```
  File created:    31-Jul-2023 14:50:58   {WMEDLEY}<sources>LLDISPLAY.;19

        edit by:   rmk

     changes to:   (FNS BITMAPEQUAL)

  previous date:   31-Jul-2023 14:45:32   {WMEDLEY}<sources>LLDISPLAY.;18

    Read Table:    INTERLISP

       Package:    INTERLISP

        Format:     XCCS


(RPAQQ LLDISPLAYCOMS
       [(DECLARE%: DONTCOPY (EXPORT (RECORDS PILOTBBT \DISPLAYDATA DISPLAYSTATE DISPLAYINFO)
                                   (MACROS \GETDISPLAYDATA)))
                                                                 ; User-visible records are on ADISPLAY --- must be init'ed here
        (INITRECORDS BITMAP PILOTBBT REGION \DISPLAYDATA)
        [COMS                                                    ; BITMASKS
              (FNS \FBITMAPBIT \FBITMAPBIT.UFN \NEWPAGE.DISPLAY INITBITMASKS)
              (OPTIMIZERS \FBITMAPBIT)
              [EXPORT (DECLARE%: DONTCOPY (MACROS \BITMASK \4BITMASK \NOTBITMASK \NOT4BITMASK)
                             (GLOBALVARS BITMASKARRAY NOTBITMASKARRAY 4BITMASKARRAY NOT4BITMASKARRAY)
                             (CONSTANTS (WORDMASK 65535]
              (DECLARE%: DONTEVAL@LOAD DOCOPY (P (INITBITMASKS]
        [COMS                                                    ; init cursor
              (FNS \CreateCursorBitMap)
              (DECLARE%: DONTEVAL@LOAD DOCOPY (VARS (CursorBitMap (\CreateCursorBitMap]
        [COMS                                                    ; bitmap functions.
              (FNS BITBLT BLTSHADE \BITBLTSUB \GETPILOTBBTSCRATCHBM BITMAPCOPY BITMAPCREATE BITMAPBIT
                   BITMAPEQUAL BLTCHAR \BLTCHAR \MEDW.BLTCHAR \CHANGECHARSET.DISPLAY \INDICATESTRING
                   \SLOWBLTCHAR TEXTUREP INVERT.TEXTURE INVERT.TEXTURE.BITMAP BITMAPWIDTH READBITMAP
                   \INSUREBITSPERPIXEL MAXIMUMCOLOR OPPOSITECOLOR MAXIMUMSHADE OPPOSITESHADE \MEDW.BITBLT)
              (FUNCTIONS FINISH-READING-BITMAP)
              (CONSTANTS (MINIMUMCOLOR 0)
                     (MINIMUMSHADE 0))
              (P (MOVD 'BITMAPBIT '\BITMAPBIT))
              (DECLARE%: DONTCOPY (EXPORT (MACROS \INVALIDATEDISPLAYCACHE)))
              (OPTIMIZERS BITMAPBIT BITMAPP)
              (FNS BITMAPBIT.EXPANDER)
              (FNS \BITBLT.DISPLAY \BITBLT.BITMAP \BITBLT.MERGE \BLTSHADE.DISPLAY \BLTSHADE.BITMAP)
              (FNS ;; For SunLoadup

                   \BITBLT.BITMAP.SLOW)
              (FNS ;;  punt case for C funcs.bitblt_bitmap,bitshade.bitmap

                   \PUNT.BLTSHADE.BITMAP \PUNT.BITBLT.BITMAP)
              (FNS ;; from SUMEX-AIM

                   \SCALEDBITBLT.DISPLAY \BACKCOLOR.DISPLAY)
              (DECLARE%: DONTCOPY (CONSTANTS (\DisplayWordAlign 16)
                                        (\MaxBitMapWidth 65535)
                                        (\MaxBitMapHeight 65535)
                                        (\MaxBitMapWords 131066))
                     (EXPORT (MACROS \DSPGETCHARWIDTH \DSPGETCHARIMAGEWIDTH \DSPGETCHAROFFSET \CONVERTOP
                                    \SFInvert \SFReplicate \SETPBTFUNCTION \BITBLT1))
                     (GLOBALVARS \SYSBBTEXTURE \BBSCRATCHTEXTURE \SYSPILOTBBT \PILOTBBTSCRATCHBM))
              (VARS (\BBSCRATCHTEXTURE)
                    (\PILOTBBTSCRATCHBM))
              [DECLARE%: DONTEVAL@LOAD DOCOPY (P (MOVD? 'BITBLT 'BKBITBLT]
                                                                 ; macro for this file so that BITBLT can be broken by users
              (EXPORT (DECLARE%: DONTCOPY DONTEVAL@LOAD DOEVAL@COMPILE (P (PUTPROP 'BITBLT 'MACRO
                                                                                   '(= . BKBITBLT]
        (COMS                                                    ; display stream functions
              (FNS DISPLAYSTREAMP DSPSOURCETYPE DSPXOFFSET DSPYOFFSET)
              (FNS DSPDESTINATION DSPTEXTURE \DISPLAYSTREAMINCRXPOSITION \SFFixDestination \SFFixClippingRegion
                   \SFFixFont \SFFIXLINELENGTH \UPDATE-SYNONYM-STREAM-LINELENGTH-FIELD \SFFixY)
              [COMS (FNS \SIMPLE.DSPCREATE \COMMON.DSPCREATE)
                                                                 ; MOVD? so we don't trash a later redefinition
                    (P (MOVD? '\SIMPLE.DSPCREATE 'DSPCREATE]
              (FNS \MEDW.XOFFSET \MEDW.YOFFSET)
              (FNS \DSPCLIPPINGREGION.DISPLAY \DSPFONT.DISPLAY \DISPLAY.PILOTBITBLT \DSPLINEFEED.DISPLAY
                   \DSPLEFTMARGIN.DISPLAY \DSPOPERATION.DISPLAY \DSPRIGHTMARGIN.DISPLAY \DSPXPOSITION.DISPLAY
                   \DSPYPOSITION.DISPLAY)
              (P (MOVD? '\ILLEGAL.ARG '\COERCETODS)
                 (MOVD? 'NILL 'WFROMDS)
                 (MOVD? 'NILL 'WINDOWP)
                 (MOVD? 'NILL 'INVERTW))
              (INITVARS (PROMPTWINDOW T)
                     (\WINDOWWORLD NIL)
                     (\MAINSCREEN NIL)))
        [COMS                                                    ; Stub for window package
              (INITVARS (\TOPWDS)
                     (\SCREENBITMAPS))
              (P (MOVD? 'NILL '\TOTOPWDS))
              (DECLARE%: DONTCOPY EVAL@COMPILE (EXPORT (MACROS \INSURETOPWDS .WHILE.TOP.DS. .WHILE.CURSOR.DOWN.)
                                                      (ADDVARS (GLOBALVARS \TOPWDS]
        (COMS                                                      ; DisplayStream TTY functions
              (FNS TTYDISPLAYSTREAM)
              (EXPORT (OPTIMIZERS TTYDISPLAYSTREAM))
              (FNS DSPSCROLL PAGEHEIGHT)
              (INITVARS (\CURRENTTTYDEVICE 'BCPLDISPLAY))
              (FNS \DSPRESET.DISPLAY)
              (COMS (INITVARS (*DRIBBLE-OUTPUT* NIL))
                    (FUNCTIONS \MAYBE-DRIBBLE-CHAR)
                    (FNS \DSPPRINTCHAR \DSPPRINTCR/LF))
              (FNS \TTYBACKGROUND)
              (FNS DSPBACKUP)
              (INITVARS (\CARET.UP))
              (DECLARE%: DONTEVAL@LOAD DOCOPY (VARS (BELLCNT 2)
                                                    (BELLRATE 60)
                                                    (\DisplayStoppedForLogout)
                                                    (TtyDisplayStream)))
              (FNS COLORDISPLAYP)
              (FNS DISPLAYBEFOREEXIT DISPLAYAFTERENTRY)
              (EXPORT (GLOBALVARS BELLCNT BELLRATE TTYBACKGROUNDFNS \DisplayStoppedForLogout \CARET.UP)
                     (MACROS \CHECKCARET)))
        [COMS                                                      ; transformation related functions.
              (FNS \DSPCLIPTRANSFORMX \DSPCLIPTRANSFORMY \DSPTRANSFORMREGION \DSPUNTRANSFORMY \DSPUNTRANSFORMX
                   \OFFSETCLIPPINGREGION)
              (DECLARE%: DONTCOPY (EXPORT (MACROS \DSPTRANSFORMX \DSPTRANSFORMY \OFFSETBOTTOM \OFFSETLEFT]
        [COMS                                                      ; screen related functions
              (FNS UPDATESCREENDIMENSIONS \CreateScreenBitMap)
              (DECLARE%: DONTEVAL@LOAD DOCOPY (P (UPDATESCREENDIMENSIONS))
                     (INITVARS (SCREENHEIGHT 808)
                            (SCREENWIDTH 1024)
                            (\OLDSCREENHEIGHT 808)
                            (\OLDSCREENWIDTH 1024)
                            (\MaxScreenPage -1)
                            (ScreenBitMap (\CreateScreenBitMap SCREENWIDTH SCREENHEIGHT))
                            (ColorScreenBitMap NIL)))
              (GLOBALVARS \OLDSCREENHEIGHT \OLDSCREENWIDTH \MaxScreenPage ScreenBitMap)
              (DECLARE%: DONTEVAL@LOAD DOCOPY (P (CURSOR.INIT]
        [COMS                                                      ; initialization
              (INITVARS (\DISPLAYINFOALIST))
              (FNS \CoerceToDisplayDevice \CREATEDISPLAY DISPLAYSTREAMINIT \STARTDISPLAY
                   \MOVE.WINDOWS.ONTO.SCREEN \UPDATE.PBT.RASTERWIDTHS \STOPDISPLAY \DEFINEDISPLAYINFO)
              (DECLARE%: EVAL@COMPILE DONTCOPY (ADDVARS (DONTCOMPILEFNS \UPDATE.PBT.RASTERWIDTHS)))
              (EXPORT (MACROS DISPLAYINITIALIZEDP DISPLAYSTARTEDP)
                     (GLOBALVARS \DisplayStarted \DisplayStreamsInitialized \DisplayInitialed WHOLEDISPLAY
                            WHOLESCREEN SCREENWIDTH SCREENHEIGHT))
              (ADDVARS (GLOBALVARS WHOLESCREEN))
              (FNS INITIALIZEDISPLAYSTREAMS)
              (DECLARE%: DOCOPY DONTEVAL@LOAD (VARS (\DisplayStarted NIL)
                                                    (\LastTTYLines 12))
                     (P (INITIALIZEDISPLAYSTREAMS)
                        (DISPLAYSTREAMINIT 1000]
        (PROP FILETYPE LLDISPLAY)
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                               (NLAML)
                                                                               (LAMA])

(DECLARE%: DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

(DATATYPE PILOTBBT ((PBTDESTLO WORD)
                    (PBTDESTHI WORD)
                    (PBTDESTBIT WORD)
                    (PBTDESTBPL SIGNEDWORD)
                    (PBTSOURCELO WORD)
                    (PBTSOURCEHI WORD)
                    (PBTSOURCEBIT WORD)
                    (PBTSOURCEBPL SIGNEDWORD)
                    (PBTWIDTH WORD)
                    (PBTHEIGHT WORD)
                    (PBTFLAGS WORD)
                    (NIL 5 WORD))
       (BLOCKRECORD PILOTBBT ((NIL 7 WORD)
                              (NIL BITS 4)
                              (PBTGRAYOFFSET BITS 4)
                              (PBTGRAYWIDTHLESSONE BITS 4)
                              (PBTGRAYHEIGHTLESSONE BITS 4)
                              (NIL 2 WORD)
                              (PBTBACKWARD FLAG)
                              (PBTDISJOINT FLAG)
                              (PBTDISJOINTITEMS FLAG)
                              (PBTUSEGRAY FLAG)
                              (PBTSOURCETYPE BITS 1)
                              (PBTOPERATION BITS 2)
                              (NIL BITS 9)))
       [ACCESSFNS PILOTBBT ([PBTSOURCE (\VAG2 (fetch PBTSOURCEHI of DATUM)
                                             (fetch PBTSOURCELO of DATUM))
                                    (PROGN (replace PBTSOURCEHI of DATUM with (\HILOC NEWVALUE))
                                           (replace PBTSOURCELO of DATUM with (\LOLOC NEWVALUE]
                             (PBTDEST (\VAG2 (fetch PBTDESTHI of DATUM)
                                             (fetch PBTDESTLO of DATUM))
                                    (PROGN (replace PBTDESTHI of DATUM with (\HILOC NEWVALUE))
                                           (replace PBTDESTLO of DATUM with (\LOLOC NEWVALUE]
       (SYSTEM))

(DATATYPE \DISPLAYDATA (DDXPOSITION DDYPOSITION DDXOFFSET DDYOFFSET DDDestination DDClippingRegion DDFONT
                              DDSlowPrintingCase DDWIDTHSCACHE DDOFFSETSCACHE DDCOLOR DDLINEFEED DDRightMargin
                              DDLeftMargin DDScroll DDOPERATION DDSOURCETYPE (DDClippingLeft WORD)
                              (DDClippingRight WORD)
                              (DDClippingBottom WORD)
                              (DDClippingTop WORD)
                              (NIL WORD)
                              (DDHELDFLG FLAG)
                              (XWINDOWHINT XPOINTER)
                              (DDPILOTBBT POINTER)
                              DDXSCALE DDYSCALE DDCHARIMAGEWIDTHS DDEOLFN DDPAGEFULLFN DDTexture DDMICAXPOS
                              DDMICAYPOS DDMICARIGHTMARGIN DDCHARSET (DDCHARSETASCENT WORD)
                              (DDCHARSETDESCENT WORD)
                              DDCHARHEIGHTDELTA
                              (DDSPACEWIDTH WORD))
       DDPILOTBBT _ (create PILOTBBT
                           PBTDISJOINT _ T)
       DDLeftMargin _ 0 DDRightMargin _ SCREENWIDTH DDXPOSITION _ 0 DDYPOSITION _ 0 DDXOFFSET _ 0 DDYOFFSET _ 0
       DDClippingRegion _ (create REGION)
       DDDestination _ ScreenBitMap DDXSCALE _ 1 DDYSCALE _ 1 DDTexture _ 0
       [ACCESSFNS ([DDFOREGROUNDCOLOR (PROG ((VAL (fetch (\DISPLAYDATA DDCOLOR) of DATUM)))
                                            (OR (FIXP VAL)
                                                (BITMAPP VAL)
                                                (AND (NULL VAL)
                                                     1)
                                                (CAR VAL)
                                                (MAXIMUMCOLOR (BITSPERPIXEL (fetch (\DISPLAYDATA DDDestination)
                                                                                   of DATUM]
                   (DDBACKGROUNDCOLOR (OR (fetch (\DISPLAYDATA DDTexture) of DATUM)
                                          0]
       (SYSTEM))

(RECORD DISPLAYSTATE (ONOFF))

(RECORD DISPLAYINFO (DITYPE DIWIDTH DIHEIGHT DIBITSPERPIXEL DIWSOPS))
)

(/DECLAREDATATYPE 'PILOTBBT
       '(WORD WORD WORD SIGNEDWORD WORD WORD WORD SIGNEDWORD WORD WORD WORD WORD WORD WORD WORD WORD)

       ;; ---field descriptor list elided by lister---

       '16)

(/DECLAREDATATYPE '\DISPLAYDATA
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
               POINTER POINTER POINTER POINTER WORD WORD WORD WORD WORD FLAG XPOINTER POINTER POINTER POINTER
               POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER WORD WORD POINTER WORD)

       ;; ---field descriptor list elided by lister---

       '68)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \GETDISPLAYDATA MACRO [ARGS (COND
                                           [(CADR ARGS)
                                            (SUBPAIR '(STRM STRMVAR)
                                                   ARGS
                                                   '(\DTEST (fetch (STREAM IMAGEDATA) of (SETQ STRMVAR
                                                                                             (\OUTSTREAMARG STRM)))
                                                           '\DISPLAYDATA]
                                           (T (SUBST (CAR ARGS)
                                                     'STRM
                                                     '(\DTEST (fetch (STREAM IMAGEDATA) of (\OUTSTREAMARG STRM))
                                                             '\DISPLAYDATA])
)
)

;; END EXPORTED DEFINITIONS

;; User-visible records are on ADISPLAY --- must be init'ed here

(/DECLAREDATATYPE 'BITMAP '(POINTER WORD WORD WORD WORD)

       ;; ---field descriptor list elided by lister---

       '6)

(/DECLAREDATATYPE 'PILOTBBT
       '(WORD WORD WORD SIGNEDWORD WORD WORD WORD SIGNEDWORD WORD WORD WORD WORD WORD WORD WORD WORD)
```

```
        ;; ---field descriptor list elided by lister---
        '16)

(/DECLAREDATATYPE '\DISPLAYDATA
        '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
                POINTER POINTER POINTER POINTER WORD WORD WORD WORD WORD FLAG XPOINTER POINTER POINTER POINTER
                POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER WORD WORD POINTER WORD)
        ;; ---field descriptor list elided by lister---
        '68)

;; BITMASKS

(DEFINEQ

(\FBITMAPBIT
  [LAMBDA (BASE X Y OPERATION HEIGHTMINUS1 RASTERWIDTH)          ; Edited  6-Oct-89 14:59 by jds
    ;; fast version of stuffing a bit into a bitmap.
    (\FBITMAPBIT.UFN BASE X Y (SELECTQ OPERATION
                                   (INVERT 0)
                                   (ERASE 1)
                                   (READ 2)
                                   3)
           HEIGHTMINUS1 RASTERWIDTH])

(\FBITMAPBIT.UFN
  [LAMBDA (BASE X Y OPERATION HEIGHTMINUS1 RASTERWIDTH)          ; Edited  6-Oct-89 15:00 by jds
    ;; fast version of stuffing a bit into a bitmap.
    ;; UFN FOR MISC6 sub-op 0.
    (LET ([WORDBASE (\ADDBASE BASE (IPLUS (ITIMES (IDIFFERENCE HEIGHTMINUS1 Y)
                                                  RASTERWIDTH)
                                          (LRSH X 4]
          (BITMASK (\BITMASK X)))
         (PROG1 (COND
                   ((ZEROP (LOGAND BITMASK (fetch (BITMAPWORD BITS) of WORDBASE)))
                    0)
                   (T 1))
             (change (fetch (BITMAPWORD BITS) of WORDBASE)
                    (SELECTQ OPERATION
                        (0 (LOGXOR DATUM BITMASK))
                        (1 (LOGAND DATUM (\NOTBITMASK X)))
                        (2 ;; Just read the value out.
                           DATUM)
                        (LOGOR DATUM BITMASK))))])

(\NEWPAGE.DISPLAY
  [LAMBDA (STREAM)                                               (* hdj "10-Dec-84 12:31")
    (DSPRESET STREAM])

(INITBITMASKS
  [LAMBDA NIL                                                    (* rrb "24-SEP-82 15:13")
    ;; initialization of bit masks for line drawing routines.  BITMASK is an array of single bit masks;  NOTBITMASK is an array of masks for getting
    ;; everything except the nth bit.
    (SETQ BITMASKARRAY (ARRAY 16 'SMALLPOSP 0 0))
    (SETQ NOTBITMASKARRAY (ARRAY 16 'SMALLPOSP 0 0))
    (for I from 0 to 15 bind (MASK _ (CONSTANT (EXPT 2 15))) do (SETA BITMASKARRAY I MASK)
                                                                (SETA NOTBITMASKARRAY I (LOGXOR MASK WORDMASK))
                                                                (SETQ MASK (LRSH MASK 1)))
    (SETQ 4BITMASKARRAY (ARRAY 4 'SMALLPOSP 0 0))
    (SETQ NOT4BITMASKARRAY (ARRAY 4 'SMALLPOSP 0 0))
    (for I from 0 to 3 bind [MASK _ (CONSTANT (IDIFFERENCE (EXPT 2 16)
                                                          (EXPT 2 12]
       do (SETA 4BITMASKARRAY I MASK)
          (SETA NOT4BITMASKARRAY I (LOGXOR MASK WORDMASK))
          (SETQ MASK (LRSH MASK 4])
)

(DEFOPTIMIZER \FBITMAPBIT (BASE X Y OPERATION HEIGHTMINUS1 RASTERWIDTH)
                             `((OPCODES MISC7 1)
                               ,BASE
                               ,X
                               ,Y
                               ,[COND
                                   ([OR (AND (LISTP OPERATION)
                                             (EQ (CAR OPERATION)
                                                 'QUOTE]
                                   (SELECTQ (EVAL OPERATION)
                                        (INVERT 0)
                                        (ERASE 1)
                                        (READ 2)
                                        3))
                                  (T `(SELECTQ ,OPERATION
                                           (INVERT 0)
                                           (ERASE 1)
                                           (READ 2)
                                           3]
                              ,HEIGHTMINUS1
                              ,RASTERWIDTH NIL))
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS \BITMASK MACRO ((N)
                          (\WORDELT BITMASKARRAY (LOGAND N 15))))

(PUTPROPS \4BITMASK MACRO ((N)
                           (\WORDELT 4BITMASKARRAY (LOGAND N 3))))

(PUTPROPS \NOTBITMASK MACRO ((N)
                             (DECLARE (GLOBALVARS NOTBITMASKARRAY))
                             (\WORDELT NOTBITMASKARRAY (LOGAND N 15))))

(PUTPROPS \NOT4BITMASK MACRO ((N)
                              (\WORDELT NOT4BITMASKARRAY (LOGAND N 3))))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS BITMASKARRAY NOTBITMASKARRAY 4BITMASKARRAY NOT4BITMASKARRAY)
)

(DECLARE%: EVAL@COMPILE

(RPAQQ WORDMASK 65535)

(CONSTANTS (WORDMASK 65535))
)
)
```

;; END EXPORTED DEFINITIONS

```
(DECLARE%: DONTEVAL@LOAD DOCOPY

(INITBITMASKS)
)
```

;; init cursor

```
(DEFINEQ

(\CreateCursorBitMap
  [LAMBDA NIL                                                    (* lmm "13-MAY-82 00:24")
```

;; creates a BITMAP which points at the cursor bits.

;; pointer to cursor is stored using hiloc and loloc rather that BITMAPBASE so that it won't be reference counted. It is on an odd boundary.

```
    (create BITMAP
           BITMAPRASTERWIDTH _ 1
           BITMAPWIDTH _ 16
           BITMAPHEIGHT _ 16
           BITMAPBASE _ \EM.CURSORBITMAP])
)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(RPAQ CursorBitMap (\CreateCursorBitMap))
)
```

;; bitmap functions.

```
(DEFINEQ

(BITBLT
  [LAMBDA (SOURCE SOURCELEFT SOURCEBOTTOM DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT SOURCETYPE
                 OPERATION TEXTURE CLIPPINGREGION)                ; Edited 12-Jan-88 23:05 by FS
    (DECLARE (LOCALVARS . T))
```

;; IRM defined defaults

```
    (OR DESTINATIONLEFT (SETQ DESTINATIONLEFT 0))
    (OR DESTINATIONBOTTOM (SETQ DESTINATIONBOTTOM 0))
    (COND
       [(EQ SOURCETYPE 'TEXTURE)
        (COND
           ((type? BITMAP DESTINATION)
            (\BLTSHADE.BITMAP TEXTURE DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT OPERATION
                   CLIPPINGREGION))
           (T (PROG ((STREAM (\OUTSTREAMARG DESTINATION)))
                    (RETURN (IMAGEOP 'IMBLTSHADE STREAM TEXTURE STREAM DESTINATIONLEFT DESTINATIONBOTTOM WIDTH
                                   HEIGHT OPERATION CLIPPINGREGION]
       (T (PROG (SOURCEDD SOURCEBM CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM)
                [COND
                   [(type? BITMAP SOURCE)
                    (OR SOURCELEFT (SETQ SOURCELEFT 0))
                    (OR SOURCEBOTTOM (SETQ SOURCEBOTTOM 0))
                    (SETQ SOURCEBM SOURCE)
                    (SETQ CLIPPEDSOURCELEFT SOURCELEFT)
                    (SETQ CLIPPEDSOURCEBOTTOM SOURCEBOTTOM)       ; limit the WIDTH and HEIGHT to the source size.
                    [SETQ WIDTH (COND
                                   (WIDTH (IMIN WIDTH (IDIFFERENCE (fetch (BITMAP BITMAPWIDTH) of SOURCE)
                                                             SOURCELEFT)))
                                   (T (fetch (BITMAP BITMAPWIDTH) of SOURCE]
                    (SETQ HEIGHT (COND
                                    (HEIGHT (IMIN HEIGHT (IDIFFERENCE (fetch (BITMAP BITMAPHEIGHT) of SOURCE)
                                                                SOURCEBOTTOM)))
                                    (T (fetch (BITMAP BITMAPHEIGHT) of SOURCE]
                   ((SETQ SOURCEDD (\GETDISPLAYDATA SOURCE))
                    [OR SOURCELEFT (SETQ SOURCELEFT (fetch (REGION LEFT) of (ffetch (\DISPLAYDATA DDClippingRegion)
                                                                                 of SOURCEDD]
                    [OR SOURCEBOTTOM (SETQ SOURCEBOTTOM (fetch (REGION BOTTOM) of (ffetch (\DISPLAYDATA
                                                                                            DDClippingRegion)
                                                                                       of SOURCEDD]
                                                                  ; do transformations coming out of source
                    (SETQ SOURCEBM (fetch (\DISPLAYDATA DDDestination) of SOURCEDD))
                    (SETQ CLIPPEDSOURCELEFT (IMAX (SETQ SOURCELEFT (\DSPTRANSFORMX SOURCELEFT SOURCEDD))
                                                  (fetch (\DISPLAYDATA DDClippingLeft) of SOURCEDD)))
                    (SETQ CLIPPEDSOURCEBOTTOM (IMAX (SETQ SOURCEBOTTOM (\DSPTRANSFORMY SOURCEBOTTOM SOURCEDD))
                                                    (fetch (\DISPLAYDATA DDClippingBottom) of SOURCEDD)))
                                                                  ; limit the WIDTH and HEIGHT by the source dimensions.
                    [SETQ WIDTH (COND
                                   (WIDTH (IMIN WIDTH (IDIFFERENCE (fetch (\DISPLAYDATA DDClippingRight)
                                                                      of SOURCEDD)
                                                             CLIPPEDSOURCELEFT)))
                                   (T (IDIFFERENCE (fetch (\DISPLAYDATA DDClippingRight) of SOURCEDD)
                                             CLIPPEDSOURCELEFT]
                    [SETQ HEIGHT (COND
                                    (HEIGHT (IMIN HEIGHT (IDIFFERENCE (fetch (\DISPLAYDATA DDClippingTop)
                                                                         of SOURCEDD)
                                                                CLIPPEDSOURCEBOTTOM)))
                                    (T (IDIFFERENCE (fetch (\DISPLAYDATA DDClippingTop) of SOURCEDD)
                                              CLIPPEDSOURCEBOTTOM]
                                                                  ; if texture is not given, use the display stream's.
                    (OR TEXTURE (SETQ TEXTURE (ffetch (\DISPLAYDATA DDTexture) of SOURCEDD]
                (COND
                   ((OR (IGEQ 0 WIDTH)
                        (IGEQ 0 HEIGHT))                          ; if either width or height is 0, don't do anything.
                    (RETURN)))
                (RETURN (COND
                           [(type? BITMAP DESTINATION)
                            (COND
                               ((WINDOWP SOURCE)

                                ;; bring source window to the top.  Note: this doesn't work if the user passes in a display stream onto the
                                ;; screen instead of a window.

                                (.WHILE.TOP.DS. (\OUTSTREAMARG SOURCE)
                                       (\BITBLT.BITMAP SOURCEBM SOURCELEFT SOURCEBOTTOM DESTINATION
                                              DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT SOURCETYPE
                                              OPERATION TEXTURE CLIPPINGREGION CLIPPEDSOURCELEFT
                                              CLIPPEDSOURCEBOTTOM)))
                               (T (\BITBLT.BITMAP SOURCEBM SOURCELEFT SOURCEBOTTOM DESTINATION DESTINATIONLEFT
                                         DESTINATIONBOTTOM WIDTH HEIGHT SOURCETYPE OPERATION TEXTURE
                                         CLIPPINGREGION CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM]
                           (T (PROG (STREAM)
                                    (SETQ STREAM (\OUTSTREAMARG DESTINATION))
                                    (COND
                                       ((AND (NEQ SOURCE DESTINATION)
                                             (WINDOWP SOURCE))

                                        ;; both source and destination are windows, see if they overlap and use an intermediate bitmap.
                                        ;; Note: this doesn't work if the user passes in a display stream onto the screen instead of a window.

                                        [COND
                                           ((WINDOWP DESTINATION)
                                            (COND
                                               ((WOVERLAPP SOURCE DESTINATION)
                                                (RETURN (PROG (SCRATCHBM)
                                                              (.WHILE.TOP.DS. (\OUTSTREAMARG SOURCE)
                                                                     (BITBLT SOURCEBM SOURCELEFT SOURCEBOTTOM
                                                                            (SETQ SCRATCHBM (BITMAPCREATE WIDTH
                                                                                                   HEIGHT))
                                                                            0 0 WIDTH HEIGHT 'INPUT 'REPLACE))
                                                              (RETURN (BITBLT SCRATCHBM 0 0 STREAM
                                                                             DESTINATIONLEFT DESTINATIONBOTTOM
                                                                             WIDTH HEIGHT SOURCETYPE OPERATION
                                                                             TEXTURE CLIPPINGREGION]
                                                                ; bring the source to the top.  this should be done uninterruptably
                                                                ; but is better than nothing.
                                        (TOTOPW SOURCE)))
                                    (IMAGEOP 'IMBITBLT STREAM SOURCEBM SOURCELEFT SOURCEBOTTOM STREAM
                                           DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT SOURCETYPE OPERATION
                                           TEXTURE CLIPPINGREGION CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM])
```

(**BLTSHADE**

```
  [LAMBDA (TEXTURE DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT OPERATION CLIPPINGREGION)
                                                                     (* rrb " 7-Mar-86 11:26")
    (DECLARE (LOCALVARS . T))
    (COND
       ((type? BITMAP DESTINATION)
        (\BLTSHADE.BITMAP TEXTURE DESTINATION (OR DESTINATIONLEFT 0)
               (OR DESTINATIONBOTTOM 0)
               WIDTH HEIGHT OPERATION CLIPPINGREGION))
       (T (PROG ((STREAM (\OUTSTREAMARG DESTINATION)))
                (RETURN (IMAGEOP 'IMBLTSHADE STREAM TEXTURE STREAM (OR DESTINATIONLEFT 0)
                               (OR DESTINATIONBOTTOM 0)
                               WIDTH HEIGHT (OR OPERATION (DSPOPERATION NIL STREAM))
                               CLIPPINGREGION])
```

(\BITBLTSUB

```
  [LAMBDA (PILOTBBT SourceBitMap SLX STY DestinationBitMap DLX DTY HEIGHT SourceType Operation Texture
                 WindowXOffset WindowYOffset)                        (* rrb "13-Feb-86 14:42")

    ;; rrb 13-Feb-86 Added WindowYOffset and WindowXOffset so that textures could be aligned to the window rather than the underlying Screen
    ;; bitmap.  I only changed the calls in \BLTSHADE.1BITDISPLAY and \BLTSHADE.COLORDISPLAY

    (PROG (DBMR SBMR GRAY SOURCEADDR DESTADDR X)
          (SETQ DBMR (fetch (BITMAP BITMAPRASTERWIDTH) of DestinationBitMap))
          (replace (PILOTBBT PBTFLAGS) of PILOTBBT with 0)
          (replace (PILOTBBT PBTDESTBPL) of PILOTBBT with (UNFOLD DBMR BITSPERWORD))
          (SETQ DESTADDR (\ADDBASE (fetch (BITMAP BITMAPBASE) of DestinationBitMap)
                                (ITIMES DBMR DTY)))                  ; Combine Destination base and top Y into a single Destination
                                                                     ; word offset
          (replace (PILOTBBT PBTDESTBIT) of PILOTBBT with DLX)
          (SELECTQ SourceType
               (TEXTURE (replace (PILOTBBT PBTUSEGRAY) of PILOTBBT with T)
                        (replace (PILOTBBT PBTSOURCEBIT) of PILOTBBT with (MOD (COND
                                                                                   (WindowXOffset (IDIFFERENCE DLX
                                                                                                         WindowXOffset
                                                                                                    ))
                                                                                   (T DLX))
                                                                               BITSPERWORD))

                        ;; Source is offset in a gray block where we want to start.  Microcode finds the start of the gray block by subtracting
                        ;; PBTGRAYOFFSET from it

                        (replace (PILOTBBT PBTSOURCEBPL) of PILOTBBT with 0)
                                                                     ; Zero out this word first
                        [COND
                           [(FIXP Texture)
                            (SETQ GRAY (fetch (BITMAP BITMAPBASE) of \SYSBBTEXTURE))
                            (replace (PILOTBBT PBTSOURCE) of PILOTBBT
                               with (\ADDBASE GRAY (COND
                                                      ((OR (EQ (SETQ Texture (LOGAND Texture WORDMASK))
                                                               0)
                                                           (EQ Texture BLACKSHADE))
                                                                     ; special cases of solid texture occur often
                                                       (\PUTBASE GRAY 0 Texture)
                                                                     ; PBTGRAYHEIGHTLESSONE and PBTGRAYOFFSET are both
                                                                     ; 0 in this case
                                                       0)
                                                      (T (\PUTBASE GRAY 0 (\SFReplicate (LRSH Texture 12)))
                                                         [\PUTBASE GRAY 1 (\SFReplicate (LOGAND 15
                                                                                                (LRSH Texture 8]
                                                         [\PUTBASE GRAY 2 (\SFReplicate (LOGAND 15
                                                                                                (LRSH Texture 4]
                                                         (\PUTBASE GRAY 3 (\SFReplicate (LOGAND 15 Texture)))
                                                         (replace (PILOTBBT PBTGRAYHEIGHTLESSONE) of PILOTBBT
                                                            with 3)
                                                         (replace (PILOTBBT PBTGRAYOFFSET) of PILOTBBT
                                                            with (MOD (COND
                                                                         (WindowYOffset (PLUS DTY WindowYOffset))
                                                                         (T DTY))
                                                                      4]
                           (T                                        ; A bitmap that is 16 bits wide.  BITBLT verified this back in
                                                                     ; interruptable section
```

```
                              [replace (PILOTBBT PBTGRAYHEIGHTLESSONE) of PILOTBBT
                                 with (SUB1 (SETQ X (IMIN [ffetch (BITMAP BITMAPHEIGHT)
                                                             of (SETQ Texture (\DTEST Texture 'BITMAP]
                                                           16]
                              [replace (PILOTBBT PBTGRAYOFFSET) of PILOTBBT
                                 with (SETQ X (COND
                                                 (WindowYOffset (MOD (PLUS DTY WindowYOffset)
                                                                     X))
                                                 (T (IREMAINDER DTY X]
                              (replace (PILOTBBT PBTSOURCE) of PILOTBBT with (\ADDBASE (ffetch (BITMAP BITMAPBASE)
                                                                                           of Texture)
                                                                                X])
                 (MERGE (RETURN (RAID "Hard bitblt case")))
                 (PROGN                                                    ; INPUT or INVERT
                        (replace (PILOTBBT PBTUSEGRAY) of PILOTBBT with NIL)
                        (replace (PILOTBBT PBTSOURCEBPL) of PILOTBBT with (UNFOLD (SETQ SBMR (fetch (BITMAP
                                                                                                     BITMAPRASTERWIDTH
                                                                                                          )
                                                                                                  of SourceBitMap))
                                                                                 BITSPERWORD))
                        (SETQ SOURCEADDR (\ADDBASE (fetch (BITMAP BITMAPBASE) of SourceBitMap)
                                                (ITIMES SBMR STY)))
                                                                           ; Combine Source base and top Y into a single Source word
                                                                           ; offset
                        (replace (PILOTBBT PBTSOURCEBIT) of PILOTBBT with SLX)
                        [COND
                           ((NOT (EQ SourceBitMap DestinationBitMap))
                                                                           ; Assume distinct bitmaps do not overlap, i.e. that we do not
                                                                           ; have sub-bitmaps
                            (replace (PILOTBBT PBTDISJOINT) of PILOTBBT with T))
                           [(IGREATERP STY DTY)                            ; Source > Dest means we can go top to bottom always
                            (COND
                               ((IGREATERP STY (IPLUS DTY HEIGHT))         ; Dest ends before source starts, so is completely disjoint
                                (replace (PILOTBBT PBTDISJOINT) of PILOTBBT with T))
                               (T                                          ; Not disjoint, but the items are disjoint
                                  (replace (PILOTBBT PBTDISJOINTITEMS) of PILOTBBT with T]
                           ((IGREATERP DTY (IPLUS STY HEIGHT))             ; Source ends before dest starts, so is completely disjoint
                            (replace (PILOTBBT PBTDISJOINT) of PILOTBBT with T))
                           ([OR (NOT (EQ STY DTY))
                                (AND (ILESSP SLX DLX)
                                     (ILESSP DLX (IPLUS SLX (fetch (PILOTBBT PBTWIDTH) of PILOTBBT]

                            ;; Not disjoint, with source above dest (bottom to top) or source and dest the same line with source to left of dest (right
                            ;; to left)

                            (replace (PILOTBBT PBTBACKWARD) of PILOTBBT with T)
                                                                           ; What's more, the source and dest addresses are to be of the
                                                                           ; LAST item, and bpl is negative
                                                                           ; note SBMR = DBMR if we have gotten this far
                            [SETQ SOURCEADDR (\ADDBASE SOURCEADDR (SETQ X (ITIMES SBMR (SUB1 HEIGHT]
                            (SETQ DESTADDR (\ADDBASE DESTADDR X))
                            [replace (PILOTBBT PBTSOURCEBPL) of PILOTBBT with (SETQ X (IMINUS (UNFOLD SBMR
                                                                                                      BITSPERWORD]
                            (replace (PILOTBBT PBTDESTBPL) of PILOTBBT with X)
                            (COND
                               ((NOT (EQ STY DTY))                         ; At least the items are disjoint
                                (replace (PILOTBBT PBTDISJOINTITEMS) of PILOTBBT with T]
                        (replace (PILOTBBT PBTSOURCE) of PILOTBBT with SOURCEADDR)))
            (replace (PILOTBBT PBTDEST) of PILOTBBT with DESTADDR)
            (\SETPBTFUNCTION PILOTBBT SourceType Operation)
            (RETURN (\PILOTBITBLT PILOTBBT 0])
```

(\GETPILOTBBTSCRATCHBM

```
  [LAMBDA (WIDTH HEIGHT)
    (DECLARE (GLOBALVARS \PILOTBBTSCRATCHBM))                   (* bvm%: "24-MAY-82 12:46")

    ;; Return a scratch bitmap at least WIDTH by HEIGHT.  Called only under uninterruptable bitblt, so don't worry about global resource conflicts

    (COND
       ((AND (type? BITMAP \PILOTBBTSCRATCHBM)
             (ILEQ WIDTH (fetch BITMAPWIDTH of \PILOTBBTSCRATCHBM))
             (ILEQ HEIGHT (fetch BITMAPHEIGHT of \PILOTBBTSCRATCHBM)))
        \PILOTBBTSCRATCHBM)
       (T (SETQ \PILOTBBTSCRATCHBM (BITMAPCREATE WIDTH HEIGHT])
```

(BITMAPCOPY

```
  [LAMBDA (BITMAP)                                                 (* rrb "22-DEC-82 11:09")

    ;; makes a copy of an existing BitMap

    (PROG (NEWBITMAP)
          (BITBLT (SETQ BITMAP (\DTEST BITMAP 'BITMAP))
                 0 0 (SETQ NEWBITMAP (BITMAPCREATE (BITMAPWIDTH BITMAP)
                                            (ffetch BITMAPHEIGHT of BITMAP)
                                            (ffetch BITMAPBITSPERPIXEL of BITMAP)))
                 0 0 NIL NIL 'INPUT 'REPLACE 0)
          (RETURN NEWBITMAP])
```

```
(BITMAPCREATE
  [LAMBDA (WIDTH HEIGHT BITSPERPIXEL)                              (* kbr%: " 2-Sep-85 18:50")
                                                                   ; creates a bitmap data structure.
    (PROG (RW)
          (OR (AND (IGEQ WIDTH 0)
                   (ILEQ WIDTH \MaxBitMapWidth))
              (\ILLEGAL.ARG WIDTH))
          (OR (AND (IGEQ HEIGHT 0)
                   (ILEQ HEIGHT \MaxBitMapHeight))
              (\ILLEGAL.ARG HEIGHT))
          (SETQ BITSPERPIXEL (\INSUREBITSPERPIXEL BITSPERPIXEL))
          (SETQ RW (FOLDHI (ITIMES WIDTH BITSPERPIXEL)
                          BITSPERWORD))
          (RETURN (create BITMAP
                         BITMAPRASTERWIDTH _ RW
                         BITMAPWIDTH _ WIDTH
                         BITMAPHEIGHT _ HEIGHT
                         BITMAPBITSPERPIXEL _ BITSPERPIXEL
                         BITMAPBASE _ (COND
                                         ((IGREATERP (SETQ RW (ITIMES RW HEIGHT))
                                                 \MaxBitMapWords)
                                          (ERROR (ITIMES WIDTH HEIGHT BITSPERPIXEL)
                                                 "bits in BITMAP -- too big"))
                                         (T (\ALLOCBLOCK (FOLDHI RW WORDSPERCELL)
                                                   NIL
                                                   (AND (NULL WINDFLG)
                                                        0])

(BITMAPBIT
  [LAMBDA (BITMAP X Y NEWVALUE)                                    ; Edited 10-Oct-89 11:17 by jds

;;; reads and optionally sets a bit in a bitmap.  If bitmap is a displaystream, it works on the destination through the coordinate transformations.

    ;; version of BITMAPBIT that works for multiple bit per pixel bitmaps.

    (PROG (NBITS BITX WORDX OLDVALUE HEIGHT oldword bitmapbase)
          (RETURN (COND
                     [(type? BITMAP BITMAP)
                      (SETQ NBITS (fetch (BITMAP BITMAPBITSPERPIXEL) of BITMAP))
                      (COND
                         ([OR (IGREATERP 0 X)
                              (IGEQ X (fetch (BITMAP BITMAPWIDTH) of BITMAP))
                              (IGREATERP 0 Y)
                              (IGEQ Y (SETQ HEIGHT (fetch (BITMAP BITMAPHEIGHT) of BITMAP]
                                                                   ; all bitmaps are 0 outside
                          0)
                         [(EQ NBITS 1)

                          ;; Special case for single-bit bitmaps, i.e., the display.

                          (COND
                             ((EQ NEWVALUE 0)
                              (\FBITMAPBIT (fetch (BITMAP BITMAPBASE) of BITMAP)
                                     X Y 'ERASE (SUB1 HEIGHT)
                                     (fetch (BITMAP BITMAPRASTERWIDTH) of BITMAP)))
                             ((NOT NEWVALUE)
                              (\FBITMAPBIT (fetch (BITMAP BITMAPBASE) of BITMAP)
                                     X Y 'READ (SUB1 HEIGHT)
                                     (fetch (BITMAP BITMAPRASTERWIDTH) of BITMAP)))
                             (T (\FBITMAPBIT (fetch (BITMAP BITMAPBASE) of BITMAP)
                                       X Y 'PAINT (SUB1 HEIGHT)
                                       (fetch (BITMAP BITMAPRASTERWIDTH) of BITMAP]
                         (T [SETQ bitmapbase (\ADDBASE (fetch (BITMAP BITMAPBASE) of BITMAP)
                                                    (ITIMES (SUB1 (\SFInvert BITMAP Y))
                                                           (fetch (BITMAP BITMAPRASTERWIDTH) of BITMAP]
                            [COND
                               (NEWVALUE                           ; check NEWVALUE before going uninterruptable.
                                      (COND
                                         ([NOT (AND (IGEQ NEWVALUE MINIMUMCOLOR)
                                                    (ILEQ NEWVALUE (MAXIMUMCOLOR (fetch (BITMAP
                                                                                         BITMAPBITSPERPIXEL
                                                                                          )
                                                                                  of BITMAP]
                                          (\ILLEGAL.ARG NEWVALUE]
                            (SELECTQ NBITS
                                 (1 (SETQ WORDX (FOLDLO X BITSPERWORD))

                                    ;;

                                    (SETQ oldword (\GETBASE bitmapbase WORDX))
                                    (SETQ BITX (\BITMASK X))
                                    [if NEWVALUE
                                        then (if (EQ NEWVALUE 0)
                                                 then (\PUTBASE bitmapbase WORDX (LOGAND oldword (LOGXOR BITX -1)
                                                                                         ))
                                               else (\PUTBASE bitmapbase WORDX (LOGOR oldword BITX]
                                    (if (EQ 0 (LOGAND oldword BITX))
                                        then 0
```

```
                                                else 1))
                                           (4 (SETQ BITX (LSH X 2))
                                              (SETQ WORDX (FOLDLO BITX BITSPERWORD))
                                              (SETQ oldword (\GETBASE bitmapbase WORDX))
                                              (SETQ OLDVALUE (LOGAND oldword (\4BITMASK X)))
                                              [COND
                                                 (NEWVALUE (\PUTBASE bitmapbase WORDX
                                                                  (LOGOR (LOGXOR oldword OLDVALUE)
                                                                         (LLSH NEWVALUE (ITIMES 4 (IDIFFERENCE
                                                                                                   3
                                                                                                   (LOGAND X 3]
                                                                           ; move the 4 bit current value to the right most bits.
                                              [LRSH OLDVALUE (ITIMES 4 (IDIFFERENCE 3 (LOGAND X 3])
                                           (8 (SETQ BITX (LSH X 3))
                                              (SETQ WORDX (FOLDLO BITX BITSPERWORD))
                                              [COND
                                                 ((EQ (LOGAND X 1)
                                                      0)                   ; left half of word
                                                  (SETQ oldword (\GETBASE bitmapbase WORDX))
                                                  (SETQ OLDVALUE (LOGAND oldword 65280))
                                                  [COND
                                                     (NEWVALUE (\PUTBASE bitmapbase WORDX (LOGOR (LOGXOR oldword OLDVALUE)
                                                                                                 (LLSH NEWVALUE 8]
                                                  (SETQ OLDVALUE (LRSH OLDVALUE 8)))
                                                 (T                        ; right half of word
                                                    (SETQ oldword (\GETBASE bitmapbase WORDX))
                                                    (SETQ OLDVALUE (LOGAND oldword 255))
                                                    (COND
                                                       (NEWVALUE (\PUTBASE bitmapbase WORDX (LOGOR (LOGXOR oldword
                                                                                                           OLDVALUE)
                                                                                                   NEWVALUE]
                                              OLDVALUE)
                                           (24 (SETQ BITX (ITIMES X 24))
                                               (SETQ WORDX (FOLDLO BITX BITSPERWORD))
                                               (SETQ OLDVALUE (\GETBASE24 bitmapbase X))
                                               (COND
                                                  (NEWVALUE (\PUTBASE24 bitmapbase X NEWVALUE)))
                                               OLDVALUE)
                                           (ERROR "unknown bits per pixel size." NBITS]
                            (T (PROG (TX TY DD)
                                     (SETQ DD (\GETDISPLAYDATA BITMAP BITMAP))
                                     (SETQ TX (\DSPCLIPTRANSFORMX X DD))
                                     (SETQ TY (\DSPCLIPTRANSFORMY Y DD))
                                     (RETURN (COND
                                                ((AND TX TY)
                                                 (.WHILE.TOP.DS. BITMAP (SETQ TX (BITMAPBIT (fetch (\DISPLAYDATA
                                                                                                     DDDestination)
                                                                                              of DD)
                                                                                       TX TY NEWVALUE)))
                                                 TX)
                                                (T                         ; anything outside the clipping region returns 0.
                                                   0])
```

(BITMAPEQUAL

```
  [LAMBDA (BM1 BM2)                                                ; Edited 31-Jul-2023 14:50 by rmk

    ;; T if BM1 and BM2 are both bitmaps of the same shape and contents.  The numeric fields are all SMALLP's

    (if (AND (type? BITMAP BM1)
             (type? BITMAP BM2))
        then (CL:WHEN (AND (EQ (ffetch (BITMAP BITMAPWIDTH) of BM1)
                               (ffetch (BITMAP BITMAPWIDTH) of BM2))
                           (EQ (ffetch (BITMAP BITMAPHEIGHT) of BM1)
                               (ffetch (BITMAP BITMAPHEIGHT) of BM2))
                           (EQ (ffetch (BITMAP BITMAPRASTERWIDTH) of BM1)
                               (ffetch (BITMAP BITMAPRASTERWIDTH) of BM2))
                           (EQ (ffetch (BITMAP BITMAPBITSPERPIXEL) of BM1)
                               (ffetch (BITMAP BITMAPBITSPERPIXEL) of BM2)))
                 (for I (BASE1 _ (ffetch (BITMAP BITMAPBASE) of BM1))
                        (BASE2 _ (ffetch (BITMAP BITMAPBASE) of BM2)) from 0
                    to (SUB1 (ITIMES (ffetch (BITMAP BITMAPRASTERWIDTH) of BM1)
                                     (ffetch (BITMAP BITMAPHEIGHT) of BM1)))
                    always (EQ (\GETBASE BASE1 I)
                               (\GETBASE BASE2 I))))
      else (BIGBITMAPEQUAL BM1 BM2])
```

(BLTCHAR

```
  [LAMBDA (CHARCODE DISPLAYSTREAM)                                 (* rmk%: " 4-Apr-85 11:45")
                                                                   ; user entry --- seldom used

    ;; puts a character on a display stream.  Much of the information needed by the BitBlt microcode is prestored by the routines that change it.  This is
    ;; kept in the BitBltTable.

    (\BLTCHAR (COND
                 ((\CHARCODEP CHARCODE)
                  CHARCODE)
                 (T (\ILLEGAL.ARG CHARCODE)))
```

```
          DISPLAYSTREAM
          (\GETDISPLAYDATA DISPLAYSTREAM])
```

(**\BLTCHAR**

```
  [LAMBDA (CHARCODE DISPLAYSTREAM DISPLAYDATA)                      ; Edited 25-Feb-94 16:44 by sybalsky

    ;; puts a character on a display stream.  Much of the information needed by the BitBlt microcode is prestored by the routines that change it.  This is
    ;; kept in the BitBltTable.

    ;; knows about the representation of a DisplayStream.

    (IMAGEOP 'IMBLTCHAR (SETQ DISPLAYSTREAM (\OUTSTREAMARG DISPLAYSTREAM))
           CHARCODE DISPLAYSTREAM DISPLAYDATA])
```

(**\MEDW.BLTCHAR**

```
  [LAMBDA (CHARCODE DISPLAYSTREAM DISPLAYDATA)                      (* kbr%: "25-Feb-86 22:25")

    ;; puts a character on a display stream.  Much of the information needed by the BitBlt microcode is prestored by the routines that change it.  This is
    ;; kept in the BitBltTable.

    ;; knows about the representation of a DisplayStream.

    (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 (\SLOWBLTCHAR CHARCODE 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 (freplace (PILOTBBT PBTDESTBIT) of LOCAL1 with LEFT)
                             (freplace (PILOTBBT PBTWIDTH) of LOCAL1 with (IDIFFERENCE RIGHT LEFT))
                             (freplace (PILOTBBT PBTSOURCEBIT) of LOCAL1 with (IDIFFERENCE (IPLUS (
                                                                                                  \DSPGETCHAROFFSET
                                                                                                   CHAR8CODE
                                                                                                   DISPLAYDATA)
                                                                                                  LEFT)
                                                                                           CURX))
                             (\PILOTBITBLT LOCAL1 0))
                      T])
```

(**\CHANGECHARSET.DISPLAY**

```
  [LAMBDA (DISPLAYDATA CHARSET)                                     (* gbn "13-Sep-85 11:47")

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

    (PROG [BM (PBT (ffetch DDPILOTBBT of DISPLAYDATA))
              (CSINFO (\GETCHARSETINFO CHARSET (ffetch DDFONT of DISPLAYDATA]

     ;; Since we called \GETCHARSETINFO without the NOSLUG? flag, we presume we will get back a CSINFO , even if it is a slug csinfo

          (UNINTERRUPTABLY
              (freplace DDWIDTHSCACHE of DISPLAYDATA with (ffetch (CHARSETINFO WIDTHS) of CSINFO))
              (freplace DDOFFSETSCACHE of DISPLAYDATA with (ffetch (CHARSETINFO OFFSETS) of CSINFO))
```

```
                (freplace DDCHARIMAGEWIDTHS of DISPLAYDATA with (ffetch (CHARSETINFO IMAGEWIDTHS) of CSINFO))
                (freplace DDCHARSET of DISPLAYDATA with CHARSET)
                (SETQ BM (ffetch CHARSETBITMAP of CSINFO))
                (freplace PBTSOURCEBPL of PBT with (UNFOLD (ffetch BITMAPRASTERWIDTH of BM)
                                                           BITSPERWORD))
                [COND
                   ((OR (NEQ (ffetch DDCHARSETASCENT of DISPLAYDATA)
                             (ffetch CHARSETASCENT of CSINFO))
                        (NEQ (ffetch DDCHARSETDESCENT of DISPLAYDATA)
                             (ffetch CHARSETDESCENT of CSINFO)))
                    (\SFFixY DISPLAYDATA CSINFO))
                   (T (freplace PBTSOURCE of PBT with (\ADDBASE (ffetch BITMAPBASE of BM)
                                                               (ITIMES (ffetch BITMAPRASTERWIDTH of BM)
                                                                      (ffetch DDCHARHEIGHTDELTA of DISPLAYDATA])])
```

(\INDICATESTRING

```
  [LAMBDA (CHARCODE)                                            (* jds " 3-Oct-85 16:50")

    ;; This returns the string of characters by which CHARCODE would be indicated on the display.  This could be fixed up to use a global resource
    ;; passed in from the outside, but this should almost never be called so it doesn't matter (except perhaps when SEEing a compiled file)

    (COND
       [(IGREATERP CHARCODE \MAXTHINCHAR)                       ; An NS character
        (RESETLST
            (RESETSAVE PRXFLT T)
            (RESETSAVE (RADIX 8))
            (CONCAT '%# (\CHARSET CHARCODE)
                   ","
                   (\CHAR8CODE CHARCODE)))]
       (T (CONCAT (COND
                     ((IGREATERP CHARCODE 127)                  ; An old META character
                      (SETQ CHARCODE (LOGAND CHARCODE 127))
                      '%#)
                     (T ""))
                 (COND
                    ((ILESSP CHARCODE 32)                       ; CONTROL character
                     (SETQ CHARCODE (LOGOR CHARCODE 64))
                     '^)
                    (T ""))
                 (CHARACTER CHARCODE])
```

(\SLOWBLTCHAR

```
  [LAMBDA (CHARCODE DISPLAYSTREAM)                              ; Edited  8-Nov-89 15:19 by gadener

    ;; 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)
          (SETQ CHAR8CODE (\CHAR8CODE CHARCODE))
          (SETQ DD (ffetch (STREAM IMAGEDATA) of DISPLAYSTREAM))
          (SETQ ROTATION (ffetch (FONTDESCRIPTOR ROTATION) of (ffetch (\DISPLAYDATA DDFONT) of DD)))
          (COND
             [(EQ 0 ROTATION)
              (PROG (NEWX LEFT RIGHT CURX PILOTBBT DESTBIT WIDTH SOURCEBIT)
                    (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)))
                    (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
                              (\DSPGETCHAROFFSET CHAR8CODE DD)
                              DISPLAYSTREAM
                              (IDIFFERENCE (ffetch (\DISPLAYDATA DDXPOSITION) of DD)
                                     (ffetch (CHARSETINFO CHARSETDESCENT) of CSINFO))
                              (ffetch (\DISPLAYDATA DDYPOSITION) of DD)
                              (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"])
```

(TEXTUREP

```
  [LAMBDA (OBJECT)                                                    (* bvm%: "26-MAY-82 17:51")
    (OR (FIXP OBJECT)
        (AND (type? BITMAP OBJECT)
             (EQ (fetch BITMAPRASTERWIDTH of OBJECT)
                 1)
             OBJECT])
```

(INVERT.TEXTURE

```
  [LAMBDA (TEXTURE SCRATCHBM)                                         (* bvm%: "31-MAY-82 14:41")
    (COND
       ((FIXP TEXTURE)
        (LOGXOR (LOGAND TEXTURE BLACKSHADE)
               BLACKSHADE))
       (T (INVERT.TEXTURE.BITMAP TEXTURE SCRATCHBM])
```

(INVERT.TEXTURE.BITMAP

```
  [LAMBDA (BM SCRATCHBM)                                              (* edited%: "15-SEP-82 09:17")

    ;; Returns a bitmap that is the complement of BM.  If SCRATCHBM is supplied, then does it to SCRATCHBM, else creates and returns a new
    ;; bitmap

    (COND
       ((NEQ (fetch BITMAPRASTERWIDTH of BM)
             1)
        (\ILLEGAL.ARG BM)))
    (PROG [(NEWBM (COND
                     ((type? BITMAP SCRATCHBM)
                      (COND
                         ((OR (NEQ (fetch BITMAPRASTERWIDTH of SCRATCHBM)
                                   1)
                              (IGREATERP (fetch BITMAPHEIGHT of BM)
                                     (fetch BITMAPHEIGHT of SCRATCHBM)))
                          (\ILLEGAL.ARG SCRATCHBM)))
                      SCRATCHBM)
                     (T (BITMAPCREATE BITSPERWORD (fetch BITMAPHEIGHT of BM]
          (bind (BASE1 _ (fetch BITMAPBASE of BM))
                (LASTBASE _ (\ADDBASE (fetch BITMAPBASE of NEWBM)
                                   (fetch BITMAPHEIGHT of BM)))
             for (BASE2 _ (fetch BITMAPBASE of NEWBM)) by (\ADDBASE BASE2 1) until (EQ BASE2 LASTBASE)
             do (\PUTBASE BASE2 0 (LOGXOR (\GETBASE BASE1 0)
                                          WORDMASK))
                (SETQ BASE1 (\ADDBASE BASE1 1)))
          (RETURN NEWBM])
```

(BITMAPWIDTH

```
  [LAMBDA (BITMAP)                                               (* kbr%: " 2-Sep-85 19:01")

    ;; returns the width of a bitmap in pixels

    (COND
       ((type? BITMAP BITMAP)
        (ffetch (BITMAP BITMAPWIDTH) of BITMAP))
       ((type? WINDOW BITMAP)
        (WINDOWPROP BITMAP 'WIDTH))
       (T (\ILLEGAL.ARG BITMAP])
```

(READBITMAP

```
  [LAMBDA (FILE)                                                 ; Edited  8-Aug-2021 00:18 by rmk:

;;; reads a bitmap from the input file.

    (SKIPSEPRS FILE)
    (OR (EQ (READC FILE)
            '%()
        (ERROR "BAD FORMAT OF BITMAP IN FILE"))
    (PROG [BASE BM W BITSPERPIXEL (WIDTH (RATOM FILE))
                (HEIGHT (RATOM FILE))
                (STRM (GETSTREAM FILE 'INPUT]
          [SETQ BITSPERPIXEL (SELECTQ (SKIPSEPRS STRM)
                                 ((%" %))
                                      1)
                                 (PROGN                          ; after height can come the bits per pixel.
                                        (RATOM FILE]
          (SETQ W (FOLDHI (ITIMES BITSPERPIXEL WIDTH)
                          BITSPERWORD))
          (SETQ BM (BITMAPCREATE WIDTH HEIGHT BITSPERPIXEL))
          (SETQ BASE (fetch BITMAPBASE of BM))
          (COND
             ((EQ HEIGHT 0))
             [(EQ (SKIPSEPRS STRM)
                  '%")
              (FRPTQ HEIGHT (SKIPSEPRS STRM)
                     (OR (EQ (\INCCODE STRM)
                             (CHARCODE %"))
                         (GO BAD))
                     (FRPTQ W [\PUTBASEBYTE BASE 0 (LOGOR (LLSH (IDIFFERENCE (\INCCODE STRM)
                                                                             (SUB1 (CHARCODE A)))
                                                                4)
                                                          (IDIFFERENCE (\INCCODE STRM)
                                                                 (SUB1 (CHARCODE A]
                            [\PUTBASEBYTE BASE 1 (LOGOR (LLSH (IDIFFERENCE (\INCCODE STRM)
                                                                           (SUB1 (CHARCODE A)))
                                                              4)
                                                        (IDIFFERENCE (\INCCODE STRM)
                                                               (SUB1 (CHARCODE A]
                            (SETQ BASE (\ADDBASE BASE 1)))
                     (OR (EQ (\INCCODE STRM)
                             (CHARCODE %"))
                         (GO BAD]
             (T (GO BAD)))
          (SKIPSEPRS STRM)
          (OR (EQ (\INCCODE STRM)
                  (CHARCODE %)))
              (GO BAD))
          (RETURN BM)
      BAD (ERROR "BAD FORMAT OF BITMAP IN FILE"])
```

(\INSUREBITSPERPIXEL

```
  [LAMBDA (NBITS)                                                (* kbr%: "10-Aug-85 15:49")

    ;; determines if NBITS is a legal color bits per pixel.

    (SELECTQ NBITS
         (NIL 1)
         ((1 4 8 24)
              NBITS)
         (\ILLEGAL.ARG NBITS])
```

(MAXIMUMCOLOR

```
  [LAMBDA (BITSPERPIXEL)                                         (* kbr%: "29-Jan-86 12:12")
    (MASK.1'S 0 BITSPERPIXEL])
```

(OPPOSITECOLOR

```
  [LAMBDA (COLOR BITSPERPIXEL)                                   (* kbr%: " 5-Jun-85 18:36")
    (IDIFFERENCE (MAXIMUMCOLOR BITSPERPIXEL)
           COLOR])
```

(MAXIMUMSHADE

```
  [LAMBDA (BITSPERPIXEL)                                         (* kbr%: " 5-Jun-85 18:37")
    (COND
       ((EQ BITSPERPIXEL 1)
        BLACKSHADE)
       (T (MAXIMUMCOLOR BITSPERPIXEL])
```

(OPPOSITESHADE

```
  [LAMBDA (SHADE BITSPERPIXEL)                                   (* kbr%: " 5-Jun-85 18:39")
    (IDIFFERENCE (MAXIMUMSHADE BITSPERPIXEL)
           SHADE])
```

(\MEDW.BITBLT

```
  [LAMBDA (SOURCE SOURCELEFT SOURCEBOTTOM DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT SOURCETYPE
                 OPERATION TEXTURE CLIPPINGREGION)                ; Edited 18-Jan-94 17:01 by nilsson
    (OR (IMAGESTREAMP SOURCE)
        (IMAGESTREAMP DESTINATION)
        (SHOULDNT "Neither SOURCE nor DESTINATION is an imagestream."))
    (COND
       ((BITMAPP SOURCE)
        (LET ((DSTWIN (WFROMDS DESTINATION T))
              (DD (\GETDISPLAYDATA DESTINATION)))
             (WINDOWOP 'BBTTOWIN (fetch (WINDOW SCREEN) of DSTWIN)
                    SOURCE SOURCELEFT SOURCEBOTTOM DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT
                    SOURCETYPE OPERATION TEXTURE CLIPPINGREGION SOURCELEFT SOURCEBOTTOM)))
       [(BITMAPP DESTINATION)
        (LET* ((SRCWIN (WFROMDS SOURCE T))
               (DD (\GETDISPLAYDATA SOURCE))
               (SOURCELEFTTRANSFORMED (OR (\DSPTRANSFORMX SOURCELEFT DD)
                                          SOURCELEFT))
               (SOURCEBOTTOMTRANSFORMED (OR (\DSPTRANSFORMY SOURCEBOTTOM DD)
                                            SOURCEBOTTOM)))
              (WINDOWOP 'BBTFROMWIN (fetch (WINDOW SCREEN) of SRCWIN)
                     (fetch (\DISPLAYDATA DDDestination) of DD)
                     SOURCELEFTTRANSFORMED SOURCEBOTTOMTRANSFORMED DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM
                     WIDTH HEIGHT SOURCETYPE OPERATION TEXTURE CLIPPINGREGION (IMAX SOURCELEFTTRANSFORMED
                                                                                  (fetch (\DISPLAYDATA
                                                                                                 DDClippingLeft)
                                                                                     of DD))
                     (IMAX SOURCEBOTTOMTRANSFORMED (fetch (\DISPLAYDATA DDClippingBottom) of DD]
       [(EQ (DSPDESTINATION NIL SOURCE)
            (DSPDESTINATION NIL DESTINATION))                    ; SOURCE and DESTINATION are on the same SCREEN.
                                                                 ; Optimized special case.

        ;; Make sure the windows are open and on  top

        ;; If they are overlapping use an intermediate bitmap, else just shovle bits.

        (LET* ((SRCWIN (WFROMDS SOURCE T))
               (DD (\GETDISPLAYDATA SOURCE))
               (SOURCELEFTTRANSFORMED (OR (\DSPTRANSFORMX SOURCELEFT DD)
                                          SOURCELEFT))
               (SOURCEBOTTOMTRANSFORMED (OR (\DSPTRANSFORMY SOURCEBOTTOM DD)
                                            SOURCEBOTTOM)))
              (\INSURETOPWDS SOURCE)
              (WINDOWOP 'BBTWINWIN (fetch (WINDOW SCREEN) of SRCWIN)
                     (fetch (\DISPLAYDATA DDDestination) of DD)
                     SOURCELEFTTRANSFORMED SOURCEBOTTOMTRANSFORMED DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM
                     WIDTH HEIGHT SOURCETYPE OPERATION TEXTURE CLIPPINGREGION (IMAX SOURCELEFTTRANSFORMED
                                                                                  (fetch (\DISPLAYDATA
                                                                                                 DDClippingLeft)
                                                                                     of DD))
                     (IMAX SOURCEBOTTOMTRANSFORMED (fetch (\DISPLAYDATA DDClippingBottom) of DD]
       (T (SHOULDNT "Invalid argument to \XW.BIBLT")))
    T])
)
```

(CL:DEFUN **FINISH-READING-BITMAP** (STREAM)

;;; The syntax for bitmaps is

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

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

;;; When we enter this function, called from HASH-STAR, the stream should be pointing at the (.

```
   (LET
    ((DIMENSIONS (READ STREAM)))
    (CL:ASSERT (CL:LISTP DIMENSIONS)
           '(DIMENSIONS)
           "BUG: FINISH-READING-BITMAP called with non-list on stream: ~S" DIMENSIONS)
    (DESTRUCTURING-BIND
     (WIDTH HEIGHT &OPTIONAL (BITS-PER-PIXEL 1)
            &REST EXTRAS)
     DIMENSIONS                                                       ; Parsing the dimensions.
     (IF (OR (NOT (FIXP WIDTH))
             (NOT (FIXP HEIGHT))
             (NOT (FIXP BITS-PER-PIXEL))
             (NOT (NULL EXTRAS)))
         THEN (CL:ERROR "Bad bitmap dimension specification: ~S" DIMENSIONS))
     (LET ((BITMAP NIL)
           (BASE NIL)
           (QUAD-CHARS-PER-ROW (FOLDHI (CL:* WIDTH BITS-PER-PIXEL)
                                       16)))
          [IF *READ-SUPPRESS*
              THEN (CL:DOTIMES (I (CL:* HEIGHT QUAD-CHARS-PER-ROW 4))
                          (CL:READ-CHAR STREAM))
            ELSE (SETQ BITMAP (BITMAPCREATE WIDTH HEIGHT BITS-PER-PIXEL))
                 (SETQ BASE (FETCH BITMAPBASE OF BITMAP))
                 (LET [(STREAM (\GETSTREAM STREAM 'INPUT]
                      (CL:DOTIMES (ROW HEIGHT)
                          [IF (ZEROP (FETCH (STREAM CHARSET) OF STREAM))
                              THEN                                    ; Do it the quicker way
                                   (CL:DOTIMES (QUAD QUAD-CHARS-PER-ROW)
                                       (LET [(NIB00 (- (\BIN STREAM)
                                                       (CHARCODE @)))
                                             (NIB01 (- (\BIN STREAM)
                                                       (CHARCODE @)))
                                             (NIB10 (- (\BIN STREAM)
                                                       (CHARCODE @)))
                                             (NIB11 (- (\BIN STREAM)
                                                       (CHARCODE @]
                                            (IF (OR (NOT (<= 0 NIB00 15))
                                                    (NOT (<= 0 NIB01 15))
                                                    (NOT (<= 0 NIB10 15))
                                                    (NOT (<= 0 NIB11 15)))
                                                THEN (CL:ERROR "Illegal character in bitmap contents
                                                             specification."))
                                            (\PUTBASEBYTE BASE 0 (LOGOR (LLSH NIB00 4)
                                                                        NIB01))
                                            (\PUTBASEBYTE BASE 1 (LOGOR (LLSH NIB10 4)
                                                                        NIB11)))
                                       (SETQ BASE (\ADDBASE BASE 1)))
                            ELSE                                      ; Somewhat slower...
                                 (CL:DOTIMES (QUAD QUAD-CHARS-PER-ROW)
                                     (LET [(NIB00 (- (READCCODE STREAM)
                                                     (CHARCODE @)))
                                           (NIB01 (- (READCCODE STREAM)
                                                     (CHARCODE @)))
                                           (NIB10 (- (READCCODE STREAM)
                                                     (CHARCODE @)))
                                           (NIB11 (- (READCCODE STREAM)
                                                     (CHARCODE @]
                                          (IF (OR (NOT (<= 0 NIB00 15))
                                                  (NOT (<= 0 NIB01 15))
                                                  (NOT (<= 0 NIB10 15))
                                                  (NOT (<= 0 NIB11 15)))
                                              THEN (CL:ERROR "Illegal character in bitmap contents
                                                           specification."))
                                          (\PUTBASEBYTE BASE 0 (LOGOR (LLSH NIB00 4)
                                                                      NIB01))
                                          (\PUTBASEBYTE BASE 1 (LOGOR (LLSH NIB10 4)
                                                                      NIB11)))
                                     (SETQ BASE (\ADDBASE BASE 1)))])]
          BITMAP))))

(DECLARE%: EVAL@COMPILE

(RPAQQ MINIMUMCOLOR 0)

(RPAQQ MINIMUMSHADE 0)

(CONSTANTS (MINIMUMCOLOR 0)
       (MINIMUMSHADE 0))
)

(MOVD 'BITMAPBIT '\BITMAPBIT)

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE
(PUTPROPS \INVALIDATEDISPLAYCACHE MACRO ((DISPLAYDATA)

           (* This marks the character-printing caches of the displaystream as invalid.
           Needed when the font or Y position changes)

                                          (freplace (\DISPLAYDATA DDCHARSET) of DISPLAYDATA with MAX.SMALLP)
                                          (freplace (\DISPLAYDATA DDCHARSETASCENT) of DISPLAYDATA with MAX.SMALLP)
                                          ))
)
)

;; END EXPORTED DEFINITIONS


(DEFOPTIMIZER BITMAPBIT (&REST ARGS)
                        (BITMAPBIT.EXPANDER ARGS))


(DEFOPTIMIZER BITMAPP (Y)
                      `((OPENLAMBDA (X)
                          (AND (type? BITMAP X)
                               X))
                        ,Y))

(DEFINEQ

(BITMAPBIT.EXPANDER
  [LAMBDA (ARGS)                                                   (* hdj "19-Mar-85 12:14")
    (PROG ((BM (CAR ARGS))
           (X (CADR ARGS))
           (Y (CADDR ARGS))
           NEWVALUE)
          [COND
             ((EQ (LENGTH ARGS)
                  4)
              (SETQ NEWVALUE (CADDDR ARGS]
          (RETURN `((OPCODES MISC4 6)
                    ,BM
                    ,X
                    ,Y
                    ,NEWVALUE])
)

(DEFINEQ

(\BITBLT.DISPLAY
  [LAMBDA (SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT
                 SOURCETYPE OPERATION TEXTURE CLIPPINGREGION)      ; Edited 16-Feb-94 10:28 by sybalsky
    (DECLARE (LOCALVARS . T))
    (PROG (SOURCEDD SOURCE SOURCEIMAGEOPS CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM)
          [COND
             [(type? BITMAP SOURCEBITMAP)
              (OR SOURCELEFT (SETQ SOURCELEFT 0))
              (OR SOURCEBOTTOM (SETQ SOURCEBOTTOM 0))
              (SETQ CLIPPEDSOURCELEFT SOURCELEFT)
              (SETQ CLIPPEDSOURCEBOTTOM SOURCEBOTTOM)              ; limit the WIDTH and HEIGHT to the source size.
              [SETQ WIDTH (COND
                             (WIDTH (IMIN WIDTH (IDIFFERENCE (fetch (BITMAP BITMAPWIDTH) of SOURCEBITMAP)
                                                      SOURCELEFT)))
                             (T (fetch (BITMAP BITMAPWIDTH) of SOURCEBITMAP]
              (SETQ HEIGHT (COND
                              (HEIGHT (IMIN HEIGHT (IDIFFERENCE (fetch (BITMAP BITMAPHEIGHT) of SOURCEBITMAP)
                                                        SOURCEBOTTOM)))
                              (T (fetch (BITMAP BITMAPHEIGHT) of SOURCEBITMAP]
             ((SETQ SOURCEDD (\GETDISPLAYDATA SOURCEBITMAP))
              (SETQ SOURCE SOURCEBITMAP)
              [OR SOURCELEFT (SETQ SOURCELEFT (fetch (REGION LEFT) of (ffetch (\DISPLAYDATA DDClippingRegion)
                                                                             of SOURCEDD]
              [OR SOURCEBOTTOM (SETQ SOURCEBOTTOM (fetch (REGION BOTTOM) of (ffetch (\DISPLAYDATA DDClippingRegion)
                                                                                   of SOURCEDD]
                                                                   ; do transformations coming out of source
              (SETQ SOURCEBITMAP (fetch (\DISPLAYDATA DDDestination) of SOURCEDD))
              (SETQ CLIPPEDSOURCELEFT (IMAX (SETQ SOURCELEFT (\DSPTRANSFORMX SOURCELEFT SOURCEDD))
                                            (fetch (\DISPLAYDATA DDClippingLeft) of SOURCEDD)))
              (SETQ CLIPPEDSOURCEBOTTOM (IMAX (SETQ SOURCEBOTTOM (\DSPTRANSFORMY SOURCEBOTTOM SOURCEDD))
                                              (fetch (\DISPLAYDATA DDClippingBottom) of SOURCEDD)))
                                                                   ; limit the WIDTH and HEIGHT by the source dimensions.
              [SETQ WIDTH (COND
                             (WIDTH (IMIN WIDTH (IDIFFERENCE (fetch (\DISPLAYDATA DDClippingRight) of SOURCEDD)
                                                      CLIPPEDSOURCELEFT)))
                             (T (IDIFFERENCE (fetch (\DISPLAYDATA DDClippingRight) of SOURCEDD)
                                       CLIPPEDSOURCELEFT]
              [SETQ HEIGHT (COND
                              (HEIGHT (IMIN HEIGHT (IDIFFERENCE (fetch (\DISPLAYDATA DDClippingTop) of SOURCEDD)
                                                        CLIPPEDSOURCEBOTTOM)))
                              (T (IDIFFERENCE (fetch (\DISPLAYDATA DDClippingTop) of SOURCEDD)
                                          CLIPPEDSOURCEBOTTOM]       ; if texture is not given, use the display stream's.
              (OR TEXTURE (SETQ TEXTURE (ffetch (\DISPLAYDATA DDTexture) of SOURCEDD]
          (COND
             ((OR (IGEQ 0 WIDTH)
                  (IGEQ 0 HEIGHT))                              ; if either width or height is 0, don't do anything.
              (RETURN)))
          (RETURN (COND
                     [(type? BITMAP DESTINATION)
                      (COND
                         ((WINDOWP SOURCE)

                          ;; bring source window to the top.  Note: this doesn't work if the user passes in a display stream onto the screen
                          ;; instead of a window.

                          (.WHILE.TOP.DS. (\OUTSTREAMARG SOURCEBITMAP)
                                 (\BITBLT.BITMAP SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTINATION DESTINATIONLEFT
                                        DESTINATIONBOTTOM WIDTH HEIGHT SOURCETYPE OPERATION TEXTURE
                                        CLIPPINGREGION CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM)))
                         (T (\BITBLT.BITMAP SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTINATION DESTINATIONLEFT
                                   DESTINATIONBOTTOM WIDTH HEIGHT SOURCETYPE OPERATION TEXTURE CLIPPINGREGION
                                   CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM]
                     (T (PROG (DESTSTRM)
                              (SETQ DESTSTRM (\OUTSTREAMARG DESTINATION))
                              (COND
                                 ([AND (NEQ SOURCE DESTINATION)
                                       (OR (WINDOWP SOURCE)
                                           (AND SOURCE (WFROMDS SOURCE]

                                  ;; both source and destination are windows, see if they overlap and use an intermediate bitmap.  Note: this
                                  ;; doesn't work if the user passes in a display stream onto the screen instead of a window.  ALSO use
                                  ;; bitmap if the destination is on a different screen.

                                  [COND
                                     ((WINDOWP DESTINATION)
                                      (COND
                                         ([AND (WOVERLAPP SOURCE DESTINATION)
                                               (EQ (FETCH (STREAM IMAGEOPS) OF (WINDOWPROP SOURCE 'DSP))
                                                   (FETCH (STREAM IMAGEOPS) OF (WINDOWPROP DESTINATION
                                                                                        'DSP]
                                          (RETURN (PROG (SCRATCHBM)
                                                        (.WHILE.TOP.DS. (\OUTSTREAMARG SOURCE)
                                                               (BITBLT SOURCEBITMAP SOURCELEFT SOURCEBOTTOM
                                                                      (SETQ SCRATCHBM (BITMAPCREATE WIDTH
                                                                                             HEIGHT))
                                                                      0 0 WIDTH HEIGHT 'INPUT 'REPLACE))
                                                        (RETURN (BITBLT SCRATCHBM 0 0 DESTSTRM DESTINATIONLEFT
                                                                       DESTINATIONBOTTOM WIDTH HEIGHT SOURCETYPE
                                                                       OPERATION TEXTURE CLIPPINGREGION]
                                                                   ; bring the source to the top.  this should be done uninterruptably
                                                                   ; but is better than nothing.
                                  (TOTOPW SOURCE)))
                              (COND
                                 ((OR (NOT SOURCE)
                                      (EQ (DSPDESTINATION NIL SOURCE)
                                          (DSPDESTINATION NIL DESTSTRM)))
                                  (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 (fetch (\DISPLAYDATA DDClippingBottom) of DESTDD))
                                               (SETQ right (fetch (\DISPLAYDATA DDClippingRight) of DESTDD))
                                               (SETQ top (fetch (\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
                                                                                        (fetch (REGION BOTTOM)
                                                                                           of CLIPPINGREGION)
```

```
                                                                     DESTDD]
                                        [SETQ right (IMIN right
                                                         (IPLUS CRLEFT
                                                                (fetch (REGION WIDTH)
                                                                   of CLIPPINGREGION]
                                        (SETQ top (IMIN top (IPLUS CRBOTTOM
                                                                   (fetch (REGION HEIGHT)
                                                                      of CLIPPINGREGION]
                          (SETQ DESTINATIONNBITS (fetch (BITMAP BITMAPBITSPERPIXEL) of DESTBITMAP))
                          (SETQ SOURCENBITS (ffetch (BITMAP BITMAPBITSPERPIXEL) of SOURCEBITMAP))
                          [COND
                             ((NOT (EQ SOURCENBITS DESTINATIONNBITS))
                              (COND
                                 ((EQ SOURCENBITS 1)
                                  (SETQ SOURCEBITMAP (COLORIZEBITMAP SOURCEBITMAP 0 (
                                                                                 MAXIMUMCOLOR

                                                                               DESTINATIONNBITS
                                                                                 )
                                                     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 (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)))
                          (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
                                     ((NOT (EQ DESTINATIONNBITS 1))
                                      (SETQ left (ITIMES DESTINATIONNBITS left))
                                      (SETQ right (ITIMES DESTINATIONNBITS right))
                                      (SETQ stodx (ITIMES DESTINATIONNBITS stodx]
                                  [.WHILE.TOP.DS. DESTSTRM
                                         (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]
                                  (RETURN T)))
                             (T (IMAGEOP 'IMBITBLT DESTSTRM SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTSTRM
                                       DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT SOURCETYPE OPERATION
                                       TEXTURE CLIPPINGREGION CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM])
```

(\BITBLT.BITMAP

```
  [LAMBDA (SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTBITMAP DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT
                 SOURCETYPE OPERATION TEXTURE CLIPPINGREGION CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM)
                                                         (* kbr%: "15-Feb-86 20:21")
    (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])
```

```
(\BITBLT.MERGE
  [LAMBDA (PILOTBBT SOURCEBITMAP SLX STY DESTBITMAP DLX DTY WIDTH HEIGHT OPERATION TEXTURE)
                                                                    (* rmk%: "21-Jun-84 23:10")

    ;; Can't do MERGE in Pilot bitblt, so simulate by blting source to scratch bitmap, erasing bits not in Texture, then blting scratch to ultimate
    ;; destination.  Note that TEXTURE has already been complemented above in preparation for this

    (COND
       ((AND (EQ OPERATION 'REPLACE)
             (NEQ SOURCEBITMAP DESTBITMAP))                         ; Don't need a scratch bitmap, just do two blts
        (\BITBLTSUB PILOTBBT SOURCEBITMAP SLX STY DESTBITMAP DLX DTY HEIGHT 'INPUT 'REPLACE)
                                                                    ; Blt the source, then erase bits that aren't in TEXTURE
        (\BITBLTSUB PILOTBBT NIL NIL NIL DESTBITMAP DLX DTY HEIGHT 'TEXTURE 'ERASE TEXTURE))
       (T (PROG (SCRATCH (SCRATCHLEFT (MOD DLX BITSPERWORD))
                      (SCRATCHTOP (MOD DTY 4)))
                (SETQ SCRATCH (\GETPILOTBBTSCRATCHBM (IPLUS WIDTH SCRATCHLEFT)
                                     (IPLUS HEIGHT SCRATCHTOP)))    ; Get scratch bm, slightly larger than WIDTH and HEIGHT to
                                                                    ; allow texture to align
                (\BITBLTSUB PILOTBBT SOURCEBITMAP SLX STY SCRATCH SCRATCHLEFT SCRATCHTOP HEIGHT 'INPUT
                       'REPLACE)                                    ; Blt source into scratch
                (\BITBLTSUB PILOTBBT NIL NIL NIL SCRATCH SCRATCHLEFT SCRATCHTOP HEIGHT 'TEXTURE 'ERASE TEXTURE)
                                                                    ; Erase what isn't in TEXTURE
                                                                    ; Finally do original operation using the merged source
                (\BITBLTSUB PILOTBBT SCRATCH SCRATCHLEFT SCRATCHTOP DESTBITMAP DLX DTY HEIGHT 'INPUT OPERATION])

(\BLTSHADE.DISPLAY
  [LAMBDA (TEXTURE STREAM DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT OPERATION CLIPPINGREGION)
                                                                    ; Edited 28-Jan-93 17:33 by jds
                                                                    ; BLTSHADE to a display stream
    (DECLARE (LOCALVARS . T))
    (PROG (left top bottom right DESTINATIONBITMAP DESTDD DESTINATIONNBITS)
          (SETQ DESTDD (fetch (STREAM IMAGEDATA) of STREAM))

     ;; 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 STREAM)
          (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 (fetch (\DISPLAYDATA DDClippingBottom) of DESTDD))
                 (SETQ right (fetch (\DISPLAYDATA DDClippingRight) of DESTDD))
                 (SETQ top (fetch (\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 (fetch (REGION BOTTOM)
                                                                                             of CLIPPINGREGION)
                                                                                 DESTDD]
                                 [SETQ right (IMIN right (IPLUS CRLEFT (fetch (REGION WIDTH) of CLIPPINGREGION]
                                 (SETQ top (IMIN top (IPLUS CRBOTTOM (fetch (REGION HEIGHT) of CLIPPINGREGION]
          (SETQ DESTINATIONBITMAP (fetch (\DISPLAYDATA DDDestination) of DESTDD))
          (SETQ DESTINATIONNBITS (fetch (BITMAP BITMAPBITSPERPIXEL) of DESTINATIONBITMAP))

     ;; 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 NEW-ATOM)
                                  (COND
                                     ((NULL TEXTURE)                ; NIL case.  default texture to background texture.
                                      (ffetch (\DISPLAYDATA DDTexture) of DESTDD))
                                     ((NOT (EQ DESTINATIONNBITS 1))
                                                                    ; should be a color name
                                      (OR (COLORNUMBERP TEXTURE DESTINATIONNBITS T)
                                          (\ILLEGAL.ARG TEXTURE)))
                                     (T (\ILLEGAL.ARG TEXTURE))))
```

```
                              ((SMALLP FIXP)
                                   (LOGAND TEXTURE (MAXIMUMSHADE DESTINATIONNBITS)))
                              (BITMAP TEXTURE)
                              (LISTP                              ; should be a list of levels rgb or hls.
                                     (OR (AND (NOT (EQ DESTINATIONNBITS 1))
                                              (COLORNUMBERP TEXTURE DESTINATIONNBITS))
                                         (\ILLEGAL.ARG TEXTURE)))
                              (\ILLEGAL.ARG TEXTURE)))
              [COND
                 ((NOT (EQ DESTINATIONNBITS 1))
                  (SETQ left (ITIMES DESTINATIONNBITS left))
                  (SETQ right (ITIMES DESTINATIONNBITS right))
                  (SETQ TEXTURE (COLORTEXTUREFROMCOLOR# TEXTURE DESTINATIONNBITS]
              [.WHILE.TOP.DS. STREAM (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
                                                  (OR OPERATION (ffetch (\DISPLAYDATA DDOPERATION) of DESTDD))
                                                  TEXTURE
                                                  (ITIMES DESTINATIONNBITS (fetch (\DISPLAYDATA DDXOFFSET)
                                                                                  of DESTDD))
                                                  (fetch (\DISPLAYDATA DDYOFFSET) of DESTDD]
              (RETURN T])
```

(\BLTSHADE.BITMAP

```
  [LAMBDA (TEXTURE DESTINATIONBITMAP DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT OPERATION CLIPPINGREGION)
                                                                  ; Edited 28-Jan-93 17:38 by jds
    (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 NEW-ATOM)                 ; 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])

)

(DEFINEQ

(\BITBLT.BITMAP.SLOW
  [LAMBDA (SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTBITMAP DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT
                 SOURCETYPE OPERATION TEXTURE CLIPPINGREGION CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM)
                                                                 ; Edited 11-Apr-90 15:23 by nm

    ;; Copy of \BITBLT.BITMAP.  Used to smash the definition of \MAIKO.OLDBITBLT.BITMAP.
                                                                 ("kbr%:" "15-Feb-86 20:21")
    (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

(\PUNT.BLTSHADE.BITMAP
  [LAMBDA (TEXTURE DESTINATIONBITMAP DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT OPERATION CLIPPINGREGION)
                                                                        ; Edited 28-Jan-93 17:38 by jds

    ;; 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 NEW-ATOM)                         ; 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

(\SCALEDBITBLT.DISPLAY
  [LAMBDA (SOURCEBITMAP SOURCELEFT SOURCEBOTTOM DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM WIDTH HEIGHT
                 SOURCETYPE OPERATION TEXTURE CLIPPINGREGION CLIPPEDSOURCELEFT CLIPPEDSOURCEBOTTOM SCALE)
                                                                   ; Edited 28-Mar-90 18:49 by jds
    (LET (BITMAP REGION)
         (IF (NULL SCALE)
             THEN (SETQ SCALE 1))
         (IF (WINDOWP SOURCEBITMAP)
             THEN (SETQ REGION (DSPCLIPPINGREGION NIL SOURCEBITMAP))
                  (IF (NULL WIDTH)
                      THEN (SETQ WIDTH (FETCH (REGION WIDTH) OF REGION)))
                  (IF (NULL HEIGHT)
                      THEN (SETQ HEIGHT (FETCH (REGION HEIGHT) OF REGION)))
           ELSEIF (BITMAPP SOURCEBITMAP)
             THEN (IF (NULL WIDTH)
                      THEN (SETQ WIDTH (BITMAPWIDTH SOURCEBITMAP)))
                  (IF (NULL HEIGHT)
                      THEN (SETQ HEIGHT (BITMAPHEIGHT SOURCEBITMAP)))
           ELSE (SHOULDNT))
         (OR DESTINATIONBOTTOM (SETQ DESTINATIONBOTTOM (DSPYPOSITION NIL DESTINATION)))
         (OR DESTINATIONLEFT (SETQ DESTINATIONLEFT (DSPXPOSITION NIL DESTINATION)))
         (SETQ BITMAP (BITMAPCREATE WIDTH HEIGHT))
         (BITBLT SOURCEBITMAP SOURCELEFT SOURCEBOTTOM BITMAP)
         (BITBLT (EXPANDBITMAP BITMAP SCALE SCALE)
                NIL NIL DESTINATION DESTINATIONLEFT DESTINATIONBOTTOM (TIMES WIDTH SCALE)
                (TIMES HEIGHT SCALE)
                SOURCETYPE OPERATION TEXTURE CLIPPINGREGION])
```

(\BACKCOLOR.DISPLAY

```
  [LAMBDA (DISPLAYSTREAM TEXTURE)                                    ; Edited 15-Feb-94 16:50 by nilsson
    (PROG (DD BITSPERPIXEL)
          (SETQ DD (\GETDISPLAYDATA DISPLAYSTREAM))
          (RETURN (PROG1 (fetch (\DISPLAYDATA DDTexture) of DD)
                      (COND
                         ((NULL TEXTURE))
                         ((AND (BITMAPP TEXTURE)
                               (EQ (fetch (BITMAP BITMAPRASTERWIDTH) of TEXTURE)
                                   1)
                               (ILEQ (BITMAPHEIGHT TEXTURE)
                                     16))                                ; allow small bitmaps
                          (freplace (\DISPLAYDATA DDTexture) of DD with TEXTURE))
                         ((FIXP TEXTURE)
                          (freplace (\DISPLAYDATA DDTexture) of DD with (LOGAND TEXTURE WORDMASK)))
                         ((NOT (EQ (SETQ BITSPERPIXEL (fetch (BITMAP BITMAPBITSPERPIXEL)
                                                         of (fetch (\DISPLAYDATA DDDestination) of DD)))
                                   1))
                          (freplace (\DISPLAYDATA DDTexture) of DD with (COLORNUMBERP TEXTURE BITSPERPIXEL)))
                         (T (\ILLEGAL.ARG TEXTURE))))])
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RPAQQ \DisplayWordAlign 16)

(RPAQQ \MaxBitMapWidth 65535)

(RPAQQ \MaxBitMapHeight 65535)

(RPAQQ \MaxBitMapWords 131066)

(CONSTANTS (\DisplayWordAlign 16)
       (\MaxBitMapWidth 65535)
       (\MaxBitMapHeight 65535)
       (\MaxBitMapWords 131066))
)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(PUTPROPS \DSPGETCHARWIDTH MACRO ((CHARCODE DD)
                                  (\FGETWIDTH (ffetch (\DISPLAYDATA DDWIDTHSCACHE) of DD)
                                         CHARCODE)))

(PUTPROPS \DSPGETCHARIMAGEWIDTH MACRO ((CHARCODE DD)
                                       (\FGETIMAGEWIDTH (ffetch (\DISPLAYDATA DDCHARIMAGEWIDTHS) of DD)
                                              CHARCODE)))

(PUTPROPS \DSPGETCHAROFFSET MACRO ((CHARCODE DD)
                                   (\GETBASE (ffetch (\DISPLAYDATA DDOFFSETSCACHE) of DD)
                                          CHARCODE)))

(PUTPROPS \CONVERTOP MACRO ((OP)                                   (* rrb "14-NOV-80 11:14")
                                                                   (* Only for alto bitblt !!)
                            (SELECTQ OP
                                (replace 0 of NIL with NIL)
                                (PAINT 1)
                                (INVERT 2)
                                (ERASE 3)
                                0)))

(PUTPROPS \SFInvert MACRO ((BitMap y)

          (* corrects for the fact that alto bitmaps are stored with 0,0 as upper left while lisp bitmaps have 0,0 as lower left.
          The correction is actually off by one (greater) because a majority of the places that it is called actually need one more than
          corrected Y value.)

                           (IDIFFERENCE (fetch (BITMAP BITMAPHEIGHT) of BitMap)
                                  y)))

(PUTPROPS \SFReplicate MACRO [LAMBDA (pattern)
                               (LOGOR pattern (LLSH pattern 8)
                                      (SETQ pattern (LLSH pattern 4))
                                      (LLSH pattern 8])
```

```
(PUTPROPS \SETPBTFUNCTION MACRO [OPENLAMBDA (BBT SourceType Operation)
                                   (PROGN (replace (PILOTBBT PBTOPERATION) of BBT with (SELECTQ Operation
                                                                                             (ERASE 1)
                                                                                             (PAINT 2)
                                                                                             (INVERT 3)
                                                                                             0))
                                          (replace (PILOTBBT PBTSOURCETYPE) of BBT
                                             with (COND
                                                     ((EQ (EQ SourceType 'INVERT)
                                                          (EQ Operation 'ERASE))
                                                      0)
                                                     (T 1])

(PUTPROPS \BITBLT1 MACRO ((bbt)
                          (BitBltSUBR bbt)))
)
```

;; END EXPORTED DEFINITIONS

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \SYSBBTEXTURE \BBSCRATCHTEXTURE \SYSPILOTBBT \PILOTBBTSCRATCHBM)
)
)

(RPAQQ \BBSCRATCHTEXTURE NIL)

(RPAQQ \PILOTBBTSCRATCHBM NIL)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(MOVD? 'BITBLT 'BKBITBLT)
)
```

;; macro for this file so that BITBLT can be broken by users
;; FOLLOWING DEFINITIONS EXPORTED

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

(PUTPROP 'BITBLT 'MACRO '(= . BKBITBLT))
)
```

;; END EXPORTED DEFINITIONS
;; display stream functions

```
(DEFINEQ

(DISPLAYSTREAMP
  [LAMBDA (X)                                                   ; Edited 19-Feb-87 11:03 by rrb
                                                                ; Is X a displaystream?
    (AND (type? STREAM X)
         [OR (FMEMB (fetch (IMAGEOPS IMAGETYPE) of (fetch (STREAM IMAGEOPS) of X))
                    \DISPLAYSTREAMTYPES)
             (SOME (fetch (IMAGEOPS IMAGETYPE) of (fetch (STREAM IMAGEOPS) of X))
                   (FUNCTION (LAMBDA (STYPE)
                               (FMEMB STYPE \DISPLAYSTREAMTYPES]
         X])

(DSPSOURCETYPE
  [LAMBDA (SOURCETYPE DISPLAYSTREAM)                            (* rmk%: "21-AUG-83 22:34")

    ;; sets the operation field of a display stream

    (PROG ((DD (\GETDISPLAYDATA DISPLAYSTREAM)))
          (RETURN (PROG1 (fetch DDSOURCETYPE of DD)
                      [COND
                         (SOURCETYPE (OR (FMEMB SOURCETYPE '(INPUT INVERT))
                                         (LISPERROR "ILLEGAL ARG" SOURCETYPE))
                                (UNINTERRUPTABLY
                                    (freplace DDSOURCETYPE of DD with SOURCETYPE)
                                                                ; update other fields that depend on operation.
                                    (\SETPBTFUNCTION (fetch DDPILOTBBT of DD)
                                           SOURCETYPE
                                           (fetch DDOPERATION of DD)))])])

(DSPXOFFSET
  [LAMBDA (XOFFSET DISPLAYSTREAM)                               ; Edited 17-Apr-94 23:46 by sybalsky

    ;; coordinate position is stored in 15 bits in the range -2^15 to +2^15.

    (IMAGEOP 'IMXOFFSET (SETQ DISPLAYSTREAM (\OUTSTREAMARG DISPLAYSTREAM))
           XOFFSET DISPLAYSTREAM])

(DSPYOFFSET
  [LAMBDA (YOFFSET DISPLAYSTREAM)                                    ; Edited 17-Apr-94 23:46 by sybalsky
    ;; coordinate position is stored in 15 bits in the range -2^15 to +2^15.
    (IMAGEOP 'IMYOFFSET (SETQ DISPLAYSTREAM (\OUTSTREAMARG DISPLAYSTREAM))
           YOFFSET DISPLAYSTREAM])

)

(DEFINEQ

(DSPDESTINATION
  [LAMBDA (DESTINATION DISPLAYSTREAM)
    (DECLARE (GLOBALVARS \DISPLAYIMAGEOPS \4DISPLAYIMAGEOPS \8DISPLAYIMAGEOPS \24DISPLAYIMAGEOPS
                    \XDISPLAYIMAGEOPS))                              ; Edited 28-Oct-93 13:23 by nilsson
    (PROG (DD)
          (SETQ DD (\GETDISPLAYDATA DISPLAYSTREAM DISPLAYSTREAM))
          (RETURN (PROG1 (ffetch (\DISPLAYDATA DDDestination) of DD)
                      [COND
                         (DESTINATION (UNINTERRUPTABLY
                                          (replace (STREAM DEVICE) of DISPLAYSTREAM
                                             with (CL:TYPECASE DESTINATION
                                                      (BITMAP (SELECTQ (fetch (BITMAP BITMAPBITSPERPIXEL)
                                                                          of DESTINATION)
                                                                  (1 DisplayFDEV)
                                                                  (4 \4DISPLAYFDEV)
                                                                  (8 \8DISPLAYFDEV)
                                                                  (24 \24DISPLAYFDEV)
                                                                  (SHOULDNT)))
                                                      (SCREEN XDisplayFDEV)))
                                          (replace (STREAM IMAGEOPS) of DISPLAYSTREAM
                                             with (CL:TYPECASE DESTINATION
                                                      (BITMAP (SELECTQ (fetch (BITMAP BITMAPBITSPERPIXEL)
                                                                          of DESTINATION)
                                                                  (1 \DISPLAYIMAGEOPS)
                                                                  (4 \4DISPLAYIMAGEOPS)
                                                                  (8 \8DISPLAYIMAGEOPS)
                                                                  (24 \24DISPLAYIMAGEOPS)
                                                                  (SHOULDNT)))
                                                      (SCREEN \XDISPLAYIMAGEOPS)))
                                          (freplace (\DISPLAYDATA DDDestination) of DD with DESTINATION)
                                          (CL:TYPECASE DESTINATION
                                              (BITMAP (\SFFixDestination DD DISPLAYSTREAM))
                                              (SCREEN            ; do it by hand
                                                      ))))])])

(DSPTEXTURE
  [LAMBDA (TEXTURE DISPLAYSTREAM)                                    ; Edited 15-Feb-94 16:50 by nilsson
    (DSPBACKCOLOR TEXTURE DISPLAYSTREAM])

(\DISPLAYSTREAMINCRXPOSITION
  [LAMBDA (N DD)                                                     (* rmk%: "23-AUG-83 14:12")
    ;; increases the x position by N.  This is used internally.  Returns the new value.
    (add (fetch DDXPOSITION of DD)
         N])

(\SFFixDestination
  [LAMBDA (DISPLAYDATA DISPLAYSTREAM)                                (* kbr%: "29-Jan-86 10:59")
    ;; fixes up those parts of the bitblt array which are dependent upon the destination
    (PROG ((PBT (ffetch (\DISPLAYDATA DDPILOTBBT) of DISPLAYDATA))
           (BM (ffetch (\DISPLAYDATA DDDestination) of DISPLAYDATA)))
          (replace (PILOTBBT PBTDESTBPL) of PBT with (UNFOLD (ffetch (BITMAP BITMAPRASTERWIDTH) of BM)
                                                             BITSPERWORD))
                                                                     ; line width information will be updated by \SFFixFont
          (\SFFixClippingRegion DISPLAYDATA)
          (\INVALIDATEDISPLAYCACHE DISPLAYDATA)
          (\SFFixFont DISPLAYSTREAM DISPLAYDATA)
          (RETURN])

(\SFFixClippingRegion
  [LAMBDA (DISPLAYDATA)                                              (* kbr%: "29-Jan-86 11:01")
    ;; compute the top, bottom, left and right edges of the clipping region in destination coordinates to save computation every BltChar and coordinate
    ;; transformation taking into account the size of the bit map as well as the clipping region.
    (PROG ((CLIPREG (ffetch (\DISPLAYDATA DDClippingRegion) of DISPLAYDATA))
           (BM (ffetch (\DISPLAYDATA DDDestination) of DISPLAYDATA)))
          [freplace (\DISPLAYDATA DDClippingRight) of DISPLAYDATA
             with (IMAX 0 (IMIN (\DSPTRANSFORMX (IPLUS (ffetch (REGION LEFT) of CLIPREG)
                                                       (ffetch (REGION WIDTH) of CLIPREG))
                                       DISPLAYDATA)
                               (ffetch (BITMAP BITMAPWIDTH) of BM]
          (freplace (\DISPLAYDATA DDClippingLeft) of DISPLAYDATA with (IMIN (IMAX (\DSPTRANSFORMX
                                                                                     (ffetch (REGION LEFT)
                                                                                        of CLIPREG)
                                                                                     DISPLAYDATA)
                                                                                   0)
                                                                             MAX.SMALL.INTEGER))
          [freplace (\DISPLAYDATA DDClippingTop) of DISPLAYDATA
             with (IMAX 0 (IMIN (\DSPTRANSFORMY (IPLUS (ffetch (REGION BOTTOM) of CLIPREG)
                                                       (ffetch (REGION HEIGHT) of CLIPREG))
                                       DISPLAYDATA)
                                 (ffetch (BITMAP BITMAPHEIGHT) of BM]
          (freplace (\DISPLAYDATA DDClippingBottom) of DISPLAYDATA with (IMIN (IMAX (\DSPTRANSFORMY
                                                                                        (ffetch (REGION BOTTOM)
                                                                                           of CLIPREG)
                                                                                        DISPLAYDATA)
                                                                                     0)
                                                                               MAX.SMALL.INTEGER])
```

(\SFFixFont

```
  [LAMBDA (DISPLAYSTREAM DISPLAYDATA)                                   (* kbr%: "29-Jan-86 11:03")

    ;; used to fix up those parts of the bitblt table which depend upon the FONT.  DISPLAYDATA is the IMAGEDATA for DISPLAYSTREAM, for
    ;; convenience.

    [PROG [(PILOTBBT (ffetch (\DISPLAYDATA DDPILOTBBT) of DISPLAYDATA))
           (FONT (ffetch (\DISPLAYDATA DDFONT) of DISPLAYDATA))
           (BITSPERPIXEL (ffetch (BITMAP BITMAPBITSPERPIXEL) of (ffetch (\DISPLAYDATA DDDestination) of DISPLAYDATA]
          (freplace (\DISPLAYDATA DDSlowPrintingCase) of DISPLAYDATA
             with (OR (NOT (EQ BITSPERPIXEL 1))
                      (NOT (EQ (ffetch (FONTDESCRIPTOR ROTATION) of FONT)
                               0]
    (\INVALIDATEDISPLAYCACHE DISPLAYDATA)
    (\SFFIXLINELENGTH DISPLAYSTREAM])
```

(\SFFIXLINELENGTH

```
  [LAMBDA (DISPLAYSTREAM)                                               ; Edited  5-Jan-88 12:57 by sye

    ;; DISPLAYSTREAM is known to be a stream of type display.  Called by RIGHTMARGIN LEFTMARGIN and \SFFIXFONT to update the
    ;; LINELENGTH field in the stream.  also called when the display stream is created.

    (PROG ((DD (fetch IMAGEDATA of DISPLAYSTREAM)))
          [freplace (STREAM LINELENGTH) of DISPLAYSTREAM with (IMIN MAX.SMALLP
                                                                    (IMAX 1 (IQUOTIENT (IDIFFERENCE
                                                                                        (ffetch (\DISPLAYDATA
                                                                                                   DDRightMargin
                                                                                                   )
                                                                                           of DD)
                                                                                        (ffetch (\DISPLAYDATA
                                                                                                   DDLeftMargin)
                                                                                           of DD))
                                                                                     (fetch FONTAVGCHARWIDTH
                                                                                        of (ffetch DDFONT of DD]

     ;;  make sure %SYNONYM-STREAM-DEVICE was defined (during the LOADUP) before updating

     ;;  LINELENGTH fields of DISPLAYSTREAM's synonym streams

          (AND (BOUNDP '%%SYNONYM-STREAM-DEVICE)
               (\UPDATE-SYNONYM-STREAM-LINELENGTH-FIELD DISPLAYSTREAM])
```

(\UPDATE-SYNONYM-STREAM-LINELENGTH-FIELD

```
  [LAMBDA (DISPLAYSTREAM)                                               ; Edited 19-Jan-88 15:48 by amd

    ;; copy the value of LINELENGTH field from DISPLAYSTREAM   to  its synonym streams and any indirect streams built on top of them.

    ;; NB: This loses if the indirection is more than one away.

    (LET ((NEWLENGTH (ffetch (STREAM LINELENGTH) of DISPLAYSTREAM)))
         (CL:MAPC #'[LAMBDA (X)
                      (if (AND (BOUNDP (FFETCH (STREAM F1) OF X))
                               (EQ (CL:SYMBOL-VALUE (FFETCH (STREAM F1) OF X))
                                   DISPLAYSTREAM))
                          then (freplace (STREAM LINELENGTH) of X with NEWLENGTH)
                               (CL:MAPC #'[LAMBDA (Y)
                                             (AND (EQ (ffetch (STREAM F2) of Y)
                                                      X)
                                                  (freplace (STREAM LINELENGTH) of Y with NEWLENGTH]
                                      (ffetch (FDEV OPENFILELST) of %%ECHO-STREAM-DEVICE))
                               (CL:MAPC #'[LAMBDA (Y)
                                             (AND (EQ (ffetch (STREAM F2) of Y)
                                                      X)
                                                  (freplace (STREAM LINELENGTH) of Y with NEWLENGTH]
                                      (ffetch (FDEV OPENFILELST) of %%TWO-WAY-STREAM-DEVICE]
                 (ffetch (FDEV OPENFILELST) of %%SYNONYM-STREAM-DEVICE])
```

(\SFFixY

```
  [LAMBDA (DISPLAYDATA CSINFO)                                          (* rmk%: " 4-Apr-85 13:50")

    ;; makes that part of the bitblt table of a display stream which deals with the Y information consistent.  This is called from \BLTCHAR whenever a
    ;; character is being printed and the charset/y-position caches are invalid              ; assumes DISPLAYDATA has already been type checked.
    (PROG ((PBT (ffetch DDPILOTBBT of DISPLAYDATA))
           (Y (\DSPTRANSFORMY (ffetch DDYPOSITION of DISPLAYDATA)
                     DISPLAYDATA))
           TOP CHARTOP BM)
          [SETQ CHARTOP (IPLUS Y (freplace DDCHARSETASCENT of DISPLAYDATA with (ffetch CHARSETASCENT of CSINFO]
          [freplace PBTDEST of PBT with (\ADDBASE (fetch BITMAPBASE of (SETQ BM (ffetch DDDestination of DISPLAYDATA))
                                                                   )
                                                  (ITIMES (ffetch BITMAPRASTERWIDTH of BM)
                                                         (\SFInvert BM (SETQ TOP (IMAX (IMIN (ffetch DDClippingTop
                                                                                                of DISPLAYDATA)
                                                                                             CHARTOP)
                                                                                       0]
          [freplace PBTSOURCE of PBT with (\ADDBASE (ffetch BITMAPBASE of (SETQ BM (ffetch (CHARSETINFO CHARSETBITMAP)
                                                                                       of CSINFO)))
                                                    (ITIMES (ffetch BITMAPRASTERWIDTH of BM)
                                                           (freplace DDCHARHEIGHTDELTA of DISPLAYDATA
                                                              with (IMIN (IMAX (IDIFFERENCE CHARTOP TOP)
                                                                               0)
                                                                         MAX.SMALL.INTEGER]
          (freplace PBTHEIGHT of PBT with (IMAX (IDIFFERENCE TOP (IMAX (IDIFFERENCE Y (freplace DDCHARSETDESCENT
                                                                                        of DISPLAYDATA
                                                                                        with (ffetch CHARSETDESCENT
                                                                                                of CSINFO)))
                                                                       (ffetch DDClippingBottom of DISPLAYDATA)))
                                                0])

)

(DEFINEQ
```

(\SIMPLE.DSPCREATE

```
  [LAMBDA (DESTINATION)                                         ; Edited  8-Jul-2022 19:47 by rmk
                                                                ; Edited  1-Aug-2021 23:41 by rmk:

    ;; Creates a simple stream-of-type-display on the DESTINATION bitmap or display device. To be replaced by \GENERIC.DSPCREATE on
    ;; WINDOW.

    [COND
       ((NULL DESTINATION)
        (SETQ DESTINATION ScreenBitMap))
       (T (\DTEST DESTINATION 'BITMAP]
    (\COMMON.DSPCREATE DESTINATION])
```

(\COMMON.DSPCREATE

```
  [LAMBDA (DESTINATION FDEV IMAGEOPS OLDDSP)                    ; Edited  9-Jul-2022 12:07 by rmk
                                                                ; Edited  1-Aug-2021 23:41 by rmk:

    ;; Common core of \SIMPLE.DSPCREATE on LLDISPLAY and \GENERIC-DSPCREATE on WINDOW.  Parameters cover the small differences of
    ;; the core functionality.

    ;; OLDDSP, if given, must be a displaystream that gets smashed with the new field values, used apparently to recreate an old window on a new
    ;; screen. Presumably is was previously created by a non-OLDDSP call to this function.

    (LET [(DSTRM (IF OLDDSP
                     THEN ;; Preserve everything here, functions below do other updates. Maybe some of this smashing should be
                          ;; uninterruptable?

                          (CL:UNLESS (DISPLAYSTREAMP OLDDSP)
                                 (\ILLEGAL.ARG OLDDSP))
                          (replace (STREAM DEVICE) of OLDSTREAM with (OR FDEV DisplayFDEV))
                          (replace (STREAM IMAGEOPS) of OLDSTREAM with (OR IMAGEOPS \DISPLAYIMAGEOPS))
                          OLDDSP
                   ELSE ;; Maybe should get the STRMBOUTFN from the FDEV?

                        (create STREAM
                               USERCLOSEABLE _ NIL
                               STRMBOUTFN _ (FUNCTION \DSPPRINTCHAR)
                               IMAGEDATA _ (create \DISPLAYDATA)
                               IMAGEOPS _ (OR IMAGEOPS \DISPLAYIMAGEOPS)
                               DEVICE _ (OR FDEV DisplayFDEV)
                               ACCESS _ 'OUTPUT
                               READONLY-EXTERNALFORMAT _ T]     ; initial x and y positions are 0 when the data is created.
         (DSPFONT DEFAULTFONT DSTRM)                            ; dspfont can win since the (default) display imageops are filled
                                                                ; in the stream
         (DSPDESTINATION DESTINATION DSTRM)                     ; dspdestination calls \SFFixFont, which presumes there is a font
                                                                ; present.

         ;; RMK:  I don't know why there were 2 calls to DSPFONT, taking this one out:

         ;; (DSPFONT DEFAULTFONT DSTRM)

         ;; the reference to SCREENWIDTH here is for historic reasons: until 3-feb-86 the default right margin was always SCREENWIDTH.  It
         ;; should be the width of the destination and for any destination larger than the screen this is a serious bug and was fixed.  The MAX of the
         ;; right value and SCREENWIDTH was left in because existing code might be assumine a large right margin for small bitmaps and auto-CR
         ;; in without it.  rrb

         (DSPRIGHTMARGIN (MAX SCREENWIDTH (fetch (BITMAP BITMAPWIDTH) of DESTINATION))
                DSTRM)
         (DSPSOURCETYPE 'INPUT DSTRM)
         (DSPOPERATION 'REPLACE DSTRM)                          ; called to cause the updating of the bitblt table from the fields
                                                                ; initialized earlier.
```

```
          DSTRM])

)

;; MOVD? so we don't trash a later redefinition

(MOVD? '\SIMPLE.DSPCREATE 'DSPCREATE)

(DEFINEQ

(\MEDW.XOFFSET
  [LAMBDA (XOFFSET DISPLAYSTREAM)                                ; Edited 17-Apr-94 23:32 by sybalsky

    ;; Set the X OFFSET for a normal Medley window/display styream.

    ;; coordinate position is stored in 15 bits in the range -2^15 to +2^15.

    (COND
       [DISPLAYSTREAM (PROG ((DD (\GETDISPLAYDATA DISPLAYSTREAM)))
                            (RETURN (PROG1 (fetch DDXOFFSET of DD)
                                           (COND
                                              ((NULL XOFFSET))
                                              ((NUMBERP XOFFSET)
                                               (UNINTERRUPTABLY
                                                   (freplace DDXOFFSET of DD with XOFFSET)
                                                   (\SFFixClippingRegion DD)))
                                              (T (\ILLEGAL.ARG XOFFSET))))]
       (T                                                        ; check done specially for NIL so that it won't default to primary
                                                                 ; output file.
          (\ILLEGAL.ARG DISPLAYSTREAM])

(\MEDW.YOFFSET
  [LAMBDA (YOFFSET DISPLAYSTREAM)                                (* rmk%: " 4-Apr-85 13:43")
    (COND
       [DISPLAYSTREAM (PROG ((DD (\GETDISPLAYDATA DISPLAYSTREAM)))
                            (RETURN (PROG1 (ffetch DDYOFFSET of DD)
                                           (COND
                                              ((NULL YOFFSET))
                                              ((NUMBERP YOFFSET)
                                               (UNINTERRUPTABLY
                                                   (freplace DDYOFFSET of DD with YOFFSET)
                                                   (\SFFixClippingRegion DD)
                                                   (\INVALIDATEDISPLAYCACHE DD)))
                                              (T (\ILLEGAL.ARG YOFFSET))))]
       (T                                                        ; check done specially for NIL so that it won't default to primary
                                                                 ; output file.
          (\ILLEGAL.ARG DISPLAYSTREAM])
)

(DEFINEQ

(\DSPCLIPPINGREGION.DISPLAY
  [LAMBDA (DISPLAYSTREAM REGION)                                 (* rmk%: " 4-Apr-85 13:44")

    ;; sets the clipping region of a display stream.

    (PROG ((DD (\GETDISPLAYDATA DISPLAYSTREAM)))
          (RETURN (PROG1 (ffetch DDClippingRegion of DD)
                         [COND
                            (REGION (OR (type? REGION REGION)
                                        (ERROR REGION " is not a REGION."))
                                   (UNINTERRUPTABLY
                                       (freplace DDClippingRegion of DD with REGION)
                                       (\SFFixClippingRegion DD)
                                       (\INVALIDATEDISPLAYCACHE DD))])])

(\DSPFONT.DISPLAY
  [LAMBDA (DISPLAYSTREAM FONT)                                   ; Edited 11-Nov-87 15:36 by FS

    ;; sets the font that a display stream uses to print characters.  DISPLAYSTREAM is guaranteed to be a stream of type display

    (PROG (XFONT OLDFONT DD)
          (SETQ DD (fetch (STREAM IMAGEDATA) of DISPLAYSTREAM))      ; save old value to return, smash new value and update the
                                                                 ; bitchar portion of the record.
          (RETURN (PROG1 (SETQ OLDFONT (fetch (\DISPLAYDATA DDFONT) of DD))
                         [COND
                            (FONT ;; Either FONT is coerceable to a font, or its a proplist of ways to change the current font (see IRM), otherwise
                                  ;; an error.

                                  (SETQ XFONT (OR (\COERCEFONTDESC FONT DISPLAYSTREAM T)
                                                  (FONTCOPY (ffetch (\DISPLAYDATA DDFONT) of DD)
                                                         (CONS 'NOERROR (CONS T FONT)))
                                                  (ERROR "FONT NOT FOUND OR ILLEGAL FONTCOPY PARAMETER")))
                                                                 ; updating font information is fairly expensive operation.  Don't
                                                                 ; bother unless font has changed.
                                  (OR (EQ XFONT OLDFONT)
                                      (UNINTERRUPTABLY
                                            (freplace (\DISPLAYDATA DDFONT) of DD with XFONT)
                                            (freplace (\DISPLAYDATA DDLINEFEED) of DD with (IMINUS (fetch (FONTDESCRIPTOR
                                                                                                            \SFHeight)
                                                                                                     of XFONT)))
                                                                      ; This will be difference when spacefactor is implemented for the
                                                                      ; display.
                                            (freplace (\DISPLAYDATA DDSPACEWIDTH) of DD with (\FGETCHARWIDTH
                                                                                                XFONT
                                                                                                (CHARCODE SPACE)))
                                            (\SFFixFont DISPLAYSTREAM DD))])])
```

(\DISPLAY.PILOTBITBLT

```
  [LAMBDA (PILOTBBT N)                                               (* kbr%: "13-Jun-85 16:06")
    (\PILOTBITBLT PILOTBBT N])
```

(\DSPLINEFEED.DISPLAY

```
  [LAMBDA (DISPLAYSTREAM DELTAY)                                     (* rmk%: " 2-SEP-83 10:56")

    ;; sets the amount that a line feed increases the y coordinate by.

    (PROG ((DD (fetch IMAGEDATA of DISPLAYSTREAM)))
          (RETURN (PROG1 (ffetch DDLINEFEED of DD)
                      [AND DELTAY (COND
                                     ((NUMBERP DELTAY)
                                      (freplace DDLINEFEED of DD with DELTAY))
                                     (T (\ILLEGAL.ARG DELTAY])])
```

(\DSPLEFTMARGIN.DISPLAY

```
  [LAMBDA (DISPLAYSTREAM XPOSITION)                                  (* rrb " 3-Oct-85 09:28")

    ;; sets the xposition that a carriage return returns to.

    (PROG ((DD (fetch IMAGEDATA of DISPLAYSTREAM)))
          (RETURN (PROG1 (ffetch DDLeftMargin of DD)
                      [AND XPOSITION (COND
                                        ((NUMBERP XPOSITION)
                                         (UNINTERRUPTABLY
                                             (freplace DDLeftMargin of DD with XPOSITION)
                                             (\SFFIXLINELENGTH DISPLAYSTREAM)))
                                        (T (\ILLEGAL.ARG XPOSITION])])
```

(\DSPOPERATION.DISPLAY

```
  [LAMBDA (DISPLAYSTREAM OPERATION)                                  (* rmk%: "12-Sep-84 09:56")

    ;; sets the operation field of a display stream

    (PROG ((DD (\GETDISPLAYDATA DISPLAYSTREAM)))
          (RETURN (PROG1 (fetch DDOPERATION of DD)
                      [COND
                         (OPERATION (OR (FMEMB OPERATION '(PAINT REPLACE INVERT ERASE))
                                        (LISPERROR "ILLEGAL ARG" OPERATION))
                                (UNINTERRUPTABLY
                                    (freplace DDOPERATION of DD with OPERATION)
                                                                     ; update other fields that depend on operation.
                                    (\SETPBTFUNCTION (fetch DDPILOTBBT of DD)
                                           (fetch DDSOURCETYPE of DD)
                                           OPERATION))])])
```

(\DSPRIGHTMARGIN.DISPLAY

```
  [LAMBDA (DISPLAYSTREAM XPOSITION)                                  (* rrb " 3-Oct-85 09:29")

    ;; Sets the right margin that determines when a cr is inserted by print.

    (PROG (OLDRM (DD (fetch IMAGEDATA of DISPLAYSTREAM)))
          (SETQ OLDRM (ffetch DDRightMargin of DD))
          (COND
             ((NULL XPOSITION))
             [(NUMBERP XPOSITION)                                    ; Avoid fixing linelength if right margin hasn't changed.
              (OR (EQUAL XPOSITION OLDRM)
                  (UNINTERRUPTABLY
                      (freplace DDRightMargin of DD with XPOSITION)
                      (\SFFIXLINELENGTH DISPLAYSTREAM))]
             (T (\ILLEGAL.ARG XPOSITION)))
          (RETURN OLDRM])
```

(\DSPXPOSITION.DISPLAY

```
  [LAMBDA (DISPLAYSTREAM XPOSITION)                                  (* rmk%: " 2-SEP-83 10:56")

    ;; coordinate position is stored in 15 bits in the range -2^15 to +2^15.

    (PROG ((DD (fetch IMAGEDATA of DISPLAYSTREAM)))
          (RETURN (PROG1 (fetch DDXPOSITION of DD)
                      (COND
                         ((NULL XPOSITION))
                         ((NUMBERP XPOSITION)
                          (freplace DDXPOSITION of DD with XPOSITION)
```

```
                                                                  ; reset the charposition field so that PRINT etc. won't put out
                                                                  ; eols.
                               (freplace (STREAM CHARPOSITION) of DISPLAYSTREAM with 0))
                              (T (\ILLEGAL.ARG XPOSITION))))])

(\DSPYPOSITION.DISPLAY
  [LAMBDA (DISPLAYSTREAM YPOSITION)                               (* rmk%: " 4-Apr-85 13:45")
    (PROG ((DD (fetch IMAGEDATA of DISPLAYSTREAM)))
          (RETURN (PROG1 (ffetch DDYPOSITION of DD)
                      (COND
                         ((NULL YPOSITION))
                         ((NUMBERP YPOSITION)
                          (UNINTERRUPTABLY
                              (freplace DDYPOSITION of DD with YPOSITION)
                              (\INVALIDATEDISPLAYCACHE DD)))
                         (T (\ILLEGAL.ARG YPOSITION))))])
)

(MOVD? '\ILLEGAL.ARG '\COERCETODS)

(MOVD? 'NILL 'WFROMDS)

(MOVD? 'NILL 'WINDOWP)

(MOVD? 'NILL 'INVERTW)

(RPAQ? PROMPTWINDOW T)

(RPAQ? \WINDOWWORLD NIL)

(RPAQ? \MAINSCREEN NIL)

;; Stub for window package

(RPAQ? \TOPWDS )

(RPAQ? \SCREENBITMAPS )

(MOVD? 'NILL '\TOTOPWDS)

(DECLARE%: DONTCOPY EVAL@COMPILE

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

[PROGN (PUTPROPS \INSURETOPWDS DMACRO [OPENLAMBDA (DS)
                                              (OR (EQ DS \TOPWDS)
                                                  (COND
                                                     ((FMEMB (DSPDESTINATION NIL DS)
                                                             \SCREENBITMAPS)
                                                      (\TOTOPWDS DS])
       (PUTPROPS \INSURETOPWDS MACRO ((DS)                        (* For non-window implementations)
                                          (PROGN)))]

(PUTPROPS .WHILE.TOP.DS. MACRO [(FIRST . REST)
                                      (PROG (DISPINTERRUPT SOFTCURSORUP)

          (* FIRST should be a displaystream and a variable. This macro may also take a soft cursor down, similar to the way
          .WHILE.CURSOR.DOWN. does, but only if FIRST's destination is the same as the soft cursor's destination.
          *)

                                            [COND
                                               (\SOFTCURSORP (SETQ SOFTCURSORUP (AND \SOFTCURSORUPP
                                                                                     (EQ (DSPDESTINATION NIL FIRST)
                                                                                         \CURSORDESTINATION)))
                                                      (COND
                                                         (SOFTCURSORUP (SETQ DISPINTERRUPT (\GETBASE \EM.DISPINTERRUPT
                                                                                                  0))
                                                                (\PUTBASE \EM.DISPINTERRUPT 0 0)
                                                                (\SOFTCURSORDOWN]
                                            (\INSURETOPWDS FIRST)
                                            (PROGN . REST)
                                            (COND
                                               (SOFTCURSORUP (\SOFTCURSORUPCURRENT)
                                                      (\PUTBASE \EM.DISPINTERRUPT 0 DISPINTERRUPT])

(PUTPROPS .WHILE.CURSOR.DOWN. MACRO [(FIRST . REST)
                                           (PROG (DISPINTERRUPT SOFTCURSORUP)

          (* This macro should wrap around any code that draws or bitblts directly from or to a screen bitmap.
          E.g. DRAWGRAYBOX in HLDISPLAY which puts up a shadow box during GETREGION.
          The purpose of this macro is that a soft (e.g. color) cursor's bits not be taken to be screen bits while FIRST & REST are
          done. *)
```

```
                                                    [COND
                                                       (\SOFTCURSORP (SETQ SOFTCURSORUP \SOFTCURSORUPP)
                                                              (COND
                                                                 (SOFTCURSORUP (SETQ DISPINTERRUPT (\GETBASE
                                                                                                     \EM.DISPINTERRUPT
                                                                                                        0))
                                                                        (\PUTBASE \EM.DISPINTERRUPT 0 0)
                                                                        (\SOFTCURSORDOWN]
                                                    (PROGN FIRST . REST)
                                                    (COND
                                                       (SOFTCURSORUP (\SOFTCURSORUPCURRENT)
                                                              (\PUTBASE \EM.DISPINTERRUPT 0 DISPINTERRUPT])
)

(ADDTOVAR GLOBALVARS \TOPWDS)
)
```

;; END EXPORTED DEFINITIONS

;; DisplayStream TTY functions

```
(DEFINEQ

(TTYDISPLAYSTREAM
  [LAMBDA (DISPLAYSTREAM)                                      ; Edited 19-Jan-88 11:45 by jds

    ;; Makes DISPLAYSTREAM be the ttydisplaystream, and return the old value.  Only change it if DISPLAYSTREAM is non-NIL.

    (DECLARE (GLOBALVARS \DEFAULTTTYDISPLAYSTREAM))
    (PROG1 \TERM.OFD                                           ; Return the pre-existing value
        [COND
           (DISPLAYSTREAM

                  ;; Only try setting it if he really passed in a new value.

                  (SETQ DISPLAYSTREAM (\OUTSTREAMARG DISPLAYSTREAM))
                  (OR (DISPLAYSTREAMP DISPLAYSTREAM)
                      (AND (GETD 'TEXTSTREAMP)
                           (TEXTSTREAMP DISPLAYSTREAM))
                      (\ILLEGAL.ARG DISPLAYSTREAM))            ; Better be a display stream!
                  (UNINTERRUPTABLY

                      ;; make sure there's something to do

                      (COND
                         ((NEQ DISPLAYSTREAM \TERM.OFD)

                          ;; First remove the old ttydisplaystream (if any)

                          [COND
                             ((AND \TERM.OFD (NEQ \TERM.OFD \DEFAULTTTYDISPLAYSTREAM))

                              ;; make sure caret is off before changing display streams.

                              (\CHECKCARET)
                              (LET ((WIN (WFROMDS \TERM.OFD T)))
                                   (AND WIN (WINDOWPROP WIN '\LINEBUF.OFD \LINEBUF.OFD]

                          ;; Now install the new ttydisplaystream.

                          ;; if old T was the primary output, change it to the new ttydisplaystream.

                          (COND
                             ((EQ *STANDARD-OUTPUT* \TERM.OFD)
                              (SETQ *STANDARD-OUTPUT* DISPLAYSTREAM)))
                          (SETQ \TERM.OFD DISPLAYSTREAM)

                          ;; save and restore line buffer from the displaystream window if any.

                          (COND
                             ([EQ *STANDARD-INPUT* (PROG1 \LINEBUF.OFD
                                                          [PROG (WIN)
                                                                (SETQ WIN (WFROMDS DISPLAYSTREAM T))
                                                                (SETQ \LINEBUF.OFD
                                                                 (OR [COND
                                                                        (WIN (WINDOWPROP WIN 'PROCESS (THIS.PROCESS
                                                                                                        ))
                                                                   ; For the PROC world to worry about tty moving
                                                                             (WINDOWPROP WIN '\LINEBUF.OFD]
                                                                     (\CREATELINEBUFFER])]
                                                                   ; primary input is line buffer, switch it too.
                              (SETQ *STANDARD-INPUT* \LINEBUF.OFD)))
                          (SETQ TtyDisplayStream DISPLAYSTREAM)  ; just in case, for backward compatibility

                          ))                                   ; change scroll mode of tty stream to scroll.
                      [COND
                         ((FMEMB (IMAGESTREAMTYPE DISPLAYSTREAM)
                                 \DISPLAYSTREAMTYPES)
                          (DSPSCROLL 'ON DISPLAYSTREAM)        ; Reset page characteristics.
                          (PROG (DD)
                                (SETQ DD (fetch (STREAM IMAGEDATA) of DISPLAYSTREAM))
                                (PAGEHEIGHT (IQUOTIENT (IDIFFERENCE (fetch (\DISPLAYDATA DDClippingTop)
                                                                           of DD)
                                                                    (fetch (\DISPLAYDATA DDClippingBottom)
                                                                      of DD))
                                                       (IABS (fetch (\DISPLAYDATA DDLINEFEED) of DD])])])

)

;; FOLLOWING DEFINITIONS EXPORTED


(DEFOPTIMIZER TTYDISPLAYSTREAM (&REST X)
                                        (COND
                                           ((NULL (CAR X))
                                            '\TERM.OFD)
                                           (T 'IGNOREMACRO)))

;; END EXPORTED DEFINITIONS

(DEFINEQ

(DSPSCROLL
  [LAMBDA (SWITCHSETTING DISPLAYSTREAM)                              (* rmk%: "23-AUG-83 13:02")

    ;; sets the SCROLL characteristics of the font in a display stream.  If SWITCHSETTING in ON, when bottom of screen is reached, contents will be
    ;; blted DSPLineFeed bits.

    (PROG ((DD (\GETDISPLAYDATA DISPLAYSTREAM)))
          (RETURN (PROG1 (OR (ffetch DDScroll of DD)
                             'OFF)
                      [AND SWITCHSETTING (freplace DDScroll of DD with (NEQ SWITCHSETTING 'OFF])])


(PAGEHEIGHT
  [LAMBDA (N)                                                        (* rrb "23-JUL-83 15:08")

    ;; sets the page height in lines of the screen.

    (PROG1 \#DISPLAYLINES
        (COND
           ((NUMBERP N)
            (SETQ \#DISPLAYLINES N)
            (SETQ \CURRENTDISPLAYLINE 0))))])

)

(RPAQ? \CURRENTTTYDEVICE 'BCPLDISPLAY)

(DEFINEQ

(\DSPRESET.DISPLAY
  [LAMBDA (DISPLAYSTREAM)                                            ; Edited  8-Dec-93 18:09 by nilsson
    (DECLARE (GLOBALVARS \CURRENTDISPLAYLINE))                       ; resets a display stream
    (LET (CREG FONT FONTASCENT (DD (\DTEST (fetch (STREAM IMAGEDATA) of (SETQ DISPLAYSTREAM (\OUTSTREAMARG
                                                                                                        DISPLAYSTREAM)
                                                                                      ))
                                               '\DISPLAYDATA))
               (WINDOW (WFROMDS DISPLAYSTREAM T)))
         (WXOFFSET (WXOFFSET NIL WINDOW)
                WINDOW)
         (WYOFFSET (WYOFFSET NIL WINDOW)
                WINDOW)
         (SETQ CREG (ffetch (\DISPLAYDATA DDClippingRegion) of DD))
         (SETQ FONT (fetch (\DISPLAYDATA DDFONT) of DD))
         (SETQ FONTASCENT (FONTASCENT FONT))
         (SELECTQ (fetch (FONTDESCRIPTOR ROTATION) of FONT)
              (0 (\DSPXPOSITION.DISPLAY DISPLAYSTREAM (ffetch (\DISPLAYDATA DDLeftMargin) of DD))
                 (\DSPYPOSITION.DISPLAY DISPLAYSTREAM (ADD1 (IDIFFERENCE (fetch (REGION TOP) of CREG)
                                                                   FONTASCENT))))
              (90 (\DSPXPOSITION.DISPLAY DISPLAYSTREAM (IPLUS (fetch (REGION LEFT) of CREG)
                                                             FONTASCENT))
                  (\DSPYPOSITION.DISPLAY DISPLAYSTREAM (fetch (REGION BOTTOM) of CREG)))
              (270 (\DSPXPOSITION.DISPLAY DISPLAYSTREAM (IDIFFERENCE (fetch (REGION RIGHT) of CREG)
                                                                    FONTASCENT))
                   (\DSPYPOSITION.DISPLAY DISPLAYSTREAM (fetch (REGION TOP) of CREG)))
              (ERROR "only supported rotations are 0, 90 and 270"))
         (BITBLT NIL NIL NIL DISPLAYSTREAM (fetch (REGION LEFT) of CREG)
                (fetch (REGION BOTTOM) of CREG)
                (fetch (REGION WIDTH) of CREG)
                (fetch (REGION HEIGHT) of CREG)
                'TEXTURE
                'REPLACE
                (ffetch (\DISPLAYDATA DDTexture) of DD))

         ;; if this display stream is the tty display stream of a process, reset the # of lines in that process.

         (PROG ((X (WFROMDS DISPLAYSTREAM T)))
               (COND
                  ((AND X (SETQ X (WINDOWPROP X 'PROCESS))
                        (EQ (PROCESS.TTY X)
                            DISPLAYSTREAM))
                   (PROCESS.EVAL X '(SETQ \CURRENTDISPLAYLINE 0])
)

(RPAQ? *DRIBBLE-OUTPUT* NIL)

(DEFMACRO \MAYBE-DRIBBLE-CHAR (DISPLAY-STREAM CHARCODE)
   "if we are dribbling, then dribble this character"
   ;; *DRIBBLE-OUTPUT* is a per-process special.

   ;; Only dribble if *DRIBBLE-OUTPUT* is not NIL, and IS a stream; the NIL check is for speed, since the STREAMP is something like 30 of the time
   ;; spent printing to the exec window!!

   `(AND *DRIBBLE-OUTPUT* (STREAMP *DRIBBLE-OUTPUT*)
         (EQ ,DISPLAY-STREAM (TTYDISPLAYSTREAM))
         (\OUTCHAR *DRIBBLE-OUTPUT* ,CHARCODE)))

(DEFINEQ

(\DSPPRINTCHAR
  [LAMBDA (STREAM CHARCODE)                                     ; Edited 10-May-88 23:40 by MASINTER

    ;; Displays a character on a display stream.  Handles dribbling, too.

    (PROG ((DD (ffetch (STREAM IMAGEDATA) of STREAM)))
          (\CHECKCARET STREAM)
          (\MAYBE-DRIBBLE-CHAR STREAM CHARCODE)                 ; if dribbling, dribble.
          (SELECTC (ffetch (TERMCODE CCECHO) of (\SYNCODE \PRIMTERMSA CHARCODE))
              (REAL.CCE ;; All fat characters are defined as REAL according to \SYNCODE, so we don't have worry about any of the special
                        ;; cases

                        [COND
                           ((IGREATERP CHARCODE (CONSTANT (IMAX (CHARCODE EOL)
                                                                (CHARCODE CR)
                                                                (CHARCODE LF)
                                                                ERASECHARCODE)))
                                                                ; This is for sure a printing character; take the fast way out.
                            (\BLTCHAR CHARCODE STREAM DD)
                            (add (ffetch (STREAM CHARPOSITION) of STREAM)
                                 1))
                           (T                                   ; Take the slow check.
                              (SELECTC CHARCODE
                                  ((CHARCODE (EOL CR LF))
                                       (\DSPPRINTCR/LF CHARCODE STREAM)
                                       (freplace (STREAM CHARPOSITION) of STREAM with 0))
                                  (ERASECHARCODE
                                       (DSPBACKUP (CHARWIDTH (CHARCODE A)
                                                         STREAM)
                                              STREAM)           ; line buffering routines have already taken care of backing up
                                                                ; the position
                                       0)
                                  (PROGN (\BLTCHAR CHARCODE STREAM DD)
                                         (add (ffetch (STREAM CHARPOSITION) of STREAM)
                                              1])
              (INDICATE.CCE                                     ; Make sure that all the chars in the indicate-string fit on the line
                                                                ; or wrap-around together.
                             (PROG (STR)
                                   (SETQ STR (\INDICATESTRING CHARCODE))
                                                                ; This isn't right for rotated fonts.  But then there should probably
                                                                ; be a separate rotated outcharfn
                                   [COND
                                      ((IGREATERP (\STRINGWIDTH.DISPLAY STREAM STR)
                                               (IDIFFERENCE (ffetch (\DISPLAYDATA DDRightMargin) of DD)
                                                      (ffetch (\DISPLAYDATA DDXPOSITION) of DD)))
                                       (\DSPPRINTCR/LF (CHARCODE EOL)
                                              STREAM)
                                       (freplace (STREAM CHARPOSITION) of STREAM with (NCHARS STR)))
                                      (T (add (ffetch (STREAM CHARPOSITION) of STREAM)
                                              (NCHARS STR]
                                   (for I from 1 do (\BLTCHAR (OR (NTHCHARCODE STR I)
                                                                  (RETURN))
                                                           STREAM DD))))
              (SIMULATE.CCE (SELCHARQ CHARCODE
                                 ((EOL CR LF)
                                      (\DSPPRINTCR/LF CHARCODE STREAM)
                                      (freplace (STREAM CHARPOSITION) of STREAM with 0))
                                 (ESCAPE (\BLTCHAR (CHARCODE $)
                                                STREAM DD)
                                         (add (ffetch (STREAM CHARPOSITION) of STREAM)
                                              1))
                                 (BELL                          ; make switching of bits uninterruptable but allow interrupts
                                                                ; between flashes.
                                       (SELECTC \MACHINETYPE
                                           ((LIST \DANDELION \DAYBREAK \MAIKO)
                                                [PLAYTUNE '((880 . 2500])
                                           (FLASHWINDOW (WFROMDS STREAM))))
                                 (TAB (PROG (TABWIDTH (SPACEWIDTH (CHARWIDTH (CHARCODE SPACE)
                                                                         STREAM)))
                                            (SETQ TABWIDTH (UNFOLD SPACEWIDTH 8))
```

```
                                             (COND
                                                ((IGREATERP (\DISPLAYSTREAMINCRXPOSITION
                                                             (SETQ TABWIDTH
                                                              (IDIFFERENCE TABWIDTH
                                                                     (MOD (IDIFFERENCE (ffetch (\DISPLAYDATA
                                                                                                DDXPOSITION)
                                                                                          of DD)
                                                                                 (ffetch (\DISPLAYDATA
                                                                                                DDLeftMargin)
                                                                                   of DD))
                                                                          TABWIDTH)))
                                                             DD)
                                                        (ffetch (\DISPLAYDATA DDRightMargin) of DD))
                                                                        ; tab was past rightmargin, force cr.
                                                 (\DSPPRINTCR/LF (CHARCODE EOL)
                                                        STREAM)))
                                                                        ; return the number of spaces taken.
                                             (add (ffetch (STREAM CHARPOSITION) of STREAM)
                                                  (IQUOTIENT TABWIDTH SPACEWIDTH))))
                                    (PROGN                              ; this case was copied from \DSCCOUT.
                                          (\BLTCHAR CHARCODE STREAM DD)
                                          (add (ffetch (STREAM CHARPOSITION) of STREAM)
                                               1))))
                  (IGNORE.CCE)
                  (SHOULDNT])
```

(**\DSPPRINTCR/LF**

```
  [LAMBDA (CHARCODE DISPLAY-STREAM)                                     ; Edited 16-Jan-87 17:14 by hdj
```

;; CHARCODE is EOL, CR, or LF. Assumes that DISPLAY-STREAM has been type-checked by \DSPPRINTCHAR.

;; [Changed to call DSPXPOSITION and DSPYPOSITION instead of \DSPxPOSITION.DISPLAY so that it could be used in the hardcopy display
;; stream case as well. Could go back to other method if efficiency becomes an issue.]

```
    (COND
       ((EQ DISPLAY-STREAM (TTYDISPLAYSTREAM))
        (\STOPSCROLL?)                                                  ; \STOPSCROLL may have turned on the caret.
        (\CHECKCARET DISPLAY-STREAM)))
    (PROG (BTM AMOUNT/BELOW Y ROTATION FONT (DD (fetch (STREAM IMAGEDATA) of DISPLAY-STREAM)))
          (COND
             ((AND (fetch (\DISPLAYDATA DDSlowPrintingCase) of DD)
                   (NEQ (SETQ ROTATION (fetch (FONTDESCRIPTOR ROTATION) of (fetch (\DISPLAYDATA DDFONT)
                                                                                of DD)))
                        0))
              (PROG ((CLIPREG (ffetch (\DISPLAYDATA DDClippingRegion) of DD))
                     X)
                    (COND
                       ((EQ CHARCODE (CHARCODE EOL))                    ; on LF, no change in X
                        (COND
                           ((SETQ Y (fetch (\DISPLAYDATA DDEOLFN) of DD))
                                                                        ; call the eol function for ds.
                            (APPLY* Y DISPLAY-STREAM)))
                        (DSPYPOSITION (SELECTQ ROTATION
                                          (90 (fetch (REGION BOTTOM) of CLIPREG))
                                          (270 (fetch (REGION TOP) of CLIPREG))
                                          (ERROR "Only rotations supported are 0, 90 and 270"))
                               DISPLAY-STREAM)))
                    [SETQ X (IPLUS (fetch (\DISPLAYDATA DDXPOSITION) of DD)
                                   (SELECTQ ROTATION
                                       (90 (IMINUS (ffetch (\DISPLAYDATA DDLINEFEED) of DD)))
                                       (270 (ffetch (\DISPLAYDATA DDLINEFEED) of DD))
                                       (ERROR "Only rotations supported are 0, 90 and 270"]
                    [COND
                       ((AND (fetch (\DISPLAYDATA DDScroll) of DD)
                             (SELECTQ ROTATION
                                 (90 (IGREATERP [SETQ AMOUNT/BELOW (IDIFFERENCE
                                                                    (\DSPTRANSFORMX X DD)
                                                                    (IDIFFERENCE (fetch (\DISPLAYDATA
                                                                                         DDClippingRight)
                                                                                    of DD)
                                                                           (fetch (FONTDESCRIPTOR \SFDescent)
                                                                              of (fetch (\DISPLAYDATA DDFONT)
                                                                                    of DD]
                                            0))
                                 (270 (IGREATERP (SETQ AMOUNT/BELOW (IDIFFERENCE
                                                                     (IPLUS (fetch (\DISPLAYDATA DDClippingLeft)
                                                                                of DD)
                                                                            (fetch (FONTDESCRIPTOR \SFDescent)
                                                                               of (fetch (\DISPLAYDATA DDFONT)
                                                                                     of DD)))
                                                                     (\DSPTRANSFORMX X DD)))
                                             0))
                                 (SHOULDNT)))
```

;; automatically scroll up enough to make the entire next character visible. Descent check is so that the bottoms of
;; characters will be printed also.

```
                        [PROG (LFT WDTH BKGRND DBITMAP HGHT KEPTWIDTH)
                              (SETQ LFT (fetch (\DISPLAYDATA DDClippingLeft) of DD))
                              (SETQ DBITMAP (fetch (\DISPLAYDATA DDDestination) of DD))
                              (SETQ BTM (fetch (\DISPLAYDATA DDClippingBottom) of DD))
                              (SETQ HGHT (IDIFFERENCE (ffetch (\DISPLAYDATA DDClippingTop) of DD)
                                                BTM))
                              (SETQ WDTH (IDIFFERENCE (fetch (\DISPLAYDATA DDClippingRight) of DD)
                                                LFT))
                              (SETQ BKGRND (ffetch (\DISPLAYDATA DDTexture) of DD))
                              (.WHILE.TOP.DS. DISPLAY-STREAM
                                     (COND
                                        ((IGREATERP AMOUNT/BELOW WDTH)
                                                               ; scrolling more than the window size, use different method.
                                                               ; clear the window with background.
                                         (BITBLT NIL 0 0 DBITMAP LFT BTM WDTH HGHT 'TEXTURE 'REPLACE BKGRND))
                                        ((EQ ROTATION 90)
                                         (BITBLT DBITMAP (IPLUS LFT AMOUNT/BELOW)
                                                BTM DBITMAP LFT BTM (SETQ KEPTWIDTH (IDIFFERENCE WDTH
                                                                                           AMOUNT/BELOW))
                                                HGHT
                                                'INPUT
                                                'REPLACE)
                                         (BITBLT NIL 0 0 DBITMAP (IPLUS LFT KEPTWIDTH)
                                                BTM AMOUNT/BELOW HGHT 'TEXTURE 'REPLACE BKGRND))
                                        (T (BITBLT DBITMAP LFT BTM DBITMAP (IPLUS LFT AMOUNT/BELOW)
                                                  BTM
                                                  (IDIFFERENCE WDTH AMOUNT/BELOW)
                                                  HGHT
                                                  'INPUT
                                                  'REPLACE)
                                           (BITBLT NIL 0 0 DBITMAP LFT BTM AMOUNT/BELOW HGHT 'TEXTURE
                                                  'REPLACE BKGRND]
                        (SETQ X (SELECTQ ROTATION
                                      (90 (IDIFFERENCE X AMOUNT/BELOW))
                                      (IPLUS X AMOUNT/BELOW]
                    (DSPXPOSITION X DISPLAY-STREAM)))
             (T (COND
                   ((EQ CHARCODE (CHARCODE EOL))                 ; on LF, no change in X
                    (COND
                       ((SETQ Y (fetch (\DISPLAYDATA DDEOLFN) of DD))
                                                               ; call the eol function for ds.
                        (APPLY* Y DISPLAY-STREAM)))
                    (DSPXPOSITION (ffetch (\DISPLAYDATA DDLeftMargin) of DD)
                           DISPLAY-STREAM)))
                (SETQ Y (IPLUS (ffetch (\DISPLAYDATA DDYPOSITION) of DD)
                               (ffetch (\DISPLAYDATA DDLINEFEED) of DD)))
                [COND
                   ((AND (fetch (\DISPLAYDATA DDScroll) of DD)
                         (IGREATERP (SETQ AMOUNT/BELOW (IDIFFERENCE (IPLUS (SETQ BTM (fetch (\DISPLAYDATA
                                                                                            DDClippingBottom
                                                                                                  )
                                                                                        of DD))
                                                                              (fetch (FONTDESCRIPTOR \SFDescent)
                                                                              of (fetch (\DISPLAYDATA DDFONT)
                                                                                    of DD)))
                                                                    (\DSPTRANSFORMY Y DD)))
                                0))

                    ;; automatically scroll up enough to make the entire next character visible.  Descent check is so that the bottoms of characters
                    ;; will be printed also.

                    [PROG (LFT WDTH BKGRND DBITMAP HGHT)
                          (SETQ LFT (fetch (\DISPLAYDATA DDClippingLeft) of DD))
                          (SETQ DBITMAP (fetch (\DISPLAYDATA DDDestination) of DD))
                          (SETQ HGHT (IDIFFERENCE (ffetch (\DISPLAYDATA DDClippingTop) of DD)
                                            BTM))
                          (SETQ WDTH (IDIFFERENCE (fetch (\DISPLAYDATA DDClippingRight) of DD)
                                            LFT))
                          (SETQ BKGRND (ffetch (\DISPLAYDATA DDTexture) of DD))
                          (.WHILE.TOP.DS. DISPLAY-STREAM (COND
                                                            ((IGREATERP AMOUNT/BELOW HGHT)
                                                            ; scrolling more than the window size, use different method.
                                                            ; clear the window with background.
                                                             (BITBLT NIL 0 0 DBITMAP LFT BTM WDTH HGHT
                                                                    'TEXTURE
                                                                    'REPLACE BKGRND))
                                                            (T (BITBLT DBITMAP LFT BTM DBITMAP LFT
                                                                      (IPLUS BTM AMOUNT/BELOW)
                                                                      WDTH
                                                                      (IDIFFERENCE HGHT AMOUNT/BELOW)
                                                                      'INPUT
                                                                      'REPLACE)
                                                               (BITBLT NIL 0 0 DBITMAP LFT BTM WDTH AMOUNT/BELOW
                                                                      'TEXTURE
                                                                      'REPLACE BKGRND]
                    (SETQ Y (IPLUS Y AMOUNT/BELOW]
                (DSPYPOSITION Y DISPLAY-STREAM])
   )
```

```
(DEFINEQ

(\TTYBACKGROUND
  [LAMBDA NIL                                              (* lmm "30-Dec-85 20:22")

    ;; called each time through a tty keyboard wait loop.  First executes the TTYBACKGROUNDFNS which do things like flashing the caret (and
    ;; SAVEVM) and then allows other background things to run (including other processes.)

    [COND
       ((EQ (fetch KEYBOARDSTREAM of \LINEBUF.OFD)
            \KEYBOARD.STREAM)
        (OR (TTY.PROCESSP)
            (WAIT.FOR.TTY))
        (for X in TTYBACKGROUNDFNS do (APPLY* X]
    (\BACKGROUND])
)

(DEFINEQ

(DSPBACKUP
  [LAMBDA (WIDTH DISPLAYSTREAM)                            (* "Pavel" "25-Apr-86 16:37")
    (COND
       [[OR (DISPLAYSTREAMP DISPLAYSTREAM)
            (DISPLAYSTREAMP (SETQ DISPLAYSTREAM (GETSTREAM DISPLAYSTREAM 'OUTPUT]
        (PROG (FONT ROTATION BLTWIDTH XPOS (DD (\GETDISPLAYDATA DISPLAYSTREAM DISPLAYSTREAM)))
              [SETQ BLTWIDTH (IMIN WIDTH (IDIFFERENCE (SETQ XPOS (fetch DDXPOSITION of DD))
                                                (ffetch DDLeftMargin of DD]
              (SETQ FONT (fetch DDFONT of DD))
              (SETQ ROTATION (COND
                                ((fetch DDSlowPrintingCase of DD)
                                 (fetch (FONTDESCRIPTOR ROTATION) of FONT))
                                (T 0)))
              (RETURN (COND
                         ((IGREATERP BLTWIDTH 0)
                          (\CHECKCARET DISPLAYSTREAM)
                          [COND
                             ((EQ ROTATION 0)                ; uses DSPXPOSITION so that it works on both display streams
                                                             ; and hardcopy display streams.
                              (DSPXPOSITION (IDIFFERENCE XPOS BLTWIDTH)
                                     DISPLAYSTREAM)
                              (BITBLT NIL 0 0 DISPLAYSTREAM (fetch DDXPOSITION of DD)
                                     (IDIFFERENCE (ffetch DDYPOSITION of DD)
                                            (FONTDESCENT FONT))
                                     BLTWIDTH
                                     (FONTHEIGHT FONT)
                                     'TEXTURE
                                     'REPLACE))
                             ((EQ ROTATION 90)
                              (BITBLT NIL 0 0 DISPLAYSTREAM (IDIFFERENCE (fetch DDXPOSITION of DD)
                                                                   (FONTASCENT FONT))
                                     (add (fetch DDYPOSITION of DD)
                                          (IMINUS BLTWIDTH))
                                     (FONTHEIGHT FONT)
                                     BLTWIDTH
                                     'TEXTURE
                                     'REPLACE))
                             ((EQ ROTATION 270)
                              (BITBLT NIL 0 0 DISPLAYSTREAM (IDIFFERENCE (fetch DDXPOSITION of DD)
                                                                   (FONTDESCENT FONT))
                                     (add (fetch DDYPOSITION of DD)
                                          BLTWIDTH)
                                     (FONTHEIGHT FONT)
                                     BLTWIDTH
                                     'TEXTURE
                                     'REPLACE]
                          T]
       (T (FRPTQ WIDTH (PROGN (BOUT DISPLAYSTREAM (CHARCODE BS))
                              (BOUT DISPLAYSTREAM (CHARCODE SPACE))
                              (BOUT DISPLAYSTREAM (CHARCODE BS])
)

(RPAQ? \CARET.UP )

(DECLARE%: DONTEVAL@LOAD DOCOPY

(RPAQQ BELLCNT 2)

(RPAQQ BELLRATE 60)

(RPAQQ \DisplayStoppedForLogout NIL)

(RPAQQ TtyDisplayStream NIL)
)

(DEFINEQ
(COLORDISPLAYP
  [LAMBDA NIL                                                   (* gbn%: "26-Jan-86 16:16")
                                                                ; is the color display on?
    (NOT (NULL ColorScreenBitMap])
)

(DEFINEQ

(DISPLAYBEFOREEXIT
  [LAMBDA (EXITFN)                                              ; Edited 16-Nov-93 16:22 by nilsson
    (COND
       ((DISPLAYSTARTEDP)

        ;; save cursor and background border so that they can be restored by DISPLAYAFTERENTRY when this sysout is restarted.

        (SETQ \DisplayStoppedForLogout (CONS (CURSOR)
                                             (CHANGEBACKGROUNDBORDER)))
        (SELECTQ EXITFN
            (LOGOUT                                             ; Shut off display during logout
                    (SHOWDISPLAY))
            (MAKESYS                                            ; on MAKESYS, clear screen
                     (DSPRESET (TTYDISPLAYSTREAM))
                     (CLRPROMPT))
            (SYSOUT NIL)
            (SHOULDNT])

(DISPLAYAFTERENTRY
  [LAMBDA (ENTRYFN)                                             ; Edited 29-Jun-88 14:57 by drc:

    ;; set address of Cursor bitmap every time because it changes from machine to machine and StartDisplay is a convenient place to reset it.

    (replace BITMAPBASE of CursorBitMap with \EM.CURSORBITMAP)
    (COND
       (\DisplayStoppedForLogout (\STARTDISPLAY)
              (VIDEOCOLOR \VideoColor)                          ; restore videocolor
                                                                ; restore the cursor.
              (CURSOR (CAR \DisplayStoppedForLogout))           ; restore the display border.  Only does anything on a
                                                                ; DANDELION
              (CHANGEBACKGROUNDBORDER (CDR \DisplayStoppedForLogout))
              (SETQ \DisplayStoppedForLogout NIL)))             ; reset the time that the caret will flash.
    (COND
       ((GETD 'CARETRATE)

        ;; the caret rate has some global state which depends on the machine dependent clock.  This resets the internal state

        (CARETRATE (CARETRATE])
)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS BELLCNT BELLRATE TTYBACKGROUNDFNS \DisplayStoppedForLogout \CARET.UP)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \CHECKCARET MACRO ((X)
                             (AND \CARET.UP (\CARET.DOWN X))))
)

;; END EXPORTED DEFINITIONS

;; transformation related functions.

(DEFINEQ

(\DSPCLIPTRANSFORMX
  [LAMBDA (X DD)                                                (* rmk%: "23-AUG-83 15:03")

    ;; returns the transformed coordinate value of X in the system of the destination.  It also clips according to the clipping region and returns NIL if it
    ;; falls outside.

    (PROG ((TX (\DSPTRANSFORMX X DD)))
          (RETURN (AND (NOT (IGREATERP (fetch DDClippingLeft of DD)
                                       TX))
                       (IGREATERP (fetch DDClippingRight of DD)
                              TX)
                       TX])

(\DSPCLIPTRANSFORMY
  [LAMBDA (Y DD)                                                (* rmk%: "23-AUG-83 15:09")

    ;; returns the transformed coordinate value of Y in the system of the destination.  It also clips according to the clipping region and returns NIL if it
    ;; falls outside.

    (PROG ((TY (\DSPTRANSFORMY Y DD)))                          ; ClippingTop points past the top edge.
          (RETURN (AND (NOT (IGREATERP (fetch DDClippingBottom of DD)
                              TY))
                       (IGREATERP (fetch DDClippingTop of DD)
                              TY)
                       TY])

(\DSPTRANSFORMREGION
  [LAMBDA (REGION DS)                                    (* rrb " 3-DEC-80 18:11")
    ;; transforms a region into the destination coordinates of the display stream.
    (create REGION
           LEFT _ (\DSPTRANSFORMX (fetch LEFT of REGION)
                         DS)
           BOTTOM _ (\DSPTRANSFORMY (fetch BOTTOM of REGION)
                           DS)
           WIDTH _ (fetch WIDTH of REGION)
           HEIGHT _ (fetch HEIGHT of REGION])

(\DSPUNTRANSFORMY
  [LAMBDA (Y DD)                                         (* rmk%: "23-AUG-83 14:34")
    ;; transforms a y coordinate from destination coords into the display streams
    (IDIFFERENCE Y (fetch DDYOFFSET of DD])

(\DSPUNTRANSFORMX
  [LAMBDA (X DD)                                         (* rmk%: "23-AUG-83 14:25")
    ;; transforms a x coordinate from destination coords into the display streams
    (IDIFFERENCE X (fetch DDXOFFSET of DD])

(\OFFSETCLIPPINGREGION
  [LAMBDA (DD OLDREGION)                                 (* bvm%: "14-Feb-85 00:45")
    ;; calculates the clipping region from the displaydata of a display stream in destination coordinates.  if OLDREGION is given, it is reused.
    (PROG ((CREG (fetch DDClippingRegion of DD)))
          (RETURN (COND
                     (OLDREGION (replace LEFT of OLDREGION with (\DSPTRANSFORMX (fetch LEFT of CREG)
                                                                       DD))
                            (replace BOTTOM of OLDREGION with (\DSPTRANSFORMY (fetch BOTTOM of CREG)
                                                                     DD))
                            (replace WIDTH of OLDREGION with (fetch WIDTH of CREG))
                            (replace HEIGHT of OLDREGION with (fetch HEIGHT of CREG))
                            OLDREGION)
                     ((AND (EQ (fetch DDXOFFSET of DD)
                               0)
                           (EQ (fetch DDYOFFSET of DD)
                               0))                       ; special case of no offset to avoid storage creation.
                      CREG)
                     (T (create REGION
                               LEFT _ (\DSPTRANSFORMX (fetch LEFT of CREG)
                                             DD)
                               BOTTOM _ (\DSPTRANSFORMY (fetch BOTTOM of CREG)
                                               DD)
                               WIDTH _ (fetch WIDTH of CREG)
                               HEIGHT _ (fetch HEIGHT of CREG])
)

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(PUTPROPS \DSPTRANSFORMX MACRO ((X DD)                   (* transforms an x coordinate into the destination coordinate.)
                                (IPLUS X (fetch (\DISPLAYDATA DDXOFFSET) of DD))))

(PUTPROPS \DSPTRANSFORMY MACRO ((Y DD)                   (* transforms an y coordinate into the destination coordinate.)
                                (IPLUS Y (fetch (\DISPLAYDATA DDYOFFSET) of DD))))

(PUTPROPS \OFFSETBOTTOM MACRO ((X)                       (* gives the destination coordinate address of the origin.)
                               (fetch (\DISPLAYDATA DDYOFFSET) of X)))

(PUTPROPS \OFFSETLEFT MACRO ((DD)                        (* returns the x origin of display data destination coordinates.)
                             (fetch (\DISPLAYDATA DDXOFFSET) of DD)))
)
)

;; END EXPORTED DEFINITIONS

;; screen related functions

(DEFINEQ
(UPDATESCREENDIMENSIONS
  [LAMBDA NIL                                                  ; Edited 23-Apr-88 23:32 by MASINTER

;;; Sets SCREENWIDTH and SCREENHEIGHT according to machine

    (SELECTC \MACHINETYPE
         ((LIST \DOLPHIN \DORADO \DANDELION)
              (SETQ SCREENWIDTH 1024)
              (SETQ SCREENHEIGHT 808))
         (\DAYBREAK (SETQ SCREENWIDTH (\DoveDisplay.ScreenWidth))
                (SETQ SCREENHEIGHT (\DoveDisplay.ScreenHeight)))
         (\MAIKO (SETQ SCREENWIDTH (SUBCALL DSP-SCREENWIDTH))
                (SETQ SCREENHEIGHT (SUBCALL DSP-SCREENHEIGHT)))
         (SHOULDNT])

(\CreateScreenBitMap
  [LAMBDA (WIDTH HEIGHT)                                       (* bvm%: "10-Aug-85 23:24")
    (DECLARE (GLOBALVARS \MaxScreenPage))

    ;; creates and locks the pages for the display bit map.  Returns a BITMAP descriptor for it.  Uses the first words of the segment \DISPLAYREGION.

    (LET ((RASTERWIDTH (FOLDHI WIDTH BITSPERWORD))
          MAXPAