DESTCOLORBM))
                    (SETQ DESHEIGHT (fetch (BITMAP BITMAPHEIGHT) of DESTCOLORBM))
                    (SETQ DESRW (fetch (BITMAP BITMAPRASTERWIDTH) of DESTCOLORBM))
                    (for LINECOUNTER from 1 to HEIGHT do
```

(* linecounter goes from 1 to height because bitmaps are stored internally with top first so subtracting height is necessary to get offset of line and the 1 corrects for height difference.)

```
                                                          (\24BITLINEBLT (\ADDBASE SRCBASE
                                                                                   (ITIMES (IDIFFERENCE SRCHEIGHT
                                                                                                        (IPLUS LINECOUNTER
                                                                                                               SBOTTOM))
                                                                                           SRCRW))
                                                                         SLEFT
                                                                         (\ADDBASE DESBASE (ITIMES (IDIFFERENCE DESHEIGHT
                                                                                                                (IPLUS LINECOUNTER
                                                                                                                       DBOTTOM))
                                                                                                   DESRW))
                                                                         DLEFT WIDTH 0COLOR 1COLOR))))
          (SHOULDNT])
```

(\4BITLINEBLT

```
  [LAMBDA (SBASE SBITOFFSET DBASE WIDTH MAPBASE 0COLOR 1COLOR)   (* rrb "15-OCT-82 09:28")
```

(* moves one line of a black and white bitmap into a color bitmap using a mapping table.
Destination bit offset is assumed to be 0 because \BWTOCOLORBLT arranges things so that it is.)

```
    (SELECTQ (MOD SBITOFFSET 4)
         (0                                                    (* case of moving even aligned bits.)
            [PROG NIL
              ONEWRDLP
                                                               (* SBITOFFSET is either 0, 4, 8 or 12)
                 (COND
                    ((AND (EQ SBITOFFSET 0)
                          (IGREATERP WIDTH (SUB1 BITSPERWORD))) (* go to center loop.)
                     (GO LP))
                    ((IGREATERP 4 WIDTH)
                     [PROG (SWORDCONTENTS)
                           (SETQ SWORDCONTENTS (\GETBASE SBASE 0))
                           (SELECTQ WIDTH
                                (0)
                                (1 (PUTBASEBYTE DBASE 0 (LOGOR (LOGAND (\GETBASEBYTE DBASE 0)
                                                                       15)
                                                               (LLSH (COND
                                                                        ((ZEROP (LOGAND SWORDCONTENTS
                                                                                        (\BITMASK SBITOFFSET)))
                                                                         0COLOR)
                                                                        (T 1COLOR))
                                                                     4))))
                                (2 [PUTBASEBYTE DBASE 0 (LOGOR (LLSH (COND
                                                                        ((ZEROP (LOGAND SWORDCONTENTS
                                                                                        (\BITMASK SBITOFFSET)))
                                                                         0COLOR)
                                                                        (T 1COLOR))
                                                                     4)
                                                               (COND
                                                                  ([ZEROP (LOGAND SWORDCONTENTS
                                                                                  (\BITMASK (ADD1 SBITOFFSET]
                                                                   0COLOR)
                                                                  (T 1COLOR])
                                (PROGN [\PUTBASEBYTE DBASE 0 (LOGOR (LLSH (COND
                                                                            ((ZEROP (LOGAND SWORDCONTENTS
                                                                                            (\BITMASK SBITOFFSET
```

```
                                                                                                    )))
                                                                                        0COLOR)
                                                                                       (T 1COLOR))
                                                                                  4)
                                                                             (COND
                                                                                ([ZEROP (LOGAND SWORDCONTENTS
                                                                                               (\BITMASK (ADD1 SBITOFFSET
                                                                                                                  ]
                                                                                 0COLOR)
                                                                                (T 1COLOR]
                                                  (\PUTBASEBYTE DBASE 1
                                                         (LOGOR (LLSH (COND
                                                                         ([ZEROP (LOGAND SWORDCONTENTS
                                                                                        (\BITMASK (IPLUS SBITOFFSET 2]
                                                                          0COLOR)
                                                                         (T 1COLOR))
                                                                    4)
                                                                (LOGAND (\GETBASE DBASE 0)
                                                                        15]
                    (RETURN))
                   (T                                                    (* move 4 bits from source to destination.)
                      [\PUTBASE DBASE 0 (\GETBASE MAPBASE (SELECTQ SBITOFFSET
                                                                  (0 (fetch (NIBBLES N1) of SBASE))
                                                                  (4 (fetch (NIBBLES N2) of SBASE))
                                                                  (8 (fetch (NIBBLES N3) of SBASE))
                                                                  (fetch (NIBBLES N4) of SBASE]
                      (SETQ DBASE (\ADDBASE DBASE 1))
                      (SETQ WIDTH (IDIFFERENCE WIDTH 4))
                      [COND
                         ((EQ (SETQ SBITOFFSET (IPLUS SBITOFFSET 4))
                              16)
                          (SETQ SBITOFFSET 0)
                          (SETQ SBASE (\ADDBASE SBASE 1]
                      (GO ONEWRDLP)))
          LP  (COND
                 ((IGREATERP WIDTH (SUB1 BITSPERWORD))                 (* move a source word's worth of bits.)
                  (\PUTBASE DBASE 0 (\GETBASE MAPBASE (fetch (NIBBLES N1) of SBASE)))
                  (\PUTBASE DBASE 1 (\GETBASE MAPBASE (fetch (NIBBLES N2) of SBASE)))
                  (\PUTBASE DBASE 2 (\GETBASE MAPBASE (fetch (NIBBLES N3) of SBASE)))
                  (\PUTBASE DBASE 3 (\GETBASE MAPBASE (fetch (NIBBLES N4) of SBASE)))
                  (SETQ DBASE (\ADDBASE DBASE 4))
                  (SETQ SBASE (\ADDBASE SBASE 1))
                  (SETQ WIDTH (IDIFFERENCE WIDTH BITSPERWORD))
                  (GO LP))
                 (T                                                    (* finish off last less than 16 bits.)
                    (GO ONEWRDLP])
     (1
       (* moving bits that are aligned with 1 extra bit in the following word of the source.)
         [PROG NIL
           ONEWRDLP
                                                                       (* SBITOFFSET is either 0, 4, 8 or 12)
              (COND
                 ((AND (EQ SBITOFFSET 1)
                       (IGREATERP WIDTH (SUB1 BITSPERWORD)))           (* go to center loop.)
                  (GO LP))
                 ((IGREATERP 4 WIDTH)
                  [PROG (SWORDCONTENTS)
                        (SETQ SWORDCONTENTS (\GETBASE SBASE 0))
                        (SELECTQ WIDTH
                             (0)
                             (1 (PUTBASEBYTE DBASE 0 (LOGOR (LLSH (COND
                                                                     ((ZEROP (LOGAND SWORDCONTENTS
                                                                                    (\BITMASK SBITOFFSET)))
                                                                      0COLOR)
                                                                     (T 1COLOR))
                                                                4)
                                                               (LOGAND (\GETBASEBYTE DBASE 0)
                                                                       15))))
                             (2 [PUTBASEBYTE DBASE 0 (LOGOR (LLSH (COND
                                                                     ((ZEROP (LOGAND SWORDCONTENTS
                                                                                    (\BITMASK SBITOFFSET)))
                                                                      0COLOR)
                                                                     (T 1COLOR))
                                                                4)
                                                               (COND
                                                                  ([ZEROP (LOGAND SWORDCONTENTS
                                                                                  (\BITMASK (ADD1 SBITOFFSET]
                                                                   0COLOR)
                                                                  (T 1COLOR])
                             (PROGN [\PUTBASEBYTE DBASE 0 (LOGOR (LLSH (COND
                                                                         ((ZEROP (LOGAND SWORDCONTENTS
                                                                                         (\BITMASK SBITOFFSET
                                                                                                   )))
                                                                          0COLOR)
                                                                         (T 1COLOR))
```

