*File created:* 26-Oct-2021 10:53:57 **{DSK}<home>larry>medley>library>MAIKOCOLOR.;2**

*changes to:* (VARS MAIKOCOLORCOMS)
(MACROS \MAIKO.CGTHREEP \MAIKO.CGFOURP \MAIKO.CGSIXP \MAIKO.CGTWOP)
(FNS \MAIKO.COLORINIT \MAIKO.STARTCOLOR \MAIKO.STOPCOLOR \MAIKOCOLOR.EVENTFN
\MAIKO.SENDCOLORMAPENTRY \MAIKO.CHANGESCREEN CURSOREXIT CURSORSCREEN WARPCURSOR \SLOWBLTCHAR
\SOFTCURSORUP \BITBLT.DISPLAY \PUNT.SLOWBLTCHAR \PUNT.BLTSHADE.BITMAP \PUNT.BITBLT.BITMAP
BITMAPOBJ.SNAPW \MAIKO.PUNTBLTCHAR \MAIKO.BLTCHAR)

*previous date:* 23-Oct-91 14:43:35 {DSK}<home>larry>medley>library>MAIKOCOLOR.;1

*Read Table:* INTERLISP

*Package:* INTERLISP

*Format:* XCCS

```
;;
;; Copyright (c) 1988-1991 by Fuji Xerox Co., Ltd..

(RPAQQ MAIKOCOLORCOMS
       [(P (MOVD? 'BITBLT 'ORG.BITBLT)
           (MOVD? 'BLTSHADE 'ORG.BLTSHADE)
           (MOVD? '\SLOWBLTCHAR '\OLD.SLOWBLTCHAR)
           (MOVD? 'CURSOREXIT 'OLD.CURSOREXIT)
           (MOVD? '\SOFTCURSORUP '\OLD.SOFTCURSORUP))
        (FNS \MAIKO.COLORINIT \MAIKO.STARTCOLOR \MAIKO.STOPCOLOR \MAIKOCOLOR.EVENTFN \MAIKO.SENDCOLORMAPENTRY
             \MAIKO.CHANGESCREEN)
        (FNS CURSOREXIT CURSORSCREEN WARPCURSOR \SLOWBLTCHAR \SOFTCURSORUP \BITBLT.DISPLAY)
                                                                    ; these FNS defs. will be moved to original files,later
        (FNS \PUNT.SLOWBLTCHAR \MAIKO.PUNTBLTCHAR \MAIKO.BLTCHAR)
        (FNS \PUNT.BLTSHADE.BITMAP \PUNT.BITBLT.BITMAP)
        (FNS BITMAPOBJ.SNAPW)
        (DECLARE%: EVAL@COMPILE DONTCOPY (MACROS \MAIKO.CGTHREEP \MAIKO.CGFOURP \MAIKO.CGSIXP \MAIKO.CGTWOP)
               (CONSTANTS (\TO.MAIKO.MONOSCREEN 0)
                      (\TO.MAIKO.COLORSCREEN 1)
                      (\MAIKO.COLORSCREENWIDTH 1152)
                      (\MAIKO.COLORSCREENHEIGHT 900)
                      (\MAIKO.COLORPAGES 2048)
                      (\MAIKO.COLORBUF.ALIGN 4095))
               (FILES (LOADCOMP)
                      LLDISPLAY BIGBITMAPS))
        (INITVARS \MONO.PROMPTWINDOW \COLOR.PROMPTWINDOW)
        (GLOBALVARS MAIKOCOLOR.BITSPERPIXEL)
        (FILES COLOR BIGBITMAPS)
        (DECLARE%: DONTEVAL@LOAD DOCOPY (P (MOVD 'CURSOREXIT 'SAVE.CURSOREXIT)
                                           (MOVD '\MAIKO.BLTCHAR '\BILTCHAR)
                                           (\MAIKO.COLORINIT)
                                           (COLORDISPLAY 'ON 'MAIKOCOLOR)
                                           (CURSORSCREEN (COLORSCREEN)
                                                  100 100)
                                           (CHANGEBACKGROUND 36)
                                           (ADD-EXEC :TTY T :REGION '(0 650 370 150))
                                           (LOGOW])

(MOVD? 'BITBLT 'ORG.BITBLT)

(MOVD? 'BLTSHADE 'ORG.BLTSHADE)

(MOVD? '\SLOWBLTCHAR '\OLD.SLOWBLTCHAR)

(MOVD? 'CURSOREXIT 'OLD.CURSOREXIT)

(MOVD? '\SOFTCURSORUP '\OLD.SOFTCURSORUP)

(DEFINEQ

(\MAIKO.COLORINIT
  [LAMBDA NIL
    (DECLARE (GLOBALVARS \MAIKOCOLORWSOPS \MAIKOCOLORINFO))     ; Edited 28-Apr-89 16:51 by tshimizu.fx
    (SETQ \MAIKOCOLORWSOPS (create WSOPS
                                   STARTBOARD _ (FUNCTION NILL)
                                   STARTCOLOR _ (FUNCTION \MAIKO.STARTCOLOR)
                                   STOPCOLOR _ (FUNCTION \MAIKO.STOPCOLOR)
                                   EVENTFN _ (FUNCTION \MAIKOCOLOR.EVENTFN)
                                   SENDCOLORMAPENTRY _ (FUNCTION \MAIKO.SENDCOLORMAPENTRY)
                                   SENDPAGE _ (FUNCTION NILL)
                                   PILOTBITBLT _ (FUNCTION \DISPLAY.PILOTBITBLT)))
    (SETQ \MAIKOCOLORINFO (create DISPLAYINFO
                                  DITYPE _ 'MAIKOCOLOR
                                  DIWIDTH _ \MAIKO.COLORSCREENWIDTH
                                  DIHEIGHT _ \MAIKO.COLORSCREENHEIGHT
                                  DIBITSPERPIXEL _ 8
                                  DIWSOPS _ \MAIKOCOLORWSOPS))
    (\DEFINEDISPLAYINFO \MAIKOCOLORINFO])
```

## (\MAIKO.STARTCOLOR

```
  [LAMBDA (FDEV)                                          ; Edited 26-Oct-2021 10:17 by larry
                                                          ; Edited  2-Nov-88 11:13 by shimizu
    (PROG (DISPLAYSTATE)
          (SETQ DISPLAYSTATE (fetch (FDEV DEVICEINFO) of FDEV))
          (replace (DISPLAYSTATE ONOFF) of DISPLAYSTATE with 'STARTCOLOR)
          (MOVD '\DISPLAY.PILOTBITBLT '\SOFTCURSORPILOTBITBLT)
     ;; MMAP colorbuffer
          (SUBRCALL COLOR-INIT (FETCH (BITMAP BITMAPBASE) OF ColorScreenBitMap))
          (replace (DISPLAYSTATE ONOFF) of DISPLAYSTATE with 'ON])
```

## (\MAIKO.STOPCOLOR

```
  [LAMBDA (FDEV)                                          ; Edited 28-Apr-89 16:51 by tshimizu.fx
                                                          ; By Take
    (PROG (DISPLAYSTATE)
          (SETQ DISPLAYSTATE (fetch (FDEV DEVICEINFO) of FDEV))
          (replace (DISPLAYSTATE ONOFF) of DISPLAYSTATE with 'OFF])
```

## (\MAIKOCOLOR.EVENTFN

```
  [LAMBDA (FDEV EVENT)                                    ; Edited 23-Oct-91 14:18 by jds
    (COND
       ((EQ (fetch (DISPLAYSTATE ONOFF) of (fetch (FDEV DEVICEINFO) of FDEV))
            'ON)
        (SELECTQ EVENT
             ((AFTERSAVEVM AFTERLOGOUT AFTERSYSOUT AFTERMAKESYS)
                  (\MAIKO.STARTCOLOR \COLORDISPLAYFDEV)
                  (SCREENCOLORMAP (SCREENCOLORMAP))
                  (COND
                     ((EQ LASTSCREEN (COLORSCREEN))
                      (CURSORSCREEN (COLORSCREEN)
                             200 200))))
             NIL])
```

## (\MAIKO.SENDCOLORMAPENTRY

```
  [LAMBDA (FDEV COLOR# RGB)                               ; Edited 26-Oct-2021 10:17 by larry
                                                          ; Edited  1-Dec-88 18:16 by shimizu
    (SUBRCALL COLOR-MAP COLOR# (CAR RGB)
           (CADR RGB)
           (CADDR RGB])
```

## (\MAIKO.CHANGESCREEN

```
  [LAMBDA (TOSCREEN)                                      ; Edited 26-Oct-2021 10:18 by larry
                                                          ; Edited  1-Dec-88 18:32 by shimizu
    (SUBRCALL COLOR-SCREENMODE TOSCREEN])

)

(DEFINEQ
```

## (CURSOREXIT

```
  [LAMBDA NIL                                             ; Edited 11-Aug-89 13:16 by takeshi

          (* * called when cursor moves off the screen edge)

    (DECLARE (GLOBALVARS LASTSCREEN LASTMOUSEX LASTMOUSEY \MAIKO.CURRENT.SCREEN.MODE))
    (PROG (SCREEN XCOORD YCOORD SCREEN2 XCOORD2 YCOORD2)
          (SETQ SCREEN LASTSCREEN)
          (SETQ XCOORD LASTMOUSEX)
          (SETQ YCOORD LASTMOUSEY)
          [SETQ SCREEN2 (COND
                           ((EQ SCREEN \MAINSCREEN)
                            (PROGN \COLORSCREEN))
                           (T (PROGN \MAINSCREEN]                (* generalize for more than two screens
                                                                    (or alternate physical arrangement of screens.))
          (COND
             ((EQ XCOORD 0)
              (SETQ XCOORD2 (IDIFFERENCE (fetch (SCREEN SCWIDTH) of SCREEN2)
                                         2)))
             ((EQ XCOORD (SUB1 (fetch (SCREEN SCWIDTH) of SCREEN)))
              (SETQ XCOORD2 1))
             (T (RETURN)))
          [SETQ YCOORD2 (IQUOTIENT (ITIMES YCOORD (SUB1 (fetch (SCREEN SCHEIGHT) of SCREEN2)))
                                   (SUB1 (fetch (SCREEN SCHEIGHT) of SCREEN]
          (CURSORSCREEN SCREEN2 XCOORD2 YCOORD2])
```

## (CURSORSCREEN

```
  [LAMBDA (SCREEN XCOORD YCOORD)                          ; Edited 19-Jun-90 16:33 by matsuda

          (* * sets up SCREEN to be the current screen, XCOORD %, YCOORD is initial pos of cursor on SCREEN)
```

```
    (COND
       ((NULL XCOORD)
        (SETQ XCOORD 0)))
    (COND
       ((NULL YCOORD)
        (SETQ YCOORD 0)))
    (PROG (DESTINATION)
          (SETQ DESTINATION (fetch (SCREEN SCDESTINATION) of SCREEN))
          (\CURSORDOWN)
          (SETQ \CURSORSCREEN SCREEN)
          (\CURSORDESTINATION DESTINATION)
          (\CURSORUP \CURRENTCURSOR)
          (\CURSORPOSITION XCOORD YCOORD)
          (AND (EQUAL (MACHINETYPE)
                      'MAIKO)
               (COND
                  ((EQ (fetch (SCREEN SCBITSPERPIXEL) of SCREEN)
                       1)
                   (SETQ \COLOR.PROMPTWINDOW PROMPTWINDOW)
                   (\MAIKO.CHANGESCREEN \TO.MAIKO.MONOSCREEN)
                   (SETQ PROMPTWINDOW \MONO.PROMPTWINDOW))
                  (T (SETQ \MONO.PROMPTWINDOW PROMPTWINDOW)
                     (\MAIKO.CHANGESCREEN \TO.MAIKO.COLORSCREEN)
                     (SETQ PROMPTWINDOW (OR \COLOR.PROMPTWINDOW (PROG1 (SETQ W
                                                                        (CREATEW '(0 800 370 80)
                                                                                 "Prompt Window" 2))
                                                                       (SETQ DISPLAYDATA (FETCH IMAGEDATA
                                                                                                OF (FETCH (WINDOW
                                                                                                          DSP)
                                                                                                          OF W)))
                                                                       (REPLACE DDOPERATION OF DISPLAYDATA
                                                                          WITH 'ERASE)
                                                                       (REPLACE DDTexture OF DISPLAYDATA
                                                                          WITH 65535)
                                                                       (CLEARW W))])))
```

## (**WARPCURSOR**

```
  [LAMBDA (ENABLE)                                          ; Edited 20-Jul-90 19:02 by matsuda
    (COND
       (ENABLE (MOVD 'SAVE.CURSOREXIT 'CURSOREXIT)
               T)
       (T (MOVD 'NILL 'CURSOREXIT)
          NIL])
```

## (**\SLOWBLTCHAR**

```
  [LAMBDA (CHARCODE DISPLAYSTREAM)                          ; Edited 26-Oct-2021 10:19 by larry
                                                            ; Edited  7-Jun-90 14:06 by matsuda
    (SUBRCALL C-SlowBltChar CHARCODE DISPLAYSTREAM])
```

## (**\SOFTCURSORUP**

```
  [LAMBDA (NEWCURSOR)                                       ; Edited 16-Jan-89 15:44 by shimizu
                                                            (* Put soft NEWCURSOR up, assuming soft cursor is down.
                                                            *)

    (COND
       ((EQ \MACHINETYPE \MAIKO)
        (SETQ \CURRENTCURSOR NEWCURSOR))
       (T (PROG (IMAGE MASK WIDTH BWIDTH HEIGHT CURSORBITSPERPIXEL CURSORBPL UPBMBASE DOWNBMBASE)
                                                                        (* Get cursor IMAGE & MASK. *)
                (SETQ IMAGE (fetch (CURSOR CUIMAGE) of NEWCURSOR))
                (SETQ MASK (fetch (CURSOR CUMASK) of NEWCURSOR))
                (SETQ WIDTH (fetch (BITMAP BITMAPWIDTH) of IMAGE))
                (SETQ HEIGHT (fetch (BITMAP BITMAPHEIGHT) of IMAGE))
                (SETQ CURSORBITSPERPIXEL (fetch (BITMAP BITMAPBITSPERPIXEL) of IMAGE))
                                                                        (* Create new UPBM & DOWNBM caches if necessary.
                                                                        *)
                (COND
                   ((NOT (AND (type? BITMAP \SOFTCURSORUPBM)
                              (EQ (fetch (BITMAP BITMAPWIDTH) of \SOFTCURSORUPBM)
                                  WIDTH)
                              (EQ (fetch (BITMAP BITMAPHEIGHT) of \SOFTCURSORUPBM)
                                  HEIGHT)
                              (EQ (fetch (BITMAP BITMAPBITSPERPIXEL) of \SOFTCURSORUPBM)
                                  CURSORBITSPERPIXEL)))
                    (SETQ \SOFTCURSORWIDTH WIDTH)
                    (SETQ \SOFTCURSORHEIGHT HEIGHT)
                    (SETQ \SOFTCURSORUPBM (BITMAPCREATE WIDTH HEIGHT CURSORBITSPERPIXEL))
                    (SETQ \SOFTCURSORDOWNBM (BITMAPCREATE WIDTH HEIGHT CURSORBITSPERPIXEL))
                    (SETQ UPBMBASE (fetch (BITMAP BITMAPBASE) of \SOFTCURSORUPBM))
                    (\TEMPLOCKPAGES UPBMBASE 1)
                    (SETQ DOWNBMBASE (fetch (BITMAP BITMAPBASE) of \SOFTCURSORDOWNBM))
                    (\TEMPLOCKPAGES DOWNBMBASE 1)
                    (SETQ CURSORBPL (UNFOLD (fetch (BITMAP BITMAPRASTERWIDTH) of IMAGE)
                                            BITSPERWORD))
                    (SETQ BWIDTH (ITIMES (fetch (BITMAP BITMAPWIDTH) of IMAGE)
```

```
                                    (fetch (BITMAP BITMAPBITSPERPIXEL) of IMAGE)))
                (replace (PILOTBBT PBTDESTBPL) of \SOFTCURSORBBT1 with CURSORBPL)
                (replace (PILOTBBT PBTDEST) of \SOFTCURSORBBT2 with UPBMBASE)
                (replace (PILOTBBT PBTDESTBPL) of \SOFTCURSORBBT2 with CURSORBPL)
                (replace (PILOTBBT PBTSOURCE) of \SOFTCURSORBBT2 with DOWNBMBASE)
                (replace (PILOTBBT PBTSOURCEBPL) of \SOFTCURSORBBT2 with CURSORBPL)
                (replace (PILOTBBT PBTWIDTH) of \SOFTCURSORBBT2 with BWIDTH)
                (replace (PILOTBBT PBTHEIGHT) of \SOFTCURSORBBT2 with HEIGHT)
                (replace (PILOTBBT PBTDEST) of \SOFTCURSORBBT3 with UPBMBASE)
                (replace (PILOTBBT PBTDESTBPL) of \SOFTCURSORBBT3 with CURSORBPL)
                (replace (PILOTBBT PBTSOURCEBPL) of \SOFTCURSORBBT3 with CURSORBPL)
                (replace (PILOTBBT PBTWIDTH) of \SOFTCURSORBBT3 with BWIDTH)
                (replace (PILOTBBT PBTHEIGHT) of \SOFTCURSORBBT3 with HEIGHT)
                (replace (PILOTBBT PBTDEST) of \SOFTCURSORBBT4 with UPBMBASE)
                (replace (PILOTBBT PBTDESTBPL) of \SOFTCURSORBBT4 with CURSORBPL)
                (replace (PILOTBBT PBTSOURCEBPL) of \SOFTCURSORBBT4 with CURSORBPL)
                (replace (PILOTBBT PBTWIDTH) of \SOFTCURSORBBT4 with BWIDTH)
                (replace (PILOTBBT PBTHEIGHT) of \SOFTCURSORBBT4 with HEIGHT)
                (replace (PILOTBBT PBTSOURCEBPL) of \SOFTCURSORBBT5 with CURSORBPL)
                (replace (PILOTBBT PBTSOURCEBPL) of \SOFTCURSORBBT6 with CURSORBPL)))
                                                        (* Change PILOTBBTs. *)
            (replace (PILOTBBT PBTSOURCE) of \SOFTCURSORBBT3 with (fetch (BITMAP BITMAPBASE) of MASK))
            (replace (PILOTBBT PBTSOURCE) of \SOFTCURSORBBT4 with (fetch (BITMAP BITMAPBASE) of IMAGE))
                                                        (* Put up new \CURRENTCURSOR.
                                                        *)
            (SETQ \CURRENTCURSOR NEWCURSOR)
            (\TEMPLOCKPAGES \CURRENTCURSOR 1)
            (SETQ \SOFTCURSORP T)
            (\SOFTCURSORUPCURRENT])
```

## (\BITBLT.DISPLAY

```
[LAMBDA (SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTSTRM DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT
          SOURCETYPE OPERATION TEXTURE CLIPPINGREGION CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM)
                                                        ; Edited 24-Jan-91 11:57 by matsuda
(DECLARE (LOCALVARS . T))
(DECLARE (GLOBALVARS \SYSPILOTBBT \SCREENBITMAPS \BBSCRATCHTEXTURE \SOFTCURSORP \SOFTCURSORUPP
                \CURSORDESTINATION))
(PROG (stodx stody left top bottom right DESTDD DESTBITMAP DESTINATIONNBITS SOURCENBITS MAXSHADE)
      (SETQ DESTDD (fetch (STREAM IMAGEDATA) of DESTSTRM))
      (SETQ DESTBITMAP (fetch (\DISPLAYDATA DDDestination) of DESTDD))
```

;; bring it to top so that its TOTOPFNs will get called before the destination information is cached in case one of them moves, reshapes, etc. the ;; window

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

```
      (\INSURETOPWDS DESTSTRM)
      (SETQ DESTINATIONLEFT (\DSPTRANSFORMX DESTINATIONLEFT DESTDD))
      (SETQ DESTINATIONBOTTOM (\DSPTRANSFORMY DESTINATIONBOTTOM DESTDD))
      [PROGN                                            ; compute limits based on clipping regions.
             (SETQ left (fetch (\DISPLAYDATA DDClippingLeft) of DESTDD))
             (SETQ bottom (ffetch (\DISPLAYDATA DDClippingBottom) of DESTDD))
             (SETQ right (ffetch (\DISPLAYDATA DDClippingRight) of DESTDD))
             (SETQ top (ffetch (\DISPLAYDATA DDClippingTop) of DESTDD))
             (COND
                (CLIPPINGREGION                         ; hard case, two destination clipping regions: do calculations to
                                                        ; merge them.
                       (PROG (CRLEFT CRBOTTOM)
                             [SETQ left (IMAX left (SETQ CRLEFT (\DSPTRANSFORMX (fetch (REGION LEFT)
                                                                                   of CLIPPINGREGION)
                                                                                DESTDD]
                             [SETQ bottom (IMAX bottom (SETQ CRBOTTOM (\DSPTRANSFORMY (ffetch (REGION BOTTOM)
                                                                                         of CLIPPINGREGION)
                                                                                      DESTDD]
                             [SETQ right (IMIN right (IPLUS CRLEFT (ffetch (REGION WIDTH) of CLIPPINGREGION]
                             (SETQ top (IMIN top (IPLUS CRBOTTOM (fetch (REGION HEIGHT) of CLIPPINGREGION]
      (SETQ DESTINATIONNBITS (BITSPERPIXEL DESTBITMAP))
      (SETQ SOURCENBITS (BITSPERPIXEL SOURCEBITMAP))
      [COND
         ((NOT (EQ SOURCENBITS DESTINATIONNBITS))
          (COND
             ((EQ SOURCENBITS 1)
              (SETQ SOURCEBITMAP (COLORIZEBITMAP SOURCEBITMAP (ffetch DDBACKGROUNDCOLOR of DESTDD)
                                          (ffetch DDFOREGROUNDCOLOR of DESTDD)
                                          DESTINATIONNBITS)))
             [(EQ DESTINATIONNBITS 1)
              (SETQ SOURCEBITMAP (UNCOLORIZEBITMAP SOURCEBITMAP (COLORMAP DESTINATIONNBITS]
             (T ;; Between two color bitmaps with different bpp.  It seems that NOP is better than breaking.  Eventually do some kind of
                ;; output here, but don't error now.
                (RETURN]
```

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

```
      [PROGN (SETQ left (IMAX DESTINATIONLEFT left))
             (SETQ bottom (IMAX DESTINATIONBOTTOM bottom))
             [COND
                (WIDTH                                  ; WIDTH is optional
```

```
                        (SETQ right (IMIN (IPLUS DESTINATIONLEFT WIDTH)
                                          right]
                  (COND
                     (HEIGHT                                          ; HEIGHT is optional
                            (SETQ top (IMIN (IPLUS DESTINATIONBOTTOM HEIGHT)
                                            top]                      ; Clip and translate coordinates.
               (SETQ stodx (IDIFFERENCE DESTINATIONLEFT SOURCELEFT))
               (SETQ stody (IDIFFERENCE DESTINATIONBOTTOM SOURCEBOTTOM))
```

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

```
               [PROGN                                                 ; compute left margin
                      (SETQ left (IMAX CLIPPEDSOURCELEFT (IDIFFERENCE left stodx)
                                       0))                          ; compute bottom margin
                      (SETQ bottom (IMAX CLIPPEDSOURCEBOTTOM (IDIFFERENCE bottom stody)
                                         0))                        ; compute right margin
                      (SETQ right (IMIN (BITMAPWIDTH SOURCEBITMAP)
                                        (IDIFFERENCE right stodx)
                                        (IPLUS CLIPPEDSOURCELEFT WIDTH)))
                                                                     ; compute top margin
                      (SETQ top (IMIN (BITMAPHEIGHT SOURCEBITMAP)
                                      (IDIFFERENCE top stody)
                                      (IPLUS CLIPPEDSOURCEBOTTOM HEIGHT]
               (COND
                  ((OR (ILEQ right left)
                       (ILEQ top bottom))                            ; there is nothing to move.
                   (RETURN)))
               (OR OPERATION (SETQ OPERATION (ffetch (\DISPLAYDATA DDOPERATION) of DESTDD)))
               (SETQ MAXSHADE (MAXIMUMSHADE DESTINATIONNBITS))
               (SELECTQ SOURCETYPE
                    (MERGE                                           ; Need to use complement of TEXTURE
                           [COND
                              ((AND (LISTP TEXTURE)
                                    (EQ DESTINATIONNBITS 1))         ; either a color or a (texture color) filling.
                               (SETQ TEXTURE (INSURE.B&W.TEXTURE TEXTURE]
                           [SETQ TEXTURE (COND
                                         ((NULL TEXTURE)
                                          MAXSHADE)
                                         ((FIXP TEXTURE)
                                          (LOGXOR (LOGAND TEXTURE MAXSHADE)
                                                  MAXSHADE))
                                         [(type? BITMAP TEXTURE)
                                          (INVERT.TEXTURE.BITMAP TEXTURE (OR \BBSCRATCHTEXTURE (SETQ
                                                                                                \BBSCRATCHTEXTURE
                                                                                                (BITMAPCREATE
                                                                                                 16 16]
                                         ((NOT (EQ DESTINATIONNBITS 1))
                                          (COLORNUMBERP TEXTURE DESTINATIONNBITS))
                                         (T (\ILLEGAL.ARG TEXTURE]
                           [COND
                              ((NOT (EQ DESTINATIONNBITS 1))
                               (SETQ TEXTURE (COLORTEXTUREFROMCOLOR# TEXTURE DESTINATIONNBITS])
                    (TEXTURE [COND
                               ((EQ DESTINATIONNBITS 1)              ; either a color or a (texture color) filling.
                                (SETQ TEXTURE (INSURE.B&W.TEXTURE TEXTURE])
                    NIL)
               [COND
                  ((AND (NOT (EQ DESTINATIONNBITS 1))
                        (NOT (type? BIGBM SOURCEBITMAP))
                        (NOT (type? BIGBM DESTBITMAP)))
                   (SETQ left (ITIMES DESTINATIONNBITS left))
                   (SETQ right (ITIMES DESTINATIONNBITS right))
                   (SETQ stodx (ITIMES DESTINATIONNBITS stodx]
               [.WHILE.TOP.DS. DESTSTRM
                      (COND
                         [(AND (NOT (type? BIGBM SOURCEBITMAP))
                               (NOT (type? BIGBM DESTBITMAP)))
                          (PROG (HEIGHT WIDTH DTY DLX STY SLX)
                                (SETQ HEIGHT (IDIFFERENCE top bottom))
                                (SETQ WIDTH (IDIFFERENCE right left))
                                (SETQ DTY (\SFInvert DESTBITMAP (IPLUS top stody)))
                                (SETQ DLX (IPLUS left stodx))
                                (SETQ STY (\SFInvert SOURCEBITMAP top))
                                (SETQ SLX left)
                                (replace (PILOTBBT PBTWIDTH) of \SYSPILOTBBT with WIDTH)
                                (replace (PILOTBBT PBTHEIGHT) of \SYSPILOTBBT with HEIGHT)
                                (COND
                                   ((EQ SOURCETYPE 'MERGE)
                                    (\BITBLT.MERGE \SYSPILOTBBT SOURCEBITMAP SLX STY DESTBITMAP DLX DTY WIDTH HEIGHT
                                           OPERATION TEXTURE))
                                   (T (\BITBLTSUB \SYSPILOTBBT SOURCEBITMAP SLX STY DESTBITMAP DLX DTY HEIGHT
                                             SOURCETYPE OPERATION TEXTURE]
                         (T (PROG (HEIGHT WIDTH DBY DLX SBY SLX)
                                  (SETQ HEIGHT (IDIFFERENCE top bottom))
                                  (SETQ WIDTH (IDIFFERENCE right left))
                                  (SETQ DBY (IPLUS bottom stody))
                                  (SETQ DLX (IPLUS left stodx))
```

```
                              (SETQ SBY bottom)
                              (SETQ SLX left)
                              (BITBLT.BIGBM SOURCEBITMAP SLX SBY DESTBITMAP DLX DBY WIDTH HEIGHT SOURCETYPE
                                     OPERATION TEXTURE]
          (RETURN T])

)

;; these FNS defs. will be moved to original files,later

(DEFINEQ

(\PUNT.SLOWBLTCHAR
  [LAMBDA (CHARCODE DISPLAYSTREAM)                                    ; Edited 2-Jul-90 14:23 by matsuda
    ;; case of BLTCHAR where either font is rotated or destination is a color bitmap.  DISPLAYSTREAM is known to be a display stream, and its cache
    ;; fields have been updated for CHARCODE's charset
    (PROG (ROTATION CHAR8CODE DD FONTDESC)
          (SETQ CHAR8CODE (\CHAR8CODE CHARCODE))
          (SETQ DD (ffetch (STREAM IMAGEDATA) of DISPLAYSTREAM))
          (SETQ FONTDESC (ffetch (\DISPLAYDATA DDFONT) of DD))
          (SETQ ROTATION (ffetch (FONTDESCRIPTOR ROTATION) of FONTDESC))
          (COND
             ((EQ 0 ROTATION)
              (PROG (NEWX LEFT RIGHT CURX PILOTBBT DESTBIT WIDTH SOURCEBIT CSINFO)
                    (SETQ CSINFO (\GETCHARSETINFO (\CHARSET CHARCODE)
                                       (ffetch (\DISPLAYDATA DDFONT) of DD)))
                    (SETQ CURX (ffetch (\DISPLAYDATA DDXPOSITION) of DD))
                    (SETQ NEWX (IPLUS CURX (\DSPGETCHARWIDTH CHAR8CODE DD)))
                    [COND
                       ((IGREATERP NEWX (ffetch (\DISPLAYDATA DDRightMargin) of DD))
                                                                      ; past RIGHT margin, force eol
                        (\DSPPRINTCR/LF (CHARCODE EOL)
                               DISPLAYSTREAM)
                        (SETQ CURX (ffetch (\DISPLAYDATA DDXPOSITION) of DD))
                        (SETQ NEWX (IPLUS CURX (\DSPGETCHARWIDTH CHAR8CODE DD]
                                                                      ; update the x position.
                    (freplace (\DISPLAYDATA DDXPOSITION) of DD with NEWX)
                                                                      (* SETQ CURX (\DSPTRANSFORMX CURX DD))
                    (SETQ LEFT (IMAX (ffetch (\DISPLAYDATA DDClippingLeft) of DD)
                                     CURX))
                    (SETQ RIGHT (IMIN (ffetch (\DISPLAYDATA DDClippingRight) of DD)
                                      (\DSPTRANSFORMX NEWX DD)))
                    (BITBLT (ffetch (CHARSETINFO CHARSETBITMAP) of CSINFO)
                           (\DSPGETCHAROFFSET CHAR8CODE DD)
                           0 DISPLAYSTREAM CURX (IDIFFERENCE (ffetch (\DISPLAYDATA DDYPOSITION) of DD)
                                                             (ffetch (CHARSETINFO CHARSETDESCENT) of CSINFO))
                           (\DSPGETCHARWIDTH CHAR8CODE DD)
                           (IPLUS (ffetch (CHARSETINFO CHARSETASCENT) of CSINFO)
                                  (ffetch (CHARSETINFO CHARSETDESCENT) of CSINFO)))
                                                                      ; (SETQ PILOTBBT (|ffetch| (\\DISPLAYDATA DDPILOTBBT)
                                                                      ; |of| DD)) (COND ((AND (ILESSP LEFT RIGHT) (NOT (EQ
                                                                      ; (|ffetch| (PILOTBBT PBTHEIGHT) |of| PILOTBBT) 0))) (SETQ
                                                                      ; DESTBIT LEFT) (SETQ WIDTH (IDIFFERENCE RIGHT LEFT))
                                                                      ; (SETQ SOURCEBIT (IDIFFERENCE (IPLUS
                                                                      ; (\\DSPGETCHAROFFSET CHAR8CODE DD) LEFT) CURX))
                                                                      ; (SELECTQ (|ffetch| (BITMAP BITMAPBITSPERPIXEL) |of|
                                                                      ; (|ffetch| (\\DISPLAYDATA |DDDestination|) |of| DD)) (1) (4
                                                                      ; (SETQ DESTBIT (LLSH DESTBIT 2)) (SETQ WIDTH (LLSH
                                                                      ; WIDTH 2)) (SETQ SOURCEBIT (LLSH SOURCEBIT 2))) (8
                                                                      ; (SETQ DESTBIT (LLSH DESTBIT 3)) (SETQ WIDTH (LLSH
                                                                      ; WIDTH 3)) (SETQ SOURCEBIT (LLSH SOURCEBIT 3))) (24
                                                                      ; (SETQ DESTBIT (ITIMES 24 DESTBIT)) (SETQ WIDTH
                                                                      ; (ITIMES 24 WIDTH)) (SETQ SOURCEBIT (ITIMES 24
                                                                      ; SOURCEBIT))) (SHOULDNT)) (.WHILE.TOP.DS.
                                                                      ; DISPLAYSTREAM (|freplace| (PILOTBBT PBTDESTBIT) |of|
                                                                      ; PILOTBBT |with| DESTBIT) (|freplace| (PILOTBBT PBTWIDTH)
                                                                      ; |of| PILOTBBT |with| WIDTH) (|freplace| (PILOTBBT
                                                                      ; PBTSOURCEBIT) |of| PILOTBBT |with| SOURCEBIT)
                                                                      ; (\\PILOTBITBLT PILOTBBT 0)) T))

                    ))
             (T                                                       ; handle rotated fonts
                (PROG (YPOS HEIGHTMOVED CSINFO)
                      (SETQ YPOS (ffetch (\DISPLAYDATA DDYPOSITION) of DD))
                      (SETQ HEIGHTMOVED (\DSPGETCHARWIDTH CHAR8CODE DD))
                      (SETQ CSINFO (\GETCHARSETINFO (\CHARSET CHARCODE)
                                         (ffetch (\DISPLAYDATA DDFONT) of DD)))
                      (COND
                         ((EQ ROTATION 90)                            ; don't force CR for rotated fonts.
                          (\DSPYPOSITION.DISPLAY DISPLAYSTREAM (IPLUS YPOS HEIGHTMOVED))
                                                                      ; update the display stream x position.
                          (BITBLT (ffetch (CHARSETINFO CHARSETBITMAP) of CSINFO)
                                 0
                                 (\DSPGETCHAROFFSET CHAR8CODE DD)
                                 DISPLAYSTREAM
                                 (ADD1 (IDIFFERENCE (ffetch (\DISPLAYDATA DDXPOSITION) of DD)
```

```
                                        (ffetch (CHARSETINFO CHARSETASCENT) of CSINFO)))
                                YPOS
                                (IPLUS (ffetch (CHARSETINFO CHARSETASCENT) of CSINFO)
                                       (ffetch (CHARSETINFO CHARSETDESCENT) of CSINFO))
                                HEIGHTMOVED))
                        ((EQ ROTATION 270)
                         (\DSPYPOSITION.DISPLAY DISPLAYSTREAM (IDIFFERENCE YPOS HEIGHTMOVED))
                         (BITBLT (ffetch (CHARSETINFO CHARSETBITMAP) of CSINFO)
                                0
                                (\GETBASE (ffetch (\DISPLAYDATA DDOFFSETSCACHE) of DD)
                                       CHAR8CODE)
                                DISPLAYSTREAM
                                (IDIFFERENCE (ffetch (\DISPLAYDATA DDXPOSITION) of DD)
                                       (ffetch (CHARSETINFO CHARSETDESCENT) of CSINFO))
                                (ffetch (\DISPLAYDATA DDYPOSITION) of DISPLAYSTREAM)
                                (IPLUS (ffetch (CHARSETINFO CHARSETASCENT) of CSINFO)
                                       (ffetch (CHARSETINFO CHARSETDESCENT) of CSINFO))
                                HEIGHTMOVED))
                        (T (ERROR "Not implemented to rotate by other than 0, 90 or 270"])
```

## (\MAIKO.PUNTBLTCHAR

```
  [LAMBDA (CHARCODE DISPLAYSTREAM DISPLAYDATA)                    ; Edited 26-Oct-2021 10:21 by larry
                                                                  ; Edited  1-Nov-89 15:26 by takeshi

    ;; puts a character on a display stream. This function will be called when \maiko.bltchar failed. Punt from subr call

    (DECLARE (LOCALVARS . T))
    (PROG (LOCAL1 RIGHT LEFT CURX CHAR8CODE)
          (SETQ CHAR8CODE (\CHAR8CODE CHARCODE))
      CRLP
          [COND
             ((NOT (EQ (ffetch (\DISPLAYDATA DDCHARSET) of DISPLAYDATA)
                       (\CHARSET CHARCODE)))
              (\CHANGECHARSET.DISPLAY DISPLAYDATA (\CHARSET CHARCODE]
          [COND
             ((ffetch (\DISPLAYDATA DDSlowPrintingCase) of DISPLAYDATA)
              (RETURN (COND
                         ((type? STREAM DISPLAYSTREAM)
                          (\SLOWBLTCHAR CHARCODE DISPLAYSTREAM))
                         ((type? WINDOW DISPLAYSTREAM)
                          (\SLOWBLTCHAR CHARCODE (FETCH DSP OF DISPLAYSTREAM)))
                         (T (ERROR "Not Stream or Window" DISPLAYSTREAM]
          (SETQ CURX (ffetch (\DISPLAYDATA DDXPOSITION) of DISPLAYDATA))
          (SETQ RIGHT (IPLUS CURX (\DSPGETCHARIMAGEWIDTH CHAR8CODE DISPLAYDATA)))
          [COND
             ((IGREATERP RIGHT (ffetch (\DISPLAYDATA DDRightMargin) of DISPLAYDATA))
                                                                  ; would go past right margin, force a cr
              (COND
                 ((IGREATERP CURX (ffetch (\DISPLAYDATA DDLeftMargin) of DISPLAYDATA))
                                                                  ; don't bother CR if position is at left margin anyway.  This also
                                                                  ; serves to break the loop.
                  (\DSPPRINTCR/LF (CHARCODE EOL)
                         DISPLAYSTREAM)                           ; reuse the code in the test of this conditional rather than repeat it
                                                                  ; here.
                  (GO CRLP]                                       ; update the display stream x position.
          (freplace (\DISPLAYDATA DDXPOSITION) of DISPLAYDATA with (IPLUS CURX (\DSPGETCHARWIDTH CHAR8CODE
                                                                                DISPLAYDATA)))
                                                                  ; transforms an x coordinate into the destination coordinate.
          (SETQ LOCAL1 (ffetch (\DISPLAYDATA DDXOFFSET) of DISPLAYDATA))
          (SETQ CURX (IPLUS CURX LOCAL1))
          (SETQ RIGHT (IPLUS RIGHT LOCAL1))
          (COND
             ((IGREATERP RIGHT (SETQ LOCAL1 (ffetch (\DISPLAYDATA DDClippingRight) of DISPLAYDATA)))
                                                                  ; character overlaps right edge of clipping region.
              (SETQ RIGHT LOCAL1)))
          (SETQ LEFT (COND
                        ((IGREATERP CURX (SETQ LOCAL1 (ffetch (\DISPLAYDATA DDClippingLeft) of DISPLAYDATA)))
                         CURX)
                        (T LOCAL1)))
          (RETURN (COND
                     ((AND (ILESSP LEFT RIGHT)
                           (NOT (EQ (fetch (PILOTBBT PBTHEIGHT) of (SETQ LOCAL1 (ffetch (\DISPLAYDATA DDPILOTBBT)
                                                                                  of DISPLAYDATA)))
                                    0)))
                      (.WHILE.TOP.DS. DISPLAYSTREAM (SUBRCALL BLTCHAR LOCAL1 DISPLAYDATA CHAR8CODE CURX LEFT
                                                              RIGHT))
                      T])
```

## (\MAIKO.BLTCHAR

```
  [LAMBDA (CHARCODE DISPLAYSTREAM DISPLAYDATA)                    ; Edited 26-Oct-2021 10:22 by larry
                                                                  ; Edited  6-Jul-90 10:14 by matsuda
    (SUBRCALL NEW-BLTCHAR CHARCODE DISPLAYSTREAM DISPLAYDATA])

)

(DEFINEQ
```

## (\PUNT.BLTSHADE.BITMAP

```
[LAMBDA (TEXTURE DESTINATIONBITMAP DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT OPERATION CLIPPINGREGION)
                                                        ; Edited  5-Jun-90 12:12 by Takeshi

   ;; This FNS is for a punt case of \BLTSHADE.BITMAP which is implemeted in C   ;  Stolen from old definition of \BLTSHADE.BITMAP
   (DECLARE (LOCALVARS . T))
   (PROG (left bottom top right DESTINATIONNBITS)
         (SETQ left 0)
         (SETQ bottom 0)
         (SETQ top (fetch (BITMAP BITMAPHEIGHT) of DESTINATIONBITMAP))
         (SETQ right (fetch (BITMAP BITMAPWIDTH) of DESTINATIONBITMAP))
         (SETQ DESTINATIONNBITS (fetch (BITMAP BITMAPBITSPERPIXEL) of DESTINATIONBITMAP))
         (COND
            ((EQ DESTINATIONNBITS 1)                    ; DESTINATIONNBITS is NIL for the case of 1 bit per pixel.
             (SETQ DESTINATIONNBITS NIL)))
         [COND
            (CLIPPINGREGION                             ; adjust limits
                   (SETQ left (IMAX left (fetch (REGION LEFT) of CLIPPINGREGION)))
                   (SETQ bottom (IMAX bottom (fetch (REGION BOTTOM) of CLIPPINGREGION)))
                   [SETQ right (IMIN right (IPLUS (fetch (REGION WIDTH) of CLIPPINGREGION)
                                                  (fetch (REGION LEFT) of CLIPPINGREGION]
                   (SETQ top (IMIN top (IPLUS (fetch (REGION BOTTOM) of CLIPPINGREGION)
                                              (fetch (REGION HEIGHT) of CLIPPINGREGION]
         (OR DESTINATIONLEFT (SETQ DESTINATIONLEFT 0))
         (OR DESTINATIONBOTTOM (SETQ DESTINATIONBOTTOM 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.

         [PROGN (SETQ left (IMAX DESTINATIONLEFT left))
                (SETQ bottom (IMAX DESTINATIONBOTTOM bottom))
                [COND
                   (WIDTH                               ; WIDTH is optional
                          (SETQ right (IMIN (IPLUS DESTINATIONLEFT WIDTH)
                                            right]
                (COND
                   (HEIGHT                              ; HEIGHT is optional
                          (SETQ top (IMIN (IPLUS DESTINATIONBOTTOM HEIGHT)
                                          top]
         (COND
            ((OR (ILEQ right left)
                 (ILEQ top bottom))                     ; there is nothing to move.
             (RETURN)))
         (SETQ TEXTURE (SELECTQ (TYPENAME TEXTURE)
                           (LITATOM                     ; includes NIL case
                                    (COND
                                       [DESTINATIONNBITS (COND
                                                            (TEXTURE
                                                               ; should be a color name
                                                                 (OR (COLORNUMBERP TEXTURE DESTINATIONNBITS
                                                                                   T)
                                                                     (\ILLEGAL.ARG TEXTURE)))
                                                            (T (MAXIMUMCOLOR DESTINATIONNBITS]
                                       (TEXTURE (\ILLEGAL.ARG TEXTURE))
                                       (T WHITESHADE)))
                           ((SMALLP FIXP)
                                (COND
                                   [DESTINATIONNBITS
                                          ;; if fixp use the low order bits as a color number.  This picks up the case of BLACKSHADE
                                          ;; being used to INVERT.
                                          (OR (COLORNUMBERP TEXTURE DESTINATIONNBITS T)
                                              (LOGAND TEXTURE (MAXIMUMCOLOR DESTINATIONNBITS]
                                   (T (LOGAND TEXTURE BLACKSHADE))))
                           (BITMAP TEXTURE)
                           (LISTP                       ; can be a list of (TEXTURE COLOR) or a list of levels rgb or hls.
                                  (COND
                                     [DESTINATIONNBITS
                                             ;; color case: If it is a color, use it;  if it is a list that contains a color, use that;  otherwise,
                                             ;; use the texture
                                             (COND
                                                ((COLORNUMBERP TEXTURE))
                                                [(COLORNUMBERP (CAR (LISTP (CDR TEXTURE]
                                                ((FIXP (CAR TEXTURE))
                                                 (LOGAND (CAR TEXTURE)
                                                         (MAXIMUMCOLOR DESTINATIONNBITS)))
                                                ((TEXTUREP (CAR TEXTURE)))
                                                (T (\ILLEGAL.ARG TEXTURE]
                                     ((TEXTUREP (CAR TEXTURE)))
                                     ((COLORNUMBERP TEXTURE)
                                      (TEXTUREOFCOLOR TEXTURE))
                                     (T (\ILLEGAL.ARG TEXTURE))))
                           (\ILLEGAL.ARG TEXTURE)))      ; filling an area with a texture.
         [COND
            (DESTINATIONNBITS (SETQ left (ITIMES DESTINATIONNBITS left))
                   (SETQ right (ITIMES DESTINATIONNBITS right))
                   (SETQ TEXTURE (COLORTEXTUREFROMCOLOR# TEXTURE DESTINATIONNBITS]
                                                        ; easy case of black and white bitmap into black and white or
```

```
                                                                ; color to color or texture filling.
        (UNINTERRUPTABLY
            (PROG (HEIGHT)
                  (SETQ HEIGHT (IDIFFERENCE top bottom))
                  (replace (PILOTBBT PBTWIDTH) of \SYSPILOTBBT with (IDIFFERENCE right left))
                  (replace (PILOTBBT PBTHEIGHT) of \SYSPILOTBBT with HEIGHT)
                  (\BITBLTSUB \SYSPILOTBBT NIL left NIL DESTINATIONBITMAP left (\SFInvert DESTINATIONBITMAP
                                                                                        top)
                        HEIGHT
                        'TEXTURE OPERATION TEXTURE)))
        (RETURN T])
```

## (**PUNT.BITBLT.BITMAP**

```
  [LAMBDA (SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTBITMAP DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT
                SOURCETYPE OPERATION TEXTURE CLIPPINGREGION CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM)
                                                                ; Edited 5-Jun-90 11:59 by Takeshi
    ;; This FNS is for a punt case of \BITBLT.BITMAP which is implemeted in C
    ;; Stolen from old definition of \BITBLT.BITMAP
    (DECLARE (LOCALVARS . T))
    (PROG (stodx stody right top DESTINATIONNBITS left bottom SOURCENBITS)
          (SETQ top (fetch (BITMAP BITMAPHEIGHT) of DESTBITMAP))
          (SETQ DESTINATIONNBITS (fetch (BITMAP BITMAPBITSPERPIXEL) of DESTBITMAP))
          (SETQ left 0)
          (SETQ bottom 0)
          (SETQ SOURCENBITS (fetch (BITMAP BITMAPBITSPERPIXEL) of SOURCEBITMAP))
          (SETQ right (fetch (BITMAP BITMAPWIDTH) of DESTBITMAP))
          [COND
             (CLIPPINGREGION                                    ; adjust limits
                    (SETQ left (IMAX left (fetch (REGION LEFT) of CLIPPINGREGION)))
                    (SETQ bottom (IMAX bottom (fetch (REGION BOTTOM) of CLIPPINGREGION)))
                    [SETQ right (IMIN right (IPLUS (fetch (REGION WIDTH) of CLIPPINGREGION)
                                                   (fetch (REGION LEFT) of CLIPPINGREGION]
                    (SETQ top (IMIN top (IPLUS (fetch (REGION BOTTOM) of CLIPPINGREGION)
                                               (fetch (REGION HEIGHT) of CLIPPINGREGION]
    ;; left, right top and bottom are the limits in destination taking into account Clipping Regions.  Clip to region in the arguments of this call.
          [PROGN (SETQ left (IMAX DESTINATIONLEFT left))
                 (SETQ bottom (IMAX DESTINATIONBOTTOM bottom))
                 [COND
                    (WIDTH                                      ; WIDTH is optional
                           (SETQ right (IMIN (IPLUS DESTINATIONLEFT WIDTH)
                                             right]
                 (COND
                    (HEIGHT                                     ; HEIGHT is optional
                           (SETQ top (IMIN (IPLUS DESTINATIONBOTTOM HEIGHT)
                                           top]                 ; Clip and translate coordinates.
          (SETQ stodx (IDIFFERENCE DESTINATIONLEFT SOURCELEFT))
          (SETQ stody (IDIFFERENCE DESTINATIONBOTTOM SOURCEBOTTOM))
    ;; compute the source dimensions (left right bottom top) by intersecting the source bit map, the source area to be moved with the limits of the
    ;; region to be moved in the destination coordinates.
          [PROGN                                                ; compute left margin
                 (SETQ left (IMAX CLIPPEDSOURCELEFT 0 (IDIFFERENCE left stodx)))
                                                                ; compute bottom margin
                 (SETQ bottom (IMAX CLIPPEDSOURCEBOTTOM 0 (IDIFFERENCE bottom stody)))
                                                                ; compute right margin
                 (SETQ right (IMIN (ffetch (BITMAP BITMAPWIDTH) of SOURCEBITMAP)
                                   (IDIFFERENCE right stodx)
                                   (IPLUS CLIPPEDSOURCELEFT WIDTH)))
                                                                ; compute top margin
                 (SETQ top (IMIN (ffetch (BITMAP BITMAPHEIGHT) of SOURCEBITMAP)
                                 (IDIFFERENCE top stody)
                                 (IPLUS CLIPPEDSOURCEBOTTOM HEIGHT]
          (COND
             ((OR (ILEQ right left)
                  (ILEQ top bottom))                            ; there is nothing to move.
              (RETURN)))
          (SELECTQ SOURCETYPE
              (MERGE                                            ; Need to use complement of TEXTURE
                                                                ; MAY NOT WORK FOR COLOR CASE.
                     [SETQ TEXTURE (COND
                                   ((NULL TEXTURE)
                                    BLACKSHADE)
                                   ((FIXP TEXTURE)
                                    (LOGXOR (LOGAND TEXTURE BLACKSHADE)
                                            BLACKSHADE))
                                   ((AND (NOT (EQ DESTINATIONNBITS 1))
                                         (COLORNUMBERP TEXTURE DESTINATIONNBITS)))
                                   [(type? BITMAP TEXTURE)
                                    (INVERT.TEXTURE.BITMAP TEXTURE (OR \BBSCRATCHTEXTURE (SETQ
                                                                                          \BBSCRATCHTEXTURE
                                                                                          (BITMAPCREATE
                                                                                           16 16]
                                   (T (\ILLEGAL.ARG TEXTURE])
              NIL)
```

```
          (COND
             [(EQ SOURCENBITS DESTINATIONNBITS)                ; going from one to another of the same size.
              (SELECTQ DESTINATIONNBITS
                   (4                                          ; use UNFOLD with constant value rather than multiple because
                                                               ; it compiles into opcodes.
                      (SETQ left (UNFOLD left 4))
                      (SETQ right (UNFOLD right 4))
                      (SETQ stodx (UNFOLD stodx 4))            ; set texture if it will ever get looked at.
                      (AND (EQ SOURCETYPE 'MERGE)
                           (SETQ TEXTURE (COLORTEXTUREFROMCOLOR# TEXTURE DESTINATIONNBITS))))
                   (8 (SETQ left (UNFOLD left 8))
                      (SETQ right (UNFOLD right 8))
                      (SETQ stodx (UNFOLD stodx 8))
                      (AND (EQ SOURCETYPE 'MERGE)
                           (SETQ TEXTURE (COLORTEXTUREFROMCOLOR# TEXTURE DESTINATIONNBITS))))
                   (24 (SETQ left (ITIMES left 24))
                       (SETQ right (ITIMES right 24))
                       (SETQ stodx (ITIMES stodx 24))
                       (AND (EQ SOURCETYPE 'MERGE)
                            (SETQ TEXTURE (COLORTEXTUREFROMCOLOR# TEXTURE DESTINATIONNBITS))))
                   NIL)                                        ; easy case of black and white bitmap into black and white or
                                                               ; color to color or texture filling.
              (UNINTERRUPTABLY
                  [PROG (HEIGHT WIDTH DTY DLX STY SLX)
                        (SETQ HEIGHT (IDIFFERENCE top bottom))
                        (SETQ WIDTH (IDIFFERENCE right left))
                        (SETQ DTY (\SFInvert DESTBITMAP (IPLUS top stody)))
                        (SETQ DLX (IPLUS left stodx))
                        (SETQ STY (\SFInvert SOURCEBITMAP top))
                        (SETQ SLX left)
                        (replace (PILOTBBT PBTWIDTH) of \SYSPILOTBBT with WIDTH)
                        (replace (PILOTBBT PBTHEIGHT) of \SYSPILOTBBT with HEIGHT)
                        (COND
                           ((EQ SOURCETYPE 'MERGE)
                            (\BITBLT.MERGE \SYSPILOTBBT SOURCEBITMAP SLX STY DESTBITMAP DLX DTY WIDTH HEIGHT
                                   OPERATION TEXTURE))
                           (T (\BITBLTSUB \SYSPILOTBBT SOURCEBITMAP SLX STY DESTBITMAP DLX DTY HEIGHT SOURCETYPE
                                    OPERATION TEXTURE]))]
             [(EQ SOURCENBITS 1)                               ; going from a black and white bitmap to a color map
              (AND SOURCETYPE (NOT (EQ SOURCETYPE 'INPUT))
                   (ERROR "SourceType not implemented from B&W to color bitmaps." SOURCETYPE))
              (PROG (HEIGHT WIDTH DBOT DLFT)
                    (SETQ HEIGHT (IDIFFERENCE top bottom))
                    (SETQ WIDTH (IDIFFERENCE right left))
                    (SETQ DBOT (IPLUS bottom stody))
                    (SETQ DLFT (IPLUS left stodx))
                    (SELECTQ OPERATION
                         ((NIL REPLACE)
                             (\BWTOCOLORBLT SOURCEBITMAP left bottom DESTBITMAP DLFT DBOT WIDTH HEIGHT 0
                                    (MAXIMUMCOLOR DESTINATIONNBITS)
                                    DESTINATIONNBITS))
                         (PAINT)
                         (INVERT)
                         (ERASE)
                         (SHOULDNT]
             (T                                                ; going from color map into black and white map.
                (ERROR "not implemented to blt between bitmaps of different pixel size.")))
          (RETURN T])

)

(DEFINEQ

(BITMAPOBJ.SNAPW
  [LAMBDA NIL                                                  ; Edited 12-Apr-90 09:09 by matsuda

          (* * makes an image object of a prompted for region of the screen.)

    (PROG ((REG (GETREGION))
           BM)
          [SETQ BM (BITMAPCREATE (fetch (REGION WIDTH) of REG)
                                 (fetch (REGION HEIGHT) of REG)
                                 (BITSPERPIXEL (SCREENBITMAP \CURSORSCREEN]
          (BITBLT (SCREENBITMAP \CURSORSCREEN)
                  (fetch (REGION LEFT) of REG)
                  (fetch (REGION BOTTOM) of REG)
                  BM 0 0 NIL NIL 'INPUT 'REPLACE)
          (COPYINSERT (BITMAPTEDITOBJ BM 1 0))
          (RETURN])

)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: EVAL@COMPILE

[PROGN (DEFMACRO \MAIKO.CGTHREEP ()
```

```
          (EQ (LOGAND 120 (fetch DEVCONFIG of \InterfacePage))
              48))
       (PUTPROPS \MAIKO.CGTHREEP MACRO (NIL (EQ (LOGAND 120 (fetch DEVCONFIG of \InterfacePage))
                                                48)))]

(PUTPROPS \MAIKO.CGFOURP MACRO (NIL (EQ (LOGAND 120 (fetch DEVCONFIG of \InterfacePage))
                                        64)))

[PROGN (DEFMACRO \MAIKO.CGSIXP ()
          (EQ (LOGAND 120 (fetch DEVCONFIG of \InterfacePage))
              96))
       (PUTPROPS \MAIKO.CGSIXP MACRO (NIL (EQ (LOGAND 120 (fetch DEVCONFIG of \InterfacePage))
                                              96)))]

(PUTPROPS \MAIKO.CGTWOP MACRO (NIL (EQ (LOGAND 120 (fetch DEVCONFIG of \InterfacePage))
                                       24)))
)

(DECLARE%: EVAL@COMPILE

(RPAQQ \TO.MAIKO.MONOSCREEN 0)

(RPAQQ \TO.MAIKO.COLORSCREEN 1)

(RPAQQ \MAIKO.COLORSCREENWIDTH 1152)

(RPAQQ \MAIKO.COLORSCREENHEIGHT 900)

(RPAQQ \MAIKO.COLORPAGES 2048)

(RPAQQ \MAIKO.COLORBUF.ALIGN 4095)

(CONSTANTS (\TO.MAIKO.MONOSCREEN 0)
       (\TO.MAIKO.COLORSCREEN 1)
       (\MAIKO.COLORSCREENWIDTH 1152)
       (\MAIKO.COLORSCREENHEIGHT 900)
       (\MAIKO.COLORPAGES 2048)
       (\MAIKO.COLORBUF.ALIGN 4095))
)

(FILESLOAD (LOADCOMP)
       LLDISPLAY BIGBITMAPS)
)

(RPAQ? \MONO.PROMPTWINDOW NIL)

(RPAQ? \COLOR.PROMPTWINDOW NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS MAIKOCOLOR.BITSPERPIXEL)
)

(FILESLOAD COLOR BIGBITMAPS)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(MOVD 'CURSOREXIT 'SAVE.CURSOREXIT)

(MOVD '\MAIKO.BLTCHAR '\BILTCHAR)

(\MAIKO.COLORINIT)

(COLORDISPLAY 'ON 'MAIKOCOLOR)

(CURSORSCREEN (COLORSCREEN)
       100 100)

(CHANGEBACKGROUND 36)

(ADD-EXEC :TTY T :REGION '(0 650 370 150))

(LOGOW)
)

(PUTPROPS MAIKOCOLOR COPYRIGHT ("Fuji Xerox Co., Ltd." 1988 1989 1990 1991))
```

## FUNCTION INDEX



## CONSTANT INDEX



## VARIABLE INDEX



## MACRO INDEX