```
                                                                    4)
                                                                (COND
                                                                   ([ZEROP (LOGAND SWORDCONTENTS
                                                                                  (\BITMASK (ADD1 SBITOFFSET
                                                                                                          ]
                                                                    0COLOR)
                                                                   (T 1COLOR]
                                                (\PUTBASEBYTE DBASE 1
                                                       (LOGOR (LLSH (COND
                                                                       ([ZEROP (LOGAND SWORDCONTENTS
                                                                                      (\BITMASK (IPLUS SBITOFFSET 2]
                                                                        0COLOR)
                                                                       (T 1COLOR))
                                                                   4)
                                                              (LOGAND (\GETBASE DBASE 0)
                                                                      15]
                  (RETURN))
                 (T                                                    (* move 4 bits from source to destination.)
                    [\PUTBASE DBASE 0 (\GETBASE MAPBASE (SELECTQ SBITOFFSET
                                                                    (1 (fetch (ONEOFFSETBITACCESS BITS1TO4)
                                                                              of SBASE))
                                                                    (5 (fetch (ONEOFFSETBITACCESS BITS5TO8)
                                                                              of SBASE))
                                                                    (9 (fetch (ONEOFFSETBITACCESS BITS9TO12)
                                                                              of SBASE))
                                                                    (LOGOR (LLSH (fetch (ONEOFFSETBITACCESS
                                                                                          BITS13TO15)
                                                                                        of SBASE)
                                                                                 1)
                                                                           (fetch (ODD2BITNIBBLES BIT0)
                                                                                  of (SETQ SBASE (\ADDBASE SBASE 1]
                    (SETQ DBASE (\ADDBASE DBASE 1))
                    (SETQ WIDTH (IDIFFERENCE WIDTH 4))
                    (COND
                       ((EQ (SETQ SBITOFFSET (IPLUS SBITOFFSET 4))
                            17)                                      (* SBASE has already been incremented as part of fetching the
                        last 4 bits.)
                        (SETQ SBITOFFSET 1)))
                    (GO ONEWRDLP)))
      LP  (COND
             ((IGREATERP WIDTH (SUB1 BITSPERWORD))                   (* move a source word's worth of bits.)
              (\PUTBASE DBASE 0 (\GETBASE MAPBASE (fetch (ONEOFFSETBITACCESS BITS1TO4) of SBASE)))
              (\PUTBASE DBASE 1 (\GETBASE MAPBASE (fetch (ONEOFFSETBITACCESS BITS5TO8) of SBASE)))
              (\PUTBASE DBASE 2 (\GETBASE MAPBASE (fetch (ONEOFFSETBITACCESS BITS9TO12) of SBASE)))
              [\PUTBASE DBASE 3 (\GETBASE MAPBASE (LOGOR (LLSH (fetch (ONEOFFSETBITACCESS BITS13TO15)
                                                                       of SBASE)
                                                               1)
                                                         (fetch (ODD2BITNIBBLES BIT0)
                                                                of (SETQ SBASE (\ADDBASE SBASE 1]
              (SETQ DBASE (\ADDBASE DBASE 4))
              (SETQ WIDTH (IDIFFERENCE WIDTH BITSPERWORD))
              (GO LP))
             (T                                                      (* finish off last less than 16 bits.)
                (GO ONEWRDLP])
(2
  (* moving bits that are aligned with 2 extra bits in the following word of the source.)

    [PROG NIL
      ONEWRDLP
                                                                     (* SBITOFFSET is either 2, 6, 10 or 14)
          (COND
             ((AND (EQ SBITOFFSET 2)
                   (IGREATERP WIDTH (SUB1 BITSPERWORD)))             (* go to center loop.)
              (GO LP))
             ((IGREATERP 4 WIDTH)
              [PROG (SWORDCONTENTS)
                    (SETQ SWORDCONTENTS (\GETBASE SBASE 0))
                    (SELECTQ WIDTH
                         (0)
                         (1 (PUTBASEBYTE DBASE 0 (LOGOR (LLSH (COND
                                                                 ((ZEROP (LOGAND SWORDCONTENTS
                                                                                 (\BITMASK SBITOFFSET)))
                                                                  0COLOR)
                                                                 (T 1COLOR))
                                                              4)
                                                         (LOGAND (\GETBASEBYTE DBASE 0)
                                                                 15))))
                         (2 [PUTBASEBYTE DBASE 0 (LOGOR (LLSH (COND
                                                                 ((ZEROP (LOGAND SWORDCONTENTS
                                                                                 (\BITMASK SBITOFFSET)))
                                                                  0COLOR)
                                                                 (T 1COLOR))
                                                              4)
                                                         (COND
                                                            ([ZEROP (LOGAND SWORDCONTENTS
                                                                            (\BITMASK (ADD1 SBITOFFSET]
```

```
                                                    0COLOR)
                                                   (T 1COLOR])
                            (PROGN                                  (* first two bits are always in this word.)
                                   [\PUTBASEBYTE DBASE 0 (LOGOR (LLSH (COND
                                                                      ((ZEROP (LOGAND SWORDCONTENTS
                                                                                      (\BITMASK SBITOFFSET
                                                                                                ]
                                                                       0COLOR)
                                                                      (T 1COLOR))
                                                                  4)
                                                              (COND
                                                                 ([ZEROP (LOGAND SWORDCONTENTS
                                                                                 (\BITMASK (ADD1 SBITOFFSET
                                                                                                 ]
                                                                  0COLOR)
                                                                 (T 1COLOR]
                                                    (\PUTBASEBYTE
                                                     DBASE 1 (LOGOR (LLSH (COND
                                                                          ([ZEROP (COND
                                                                                    ((EQ SBITOFFSET 14)
                                                                          (* the next one is in the next word if the offset is 14)
                                                                                     (fetch (TWOOFFSETBITACCESS
                                                                                                BIT0OFNEXTWORD)
                                                                                        of SBASE))
                                                                                   (T (LOGAND SWORDCONTENTS
                                                                                              (\BITMASK (IPLUS SBITOFFSET
                                                                                                              2]
                                                                           0COLOR)
                                                                          (T 1COLOR))
                                                                      4)
                                                                (LOGAND (\GETBASE DBASE 0)
                                                                        15]
                     (RETURN))
                    (T                                              (* move 4 bits from source to destination.)
                       [\PUTBASE DBASE 0 (\GETBASE MAPBASE (SELECTQ SBITOFFSET
                                                                   (2 (fetch (TWOOFFSETBITACCESS BITS2TO5)
                                                                         of SBASE))
                                                                   (6 (fetch (TWOOFFSETBITACCESS BITS6TO9)
                                                                         of SBASE))
                                                                   (10 (fetch (TWOOFFSETBITACCESS BITS10TO13)
                                                                          of SBASE))
                                                                   (LOGOR (LLSH (fetch (TWOOFFSETBITACCESS
                                                                                           BITS14TO15)
                                                                                   of SBASE)
                                                                                2)
                                                                          (fetch (TWOOFFSETBITACCESS BITS0TO1)
                                                                             of (SETQ SBASE (\ADDBASE SBASE 1]
                       (SETQ DBASE (\ADDBASE DBASE 1))
                       (SETQ WIDTH (IDIFFERENCE WIDTH 4))
                       (COND
                          ((EQ (SETQ SBITOFFSET (IPLUS SBITOFFSET 4))
                               18)                                  (* SBASE has already been incremented as part of fetching the
                           last 4 bits.)
                           (SETQ SBITOFFSET 2)))
                       (GO ONEWRDLP)))
            LP  (COND
                   ((IGREATERP WIDTH (SUB1 BITSPERWORD))            (* move a source word's worth of bits.)
                    (\PUTBASE DBASE 0 (\GETBASE MAPBASE (fetch (TWOOFFSETBITACCESS BITS2TO5) of SBASE)))
                    (\PUTBASE DBASE 1 (\GETBASE MAPBASE (fetch (TWOOFFSETBITACCESS BITS6TO9) of SBASE)))
                    (\PUTBASE DBASE 2 (\GETBASE MAPBASE (fetch (TWOOFFSETBITACCESS BITS10TO13) of SBASE)))
                    [\PUTBASE DBASE 3 (\GETBASE MAPBASE (LOGOR (LLSH (fetch (TWOOFFSETBITACCESS BITS14TO15)
                                                                        of SBASE)
                                                                     2)
                                                              (fetch (TWOOFFSETBITACCESS BITS0TO1)
                                                                 of (SETQ SBASE (\ADDBASE SBASE 1]
                    (SETQ DBASE (\ADDBASE DBASE 4))
                    (SETQ WIDTH (IDIFFERENCE WIDTH BITSPERWORD))
                    (GO LP))
                   (T                                               (* finish off last less than 16 bits.)
                      (GO ONEWRDLP])
         (PROG NIL

           (* moving bits that are aligned with 3 extra bits in the following word of the source.)

           ONEWRDLP
                                                                    (* SBITOFFSET is either 3, 7, 11 or 15)
               (COND
                  ((AND (EQ SBITOFFSET 3)
                        (IGREATERP WIDTH (SUB1 BITSPERWORD)))       (* go to center loop.)
                   (GO LP))
                  ((IGREATERP 4 WIDTH)
                   [PROG (SWORDCONTENTS)
                         (SETQ SWORDCONTENTS (\GETBASE SBASE 0))
                         (SELECTQ WIDTH
                              (0)
                              (1 (PUTBASEBYTE DBASE 0 (LOGOR (LLSH (COND
                                                                   ((ZEROP (LOGAND SWORDCONTENTS (\BITMASK
```

```
                                                                                                    SBITOFFSET))
                                                                                                )
                                                                                             0COLOR)
                                                                                          (T 1COLOR))
                                                                                       4)
                                                                                 (LOGAND (\GETBASEBYTE DBASE 0)
                                                                                         15))))
                                                  (2 [PUTBASEBYTE DBASE 0 (LOGOR (LLSH (COND
                                                                                          ((ZEROP (LOGAND SWORDCONTENTS (\BITMASK
                                                                                                                         SBITOFFSET))
                                                                                                )
                                                                                             0COLOR)
                                                                                          (T 1COLOR))
                                                                                       4)
                                                                                 (COND
                                                                                    ([ZEROP (COND
                                                                                              ((EQ SBITOFFSET 15)
                                                                                               (* the next bit is in the next word if the offset is 15)
                                                                                                (fetch (TWOOFFSETBITACCESS
                                                                                                        BIT0OFNEXTWORD)
                                                                                                   of SBASE))
                                                                                              (T (LOGAND SWORDCONTENTS
                                                                                                         (\BITMASK (IPLUS SBITOFFSET
                                                                                                                         2]
                                                                                             0COLOR)
                                                                                          (T 1COLOR])
                                                  (PROGN                                    (* first two bits are always in this word.)
                                                         [\PUTBASEBYTE DBASE 0
                                                                (LOGOR (LLSH (COND
                                                                                ((ZEROP (LOGAND SWORDCONTENTS (\BITMASK SBITOFFSET)))
                                                                                 0COLOR)
                                                                                (T 1COLOR))
                                                                             4)
                                                                       (COND
                                                                          ([ZEROP (COND
                                                                                    ((EQ SBITOFFSET 15)
                                                                                     (* the next bit is in the next word if the offset is 15)
                                                                                      (fetch (TWOOFFSETBITACCESS BIT0OFNEXTWORD)
                                                                                         of SBASE))
                                                                                    (T (LOGAND SWORDCONTENTS (\BITMASK (IPLUS
                                                                                                                        SBITOFFSET
                                                                                                                           2]
                                                                           0COLOR)
                                                                          (T 1COLOR]
                                                         (\PUTBASEBYTE DBASE 1
                                                                (LOGOR (LLSH (COND
                                                                                ([ZEROP (COND
                                                                                          ((EQ SBITOFFSET 15)
                                                                                           (* the next one is in the next word if the offset is 15)
                                                                                            (fetch (TWOOFFSETBITACCESS BIT1OFNEXTWORD)
                                                                                               of SBASE))
                                                                                          (T (LOGAND SWORDCONTENTS
                                                                                                     (\BITMASK (IPLUS SBITOFFSET 2]
                                                                                 0COLOR)
                                                                                (T 1COLOR))
                                                                             4)
                                                                       (LOGAND (\GETBASE DBASE 0)
                                                                               15]
                                                   (RETURN))
                                                  (T                                        (* move 4 bits from source to destination.)
                                                     [\PUTBASE DBASE 0 (\GETBASE MAPBASE (SELECTQ SBITOFFSET
                                                                                           (3 (fetch (THREEOFFSETBTACCESS BITS3TO6)
                                                                                                 of SBASE))
                                                                                           (7 (fetch (THREEOFFSETBTACCESS BITS7TO10)
                                                                                                 of SBASE))
                                                                                           (11 (fetch (THREEOFFSETBTACCESS BITS11TO14)
                                                                                                  of SBASE))
                                                                                           (LOGOR (LLSH (fetch (ODD2BITNIBBLES BIT15)
                                                                                                           of SBASE)
                                                                                                        3)
                                                                                                  (fetch (THREEOFFSETBTACCESS BITS0TO2)
                                                                                                     of (SETQ SBASE (\ADDBASE SBASE 1]
                                                     (SETQ DBASE (\ADDBASE DBASE 1))
                                                     (SETQ WIDTH (IDIFFERENCE WIDTH 4))
                                                     (COND
                                                        ((EQ (SETQ SBITOFFSET (IPLUS SBITOFFSET 4))
                                                             19)                            (* SBASE has already been incremented as part of fetching the
                                                         last 4 bits.)
                                                         (SETQ SBITOFFSET 3)))
                                                     (GO ONEWRDLP)))
                                LP   (COND
                                        ((IGREATERP WIDTH (SUB1 BITSPERWORD))           (* move a source word's worth of bits.)
                                         (\PUTBASE DBASE 0 (\GETBASE MAPBASE (fetch (THREEOFFSETBTACCESS BITS3TO6) of SBASE)))
                                         (\PUTBASE DBASE 1 (\GETBASE MAPBASE (fetch (THREEOFFSETBTACCESS BITS7TO10) of SBASE)))
                                         (\PUTBASE DBASE 2 (\GETBASE MAPBASE (fetch (THREEOFFSETBTACCESS BITS11TO14) of SBASE)))
                                         [\PUTBASE DBASE 3 (\GETBASE MAPBASE (LOGOR (LLSH (fetch (ODD2BITNIBBLES BIT15) of SBASE)
                                                                                          3)
```

```
                                                        (fetch (THREEOFFSETBTACCESS BITS0TO2)
                                                           of (SETQ SBASE (\ADDBASE SBASE 1]
                (SETQ DBASE (\ADDBASE DBASE 4))
                (SETQ WIDTH (IDIFFERENCE WIDTH BITSPERWORD))
                (GO LP))
               (T                                                (* finish off last less than 16 bits.)
                  (GO ONEWRDLP])
```

(\8BITLINEBLT

```
[LAMBDA (SBASE SBITOFFSET DBASE DBITOFFSET WIDTH MAPBASE 0COLOR 1COLOR)
                                                                 (* edited%: "16-SEP-82 19:36")

    (* moves one line of a black and white bitmap into a color bitmap using a mapping table.)

  [COND
     ((EQ 1 DBITOFFSET)                                          (* move the first bit specially to get to word boundary in
                                                                 destination.)
      (\PUTBASEBYTE DBASE 1 (COND
                            ((ZEROP (LOGAND (\GETBASE SBASE 0)
                                            (\BITMASK SBITOFFSET)))
                             0COLOR)
                            (T 1COLOR)))
      [COND
         ((EQ (SETQ SBITOFFSET (ADD1 SBITOFFSET))
              BITSPERWORD)                                       (* SBITOFFSET flowed onto next word.)
          (SETQ SBITOFFSET 0)
          (SETQ SBASE (\ADDBASE SBASE 1]
      (SETQ DBITOFFSET 0)
      (SETQ DBASE (\ADDBASE DBASE 1))
      (SETQ WIDTH (SUB1 WIDTH]
  (COND
     ((ZEROP (MOD SBITOFFSET 2))                                 (* case of moving even aligned bits.)
      (PROG NIL
        LP  [COND
               ((AND (IGREATERP WIDTH (SUB1 BITSPERWORD))
                     (EQ SBITOFFSET 0))                          (* move a source word's worth of bits.)
                (\PUTBASE DBASE 0 (\GETBASE MAPBASE (fetch (2BITNIBBLES EN1) of SBASE)))
                (\PUTBASE DBASE 1 (\GETBASE MAPBASE (fetch (2BITNIBBLES EN2) of SBASE)))
                (\PUTBASE DBASE 2 (\GETBASE MAPBASE (fetch (2BITNIBBLES EN3) of SBASE)))
                (\PUTBASE DBASE 3 (\GETBASE MAPBASE (fetch (2BITNIBBLES EN4) of SBASE)))
                (\PUTBASE DBASE 4 (\GETBASE MAPBASE (fetch (2BITNIBBLES EN5) of SBASE)))
                (\PUTBASE DBASE 5 (\GETBASE MAPBASE (fetch (2BITNIBBLES EN6) of SBASE)))
                (\PUTBASE DBASE 6 (\GETBASE MAPBASE (fetch (2BITNIBBLES EN7) of SBASE)))
                (\PUTBASE DBASE 7 (\GETBASE MAPBASE (fetch (2BITNIBBLES EN8) of SBASE)))
                (SETQ DBASE (\ADDBASE DBASE 8))
                (SETQ SBASE (\ADDBASE SBASE 1))
                (SETQ WIDTH (IDIFFERENCE WIDTH BITSPERWORD)))
               ((EQ WIDTH 0)
                (RETURN))
               ((EQ WIDTH 1)                                     (* move last bit specially)
                (\PUTBASEBYTE DBASE 0 (COND
                                      ((ZEROP (LOGAND (\GETBASE SBASE 0)
                                                      (\BITMASK SBITOFFSET)))
                                       0COLOR)
                                      (T 1COLOR)))
                (RETURN))
               (T                                                (* move the rest of the first word or last word two at a time.)
                  (\PUTBASEBYTE DBASE 0 (COND
                                        ((ZEROP (LOGAND (\GETBASE SBASE 0)
                                                        (\BITMASK SBITOFFSET)))
                                         0COLOR)
                                        (T 1COLOR)))
                  (\PUTBASEBYTE DBASE 1 (COND
                                        ([ZEROP (LOGAND (\GETBASE SBASE 0)
                                                        (\BITMASK (ADD1 SBITOFFSET]
                                         0COLOR)
                                        (T 1COLOR)))
                  (SETQ DBASE (\ADDBASE DBASE 1))
                  (SETQ WIDTH (IDIFFERENCE WIDTH 2))
                  (COND
                     ((EQ SBITOFFSET 14)
                      (SETQ SBASE (\ADDBASE SBASE 1))
                      (SETQ SBITOFFSET 0))
                     (T (SETQ SBITOFFSET (IPLUS SBITOFFSET 2]
            (GO LP)))
     (T                                                          (* moving odd aligned bits.)
        (PROG NIL
          LP  [COND
                 ((AND (IGREATERP WIDTH (SUB1 BITSPERWORD))
                       (EQ SBITOFFSET 1))

    (* move a source word's worth of bits. move the 1th thru 15th bits in the first word plus the 0th bit in the next word.)

                (\PUTBASE DBASE 0 (\GETBASE MAPBASE (fetch (ODD2BITNIBBLES ODD2BIT1) of SBASE)))
                (\PUTBASE DBASE 1 (\GETBASE MAPBASE (fetch (ODD2BITNIBBLES ODD2BIT2) of SBASE)))
                (\PUTBASE DBASE 2 (\GETBASE MAPBASE (fetch (ODD2BITNIBBLES ODD2BIT3) of SBASE)))
```

```
                    (\PUTBASE DBASE 3 (\GETBASE MAPBASE (fetch (ODD2BITNIBBLES ODD2BIT4) of SBASE)))
                    (\PUTBASE DBASE 4 (\GETBASE MAPBASE (fetch (ODD2BITNIBBLES ODD2BIT5) of SBASE)))
                    (\PUTBASE DBASE 5 (\GETBASE MAPBASE (fetch (ODD2BITNIBBLES ODD2BIT6) of SBASE)))
                    (\PUTBASE DBASE 6 (\GETBASE MAPBASE (fetch (ODD2BITNIBBLES ODD2BIT7) of SBASE)))
                    (\PUTBASEBYTE DBASE 14 (COND
                                              ((ZEROP (fetch (ODD2BITNIBBLES BIT15) of SBASE))
                                               0COLOR)
                                              (T 1COLOR)))
                    (\PUTBASEBYTE DBASE 15 (COND
                                              ([ZEROP (fetch (ODD2BITNIBBLES BIT0) of (SETQ SBASE
                                                                                        (\ADDBASE SBASE 1]
                                               0COLOR)
                                              (T 1COLOR)))
                    (SETQ DBASE (\ADDBASE DBASE 8))
                    (SETQ WIDTH (IDIFFERENCE WIDTH BITSPERWORD)))
                   ((EQ WIDTH 0)
                    (RETURN))
                   ((EQ WIDTH 1)                                    (* move last bit specially)
                    (\PUTBASEBYTE DBASE 0 (COND
                                             ((ZEROP (LOGAND (\GETBASE SBASE 0)
                                                             (\BITMASK SBITOFFSET)))
                                              0COLOR)
                                             (T 1COLOR)))
                    (RETURN))
                   ((EQ SBITOFFSET 15)                              (* case of moving one bit from each of two words in the slow
                                                                    case.)
                    (\PUTBASEBYTE DBASE 0 (COND
                                             ((ZEROP (fetch (ODD2BITNIBBLES BIT15) of SBASE))
                                              0COLOR)
                                             (T 1COLOR)))
                    (\PUTBASEBYTE DBASE (SETQ SBITOFFSET 1)
                           (COND
                              ([ZEROP (fetch (ODD2BITNIBBLES BIT0) of (SETQ SBASE (\ADDBASE SBASE 1]
                               0COLOR)
                              (T 1COLOR)))
                    (SETQ WIDTH (IDIFFERENCE WIDTH 2))
                    (SETQ DBASE (\ADDBASE DBASE 1)))
                   (T
```

(* move the rest of the first word or the rest of last word two at a time.)

```
                     (\PUTBASEBYTE DBASE 0 (COND
                                              ((ZEROP (LOGAND (\GETBASE SBASE 0)
                                                              (\BITMASK SBITOFFSET)))
                                               0COLOR)
                                              (T 1COLOR)))
                     (\PUTBASEBYTE DBASE 1 (COND
                                              ([ZEROP (LOGAND (\GETBASE SBASE 0)
                                                              (\BITMASK (ADD1 SBITOFFSET]
                                               0COLOR)
                                              (T 1COLOR)))
                     (SETQ SBITOFFSET (IPLUS SBITOFFSET 2))
                     (SETQ WIDTH (IDIFFERENCE WIDTH 2))
                     (SETQ DBASE (\ADDBASE DBASE 1]
                   (GO LP])
```

(\24BITLINEBLT

```
  [LAMBDA (SBASE SLEFT DBASE DLEFT WIDTH 0COLOR 1COLOR)             (* kbr%: "15-Feb-86 10:56")
```

(* moves one line of a black and white bitmap into a color bitmap using a mapping table.)

```
    (PROG NIL
          (for SX from SLEFT to (IPLUS SLEFT WIDTH -1) as DX from DLEFT
             do (\PUTBASE24 DBASE DX (COND
                                        ([ZEROP (LOGAND (\GETBASE SBASE (FOLDLO SX BITSPERWORD))
                                                        (\BITMASK (LOGAND SX 15]
                                         0COLOR)
                                        (T 1COLOR])
```

(\GETBASE24

```
  [LAMBDA (X D)                                                     (* kbr%: "13-Feb-86 21:07")
                                                                    (* Get Dth 24bit pixel from packed X.
                                                                    *)
    (PROG (DWORD ANSWER)
          (SETQ DWORD (FOLDLO (ITIMES 24 D)
                              BITSPERWORD))
          [SETQ ANSWER (SELECTQ (LOGAND D 1)
                               (0                                   (* Get nibbles 1 0 of DWORD and nibble 1 of following word.
                                                                    *)
                                  (LOGOR (LLSH (\GETBASE X DWORD)
                                               8)
                                         (LRSH (\GETBASE X (ADD1 DWORD))
                                               8)))
                               (PROGN                               (* Get nibble 0 of DWORD and nibbles 1 0 of following word.
                                                                    *)
```

```
                                    (LOGOR (LLSH (LOGAND (\GETBASE X DWORD)
                                                         (MASK.1'S 0 8))
                                                 16)
                                           (\GETBASE X (ADD1 DWORD]
          (RETURN ANSWER])
```

(\PUTBASE24

```
  [LAMBDA (X D V)                                          (* kbr%: "13-Feb-86 21:19")
                                                           (* Set Dth 24bit pixel from packed X.
                                                           *)
    (PROG (DWORD)
          (SETQ DWORD (FOLDLO (ITIMES 24 D)
                              BITSPERWORD))
          (SELECTQ (LOGAND D 1)
               (0                                          (* Replace nibbles 1 0 of DWORD and nibble 1 of following
                                                           word. *)
                  (\PUTBASE X DWORD (LRSH V 8))
                  [\PUTBASE X (ADD1 DWORD)
                         (LOGOR (LLSH (LOGAND V (MASK.1'S 0 8))
                                      8)
                                (LOGAND (\GETBASE X DWORD)
                                        (MASK.1'S 0 8])
               (PROGN                                      (* Replace nibble 0 of DWORD and nibbles 1 0 of following
                                                           word. *)
                      (\PUTBASE X DWORD (LOGOR (LOGAND (\GETBASE X DWORD)
                                                       (LLSH (MASK.1'S 0 8)
                                                             8))
                                               (LRSH V 16)))
                      (\PUTBASE X (ADD1 DWORD)
                             (LOGAND V (MASK.1'S 0 16])
```

(COLORTEXTUREFROMCOLOR#

```
  [LAMBDA (COLOR# BITSPERPIXEL)                            (* kbr%: "27-Feb-86 16:48")
```

(* returns a TEXTURE that is COLOR# tessellated in a pattern to put down BITSPERPIXEL per pixel color)

```
    (PROG (TEXTURE)
          (COND
             ((type? BITMAP COLOR#)                        (* already is a texture.)
              (RETURN COLOR#)))
          (SETQ COLOR# (COLORNUMBERP COLOR# BITSPERPIXEL))
          (SETQ TEXTURE (SELECTQ BITSPERPIXEL
                               (4 (PROG (TEXTUREBITMAP BITPATTERN)
                                        (SETQ TEXTUREBITMAP (BITMAPCREATE 4 4 4))
                                        (SETQ BITPATTERN (LOGOR (LLSH COLOR# 12)
                                                                (LLSH COLOR# 8)
                                                                (LLSH COLOR# 4)
                                                                COLOR#))
                                        (for I from 0 to 3 do (\BITMAPWORD TEXTUREBITMAP I BITPATTERN))
                                        (RETURN TEXTUREBITMAP)))
                               (8 (PROG (TEXTUREBITMAP BITPATTERN)
                                        (SETQ TEXTUREBITMAP (BITMAPCREATE 2 4 8))
                                        (SETQ BITPATTERN (LOGOR (LLSH COLOR# 8)
                                                                COLOR#))
                                        (for I from 0 to 3 do (\BITMAPWORD TEXTUREBITMAP I BITPATTERN))
                                        (RETURN TEXTUREBITMAP)))
                               (24                         (* This isn't right, but at least it won't break you.
                                                           *)
                                   (PROG (TEXTUREBITMAP BITMAPBASE)
                                         (SETQ TEXTUREBITMAP (BITMAPCREATE 2 4 24))
                                         (SETQ BITMAPBASE (fetch (BITMAP BITMAPBASE) of TEXTUREBITMAP))
                                         (for I from 0 to 7 do (\PUTBASE24 BITMAPBASE I COLOR#))
                                         (RETURN TEXTUREBITMAP)))
                               (ERROR "Only 4, 8 and 24 bits per pixel implemented.")))
          (RETURN TEXTURE])
```

(\BITMAPWORD

```
  [LAMBDA (BM WORDN NEWBITS)                               (* edited%: " 8-SEP-82 10:54")
                                                           (* puts a words worth of bits into the WORDNth word of a
                                                           bitmap.)
    (\PUTBASE (\ADDBASE (fetch (BITMAP BITMAPBASE) of BM)
                        WORDN)
              0
              (LOGAND NEWBITS WORDMASK])

)

(DEFINEQ
```

(COLORIZEBITMAP

```
  [LAMBDA (BITMAP 0COLOR 1COLOR BITSPERPIXEL)              (* kbr%: "15-Feb-86 10:13")
```

(* creates a copy of BITMAP that is in color form allowing BITSPERPIXEL per pixel.
0COLOR and 1COLOR are the color numbers that get translated from 0 and 1 respectively.)

```
    (PROG (COLORBITMAP)
          (SETQ COLORBITMAP (BITMAPCREATE (fetch (BITMAP BITMAPWIDTH) of BITMAP)
                                          (fetch (BITMAP BITMAPHEIGHT) of BITMAP)
                                          BITSPERPIXEL))
          (\BWTOCOLORBLT BITMAP 0 0 COLORBITMAP 0 0 (fetch (BITMAP BITMAPWIDTH) of BITMAP)
                 (fetch (BITMAP BITMAPHEIGHT) of BITMAP)
                 (COLORNUMBERP 0COLOR BITSPERPIXEL)
                 (COLORNUMBERP 1COLOR BITSPERPIXEL)
                 BITSPERPIXEL)
          (RETURN COLORBITMAP])
```

(UNCOLORIZEBITMAP

```
  [LAMBDA (BITMAP COLORMAP)                                    (* kbr%: " 2-Sep-85 19:21")
    (PROG (BITSPERPIXEL MAXCOLOR MAXX MAXY BWBITMAP TABLE RGB R G B BIT BASE BWBASE RASTERWIDTH BWRASTERWIDTH
          WORD)
          (SETQ MAXX (SUB1 (fetch (BITMAP BITMAPWIDTH) of BITMAP)))
          (SETQ MAXY (SUB1 (fetch (BITMAP BITMAPHEIGHT) of BITMAP)))
          (SETQ BITSPERPIXEL (fetch (BITMAP BITMAPBITSPERPIXEL) of BITMAP))
          (SETQ COLORMAP (OR COLORMAP (COLORMAP BITSPERPIXEL)))
          (SETQ MAXCOLOR (MAXIMUMCOLOR BITSPERPIXEL))
          (SETQ BWBITMAP (BITMAPCREATE (ADD1 MAXX)
                                       (ADD1 MAXY)
                                       1))
          (SETQ TABLE (\ALLOCBLOCK (FOLDHI (ADD1 MAXCOLOR)
                                           2)))
          (for I from 0 to MAXCOLOR do (SETQ RGB (ELT COLORMAP I))
                                       (SETQ R (fetch (RGB RED) of RGB))
                                       (SETQ G (fetch (RGB GREEN) of RGB))
                                       (SETQ B (fetch (RGB BLUE) of RGB))
                                       (SETQ BIT (IDIFFERENCE 1 (IQUOTIENT (IPLUS R G B)
                                                                           384)))
                                       (\PUTBASE TABLE I BIT))
          (SETQ BASE (fetch (BITMAP BITMAPBASE) of BITMAP))
          (SETQ BWBASE (fetch (BITMAP BITMAPBASE) of BWBITMAP))
          (SETQ RASTERWIDTH (fetch (BITMAP BITMAPRASTERWIDTH) of BITMAP))
          (SETQ BWRASTERWIDTH (fetch (BITMAP BITMAPRASTERWIDTH) of BWBITMAP))
          (SELECTQ BITSPERPIXEL
              (4 [for Y from 0 to MAXY do (SETQ WORD 0)
                                          [for X from 0 to MAXX do [SETQ WORD (LOGOR (LLSH WORD 1)
                                                                                     (\GETBASE TABLE
                                                                                            (\GETBASENYBBLE BASE X
                                                                                                   ]
                                                                                 (COND
                                                                                    ((EQ (LOGAND X 15)
                                                                                         15)
                                                                                     (\PUTBASE BWBASE (FOLDLO X 16)
                                                                                            WORD)
                                                                                     (SETQ WORD 0]
                                          (COND
                                             ((NOT (EQ (LOGAND MAXX 15)
                                                       15))
                                              [SETQ WORD (LLSH WORD (IDIFFERENCE 15 (LOGAND MAXX 15]
                                              (\PUTBASE BWBASE (FOLDLO MAXX 16)
                                                     WORD)))
                                          (COND
                                             ((NOT (EQ Y MAXY))
                                              (SETQ BASE (\ADDBASE BASE RASTERWIDTH))
                                              (SETQ BWBASE (\ADDBASE BWBASE BWRASTERWIDTH])
              (8 [for Y from 0 to MAXY do (SETQ WORD 0)
                                          [for X from 0 to MAXX do [SETQ WORD (LOGOR (LLSH WORD 1)
                                                                                     (\GETBASE TABLE
                                                                                            (\GETBASEBYTE BASE X]
                                                                                 (COND
                                                                                    ((EQ (LOGAND X 15)
                                                                                         15)
                                                                                     (\PUTBASE BWBASE (FOLDLO X 16)
                                                                                            WORD)
                                                                                     (SETQ WORD 0]
                                          (COND
                                             ((NOT (EQ (LOGAND MAXX 15)
                                                       15))
                                              [SETQ WORD (LLSH WORD (IDIFFERENCE 15 (LOGAND MAXX 15]
                                              (\PUTBASE BWBASE (FOLDLO MAXX 16)
                                                     WORD)))
                                          (COND
                                             ((NOT (EQ Y MAXY))
                                              (SETQ BASE (\ADDBASE BASE RASTERWIDTH))
                                              (SETQ BWBASE (\ADDBASE BWBASE BWRASTERWIDTH])
              NIL)
          (RETURN BWBITMAP])

)

(RPAQ? \1COLORMENU NIL)
```

```
(RPAQ? \4COLORMENU NIL)

(RPAQ? \8COLORMENU NIL)

(DEFINEQ

(COLORMENU
  [LAMBDA (BITSPERPIXEL)                                    (* kbr%: " 5-Jun-85 18:24")
                                                            (* Make a BITSPERPIXEL color menu.
                                                            *)
    (PROG (MENU ITEMS MENUCOLUMNS MENUROWS BITMAP)          (* Try to find old menu. *)
          (SETQ MENU (SELECTQ BITSPERPIXEL
                         (1 \1COLORMENU)
                         (4 \4COLORMENU)
                         (8 \8COLORMENU)
                         (\ILLEGAL.ARG BITSPERPIXEL)))
          (COND
             (MENU (RETURN MENU)))                          (* Calculate menu items. *)
          (SETQ ITEMS (SELECTQ BITSPERPIXEL
                           (1 (for COLOR from 0 to 1 as SHADE in (LIST WHITESHADE BLACKSHADE)
                                 collect (LIST (PROGN (SETQ BITMAP (BITMAPCREATE 32 32))
                                                      (BLTSHADE SHADE BITMAP)
                                                      BITMAP)
                                               COLOR)))
                           (4 (for COLOR from 0 to 15 collect (LIST (PROGN (SETQ BITMAP (BITMAPCREATE 16 16 4))
                                                                           (BLTSHADE COLOR BITMAP)
                                                                           BITMAP)
                                                                    COLOR)))
                           (8 (for COLOR from 0 to 255 collect (LIST (PROGN (SETQ BITMAP (BITMAPCREATE 8 8 8))
                                                                            (BLTSHADE COLOR BITMAP)
                                                                            BITMAP)
                                                                     COLOR)))
                           (SHOULDNT)))
          (SETQ MENUROWS (SELECTQ BITSPERPIXEL
                             (1 1)
                             (4 4)
                             (8 16)
                             (SHOULDNT)))
          (SETQ MENUCOLUMNS (SELECTQ BITSPERPIXEL
                               (1 2)
                               (4 4)
                               (8 16)
                               (SHOULDNT)))
          (SETQ MENU (create MENU
                            ITEMS _ ITEMS
                            MENUROWS _ MENUROWS
                            MENUCOLUMNS _ MENUCOLUMNS
                            MENUBORDERSIZE _ 1))
          (SELECTQ BITSPERPIXEL
              (1 (SETQ \1COLORMENU MENU))
              (4 (SETQ \4COLORMENU MENU))
              (8 (SETQ \8COLORMENU MENU))
              (SHOULDNT))
          (RETURN MENU])

(CURSORCOLOR
  [LAMBDA (COLOR)                                           (* edited%: " 4-Jun-85 15:56")
    (PROG (IMAGE MASK)
          (SETQ IMAGE (fetch (CURSOR CUIMAGE) of \CURRENTCURSOR))
          (SETQ MASK (fetch (CURSOR CUMASK) of \CURRENTCURSOR))
          (BLTSHADE COLOR IMAGE)
          (BITBLT MASK NIL NIL IMAGE NIL NIL NIL NIL 'INVERT 'ERASE])

)

(DECLARE%: EVAL@COMPILE

(RECORD RGB (RED GREEN BLUE))

(RECORD HLS (HUE LIGHTNESS SATURATION))
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(BLOCKRECORD NIBBLES ((N1 BITS 4)
                      (N2 BITS 4)
                      (N3 BITS 4)
                      (N4 BITS 4)))

(BLOCKRECORD ONEOFFSETBITACCESS ((BIT0 BITS 1)
                                 (BITS1TO4 BITS 4)
                                 (BITS5TO8 BITS 4)
                                 (BITS9TO12 BITS 4)
                                 (BITS13TO15 BITS 3)))
```

```
(BLOCKRECORD TWOOFFSETBITACCESS ((BITS0TO1 BITS 2)
                                 (BITS2TO5 BITS 4)
                                 (BITS6TO9 BITS 4)
                                 (BITS10TO13 BITS 4)
                                 (BITS14TO15 BITS 2)
                                 (BIT0OFNEXTWORD BITS 1)
                                 (BIT1OFNEXTWORD BITS 1)
                                 (BITS2TO15OFNEXTWORD BITS 14)))

(BLOCKRECORD THREEOFFSETBTACCESS ((BITS0TO2 BITS 3)
                                  (BITS3TO6 BITS 4)
                                  (BITS7TO10 BITS 4)
                                  (BITS11TO14 BITS 4)
                                  (BIT15 BITS 1)))

(BLOCKRECORD 2BITNIBBLES ((EN1 BITS 2)
                          (EN2 BITS 2)
                          (EN3 BITS 2)
                          (EN4 BITS 2)
                          (EN5 BITS 2)
                          (EN6 BITS 2)
                          (EN7 BITS 2)
                          (EN8 BITS 2)))

(BLOCKRECORD ODD2BITNIBBLES ((BIT0 BITS 1)
                             (ODD2BIT1 BITS 2)
                             (ODD2BIT2 BITS 2)
                             (ODD2BIT3 BITS 2)
                             (ODD2BIT4 BITS 2)
                             (ODD2BIT5 BITS 2)
                             (ODD2BIT6 BITS 2)
                             (ODD2BIT7 BITS 2)
                             (BIT15 BITS 1)))
)
)

(DECLARE%: EVAL@COMPILE DONTCOPY

(FILESLOAD (LOADCOMP)
       MAIKOCOLOR)
)

(DECLARE%: EVAL@COMPILE

(RPAQQ BITSPERWORD 16)

(CONSTANTS (BITSPERWORD 16))
)

(RPAQ? \COLORDISPLAYFDEV )

(RPAQ? \4COLORMAP (CMYCOLORMAP 2 1 1 4))

(RPAQ? \8COLORMAP (CMYCOLORMAP 3 3 2 8))

(RPAQ? \COLORDISPLAYBITS )

(RPAQ? ColorScreenBitMap )

(RPAQ? \COLORSCREEN )

(DEFINEQ

(PSEUDOCOLOR
  [LAMBDA (TABLE DESTINATION LEFT BOTTOM WIDTH HEIGHT)          (* kbr%: " 2-Sep-85 19:08")
    (DECLARE (LOCALVARS . T))
    (PROG (left top bottom right width height DESTDD DESTSTRM)
          (COND
             ((NULL LEFT)
              (SETQ LEFT 0)))
          (COND
             ((NULL BOTTOM)
              (SETQ BOTTOM 0)))

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

          [COND
             [(type? BITMAP DESTINATION)
              (SETQ left 0)
              (SETQ bottom 0)
              (SETQ right (SUB1 (fetch (BITMAP BITMAPWIDTH) of DESTINATION)))
              (SETQ top (SUB1 (fetch (BITMAP BITMAPHEIGHT) of DESTINATION]
             ((SETQ DESTDD (\GETDISPLAYDATA DESTINATION))
              (SETQ DESTSTRM DESTINATION)
              (SETQ DESTINATION (fetch (\DISPLAYDATA DDDestination) of DESTDD))
```

```
            (SETQ LEFT (\DSPTRANSFORMX LEFT DESTDD))
            (SETQ BOTTOM (\DSPTRANSFORMY BOTTOM DESTDD))
            (PROGN                                              (* compute limits based on clipping regions.)
                   (SETQ left (fetch (\DISPLAYDATA DDClippingLeft) of DESTDD))
                   (SETQ bottom (fetch (\DISPLAYDATA DDClippingBottom) of DESTDD))
                   (SETQ right (fetch (\DISPLAYDATA DDClippingRight) of DESTDD))
                   (SETQ top (fetch (\DISPLAYDATA DDClippingTop) of DESTDD]
         (COND
            ((NOT (EQ (fetch (BITMAP BITMAPBITSPERPIXEL) of DESTINATION)
                      8))
             (ERROR "Pseudocolor only implemented for 8 bitsperpixel bitmaps" DESTINATION)))
         [PROGN (SETQ left (IMAX LEFT left))
                (SETQ bottom (IMAX BOTTOM bottom))
                [COND
                   (WIDTH                                       (* WIDTH is optional)
                          (SETQ right (IMIN (IPLUS LEFT WIDTH)
                                            right]
                (COND
                   (HEIGHT                                      (* HEIGHT is optional)
                          (SETQ top (IMIN (IPLUS BOTTOM HEIGHT)
                                          top]                  (* Clip and translate coordinates.)
         (SETQ width (IPLUS right (IMINUS left)
                            1))
         (SETQ height (IPLUS top (IMINUS bottom)
                             1))
         (COND
            (DESTSTRM (.WHILE.TOP.DS. DESTSTRM (\PSEUDOCOLOR.BITMAP TABLE DESTINATION left bottom width height
                                                                  )))
            (T (\PSEUDOCOLOR.BITMAP TABLE DESTINATION left bottom width height])
```

(\PSEUDOCOLOR.BITMAP

```
  [LAMBDA (TABLE BITMAP LEFT BOTTOM WIDTH HEIGHT)               (* kbr%: "10-Jul-85 22:33")
    (\PSEUDOCOLOR.UFN (fetch (ARRAYP BASE) of TABLE)
           BITMAP LEFT BOTTOM WIDTH 0 HEIGHT])
```

(\PSEUDOCOLOR.UFN

```
  [LAMBDA (TABLEBASE BITMAP LEFT BOTTOM WIDTH ZERO HEIGHT)      (* kbr%: "10-Jul-85 22:37")

          (* * Substitutes colors according to TABLEBASE within region of 8 bitsperpixel BITMAP.
          *)

    (PROG (BASE RASTERWIDTH BMHEIGHT TOP RIGHT ROWBASE)
          (SETQ BASE (fetch (BITMAP BITMAPBASE) of BITMAP))
          (SETQ RASTERWIDTH (fetch (BITMAP BITMAPRASTERWIDTH) of BITMAP))
          (SETQ BMHEIGHT (fetch (BITMAP BITMAPHEIGHT) of BITMAP))
          (SETQ RIGHT (IPLUS LEFT WIDTH -1))
          (SETQ BOTTOM (ITIMES RASTERWIDTH (IDIFFERENCE (SUB1 BMHEIGHT)
                                                        BOTTOM)))
          [SETQ TOP (IDIFFERENCE BOTTOM (ITIMES RASTERWIDTH (SUB1 HEIGHT]
          (for Y from TOP to BOTTOM by RASTERWIDTH do (SETQ ROWBASE (\ADDBASE BASE Y))
                                                      (for X from LEFT to RIGHT
                                                         do (\PUTBASEBYTE ROWBASE X (\GETBASE TABLEBASE
                                                                                              (\GETBASEBYTE ROWBASE
                                                                                                     X])

)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \COLORDISPLAYFDEV \COLORDISPLAYBITS ColorScreenBitMap \4COLORMAP \8COLORMAP)
)

;; NOTE: This is very bad.  I shouldn't have to and don't really want to do the following, but since about March 86, someone did something really
;; nonstandard wrt Helvetica fonts so that the in core versions are not equal to what is stored on file.  The SETFONTDESCRIPTOR and friends undoes
;; this kludge which has never been explained to LISPCORE^ by the person who brain damaged Helvetica this way.  If I don't undo this kludge by
;; someone else, then color menus come out wrong.  *

(SETFONTDESCRIPTOR 'HELVETICA 10 'MRR 0 'DISPLAY NIL)

(SETQ MENUFONT (FONTCREATE 'HELVETICA 10))

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA )
)

(PUTPROPS LLCOLOR COPYRIGHT ("Xerox Corporation" 1982 1983 1984 1985 1986 1987 1988 1989 1990 1991 1992))
```

FUNCTION INDEX

CMYCOLORMAP 3
COLORDISPLAY 1
COLORFROMRGB 6
COLORIZEBITMAP 20
COLORLEVEL 5
COLORMAP 2
COLORMAPBITS 2
COLORMAPCOPY 3
COLORMAPCREATE 5
COLORMENU 22
COLORNUMBERP 5
COLORSCREEN 4
COLORSCREENBITMAP 4
COLORTEXTUREFROMCOLOR# 20
CURSORCOLOR 22
GRAYCOLORMAP 4
INTENSITIESFROMCOLORMAP 6
PSEUDOCOLOR 23
RGBCOLORMAP 3
ROTATECOLORMAP 3
SCREENCOLORMAP 3
SCREENCOLORMAPENTRY 3
SETCOLORINTENSITY 6
SHOWCOLORTESTPATTERN 4
UNCOLORIZEBITMAP 21
\24BITLINEBLT 19
\4BITLINEBLT 13
\8BITLINEBLT 18
\BWTOCOLORBLT 12
\COLORDISPLAYBITS 4
\CREATECOLORDISPLAYFDEV 2
\CreateColorScreenBitMap 2
\DRAW24BPPCOLORLINE 8
\DRAW4BPPCOLORLINE 8
\DRAW8BPPCOLORLINE 8
\DRAWCOLORLINE1 8
\FAST8BIT 6
\GETBASE24 19
\GETCOLORBRUSH 7
\MAP4 7
\MAP8 7
\PSEUDOCOLOR.BITMAP 24
\PSEUDOCOLOR.UFN 24
\PUTBASE24 20
\SENDCOLORMAPENTRY 5
\STARTCOLOR 5
\STOPCOLOR 5

VARIABLE INDEX

COLORMONITORTYPE ..5 \1COLORMENU .....21 \4COLORMENU .....22 \8COLORMENU .....22 \COLORDISPLAYFDEV 23
ColorScreenBitMap 23 \4COLORMAP .....23 \8COLORMAP .....23 \COLORDISPLAYBITS 23 \COLORSCREEN 23

RECORD INDEX

2BITNIBBLES .....23 NIBBLES .....22 ONEOFFSETBITACCESS .....22 THREEOFFSETBTACCESS 23
HLS .....22 ODD2BITNIBBLES .....23 RGB .....22 TWOOFFSETBITACCESS 23

MACRO INDEX

.DRAW24BPPLINEX .....10 .DRAW4BPPLINEX. .....8 .DRAW8BPPLINEX 9
.DRAW24BPPLINEY .....11 .DRAW4BPPLINEY. .....10 .DRAW8BPPLINEY 10

CONSTANT INDEX

BITSPERWORD 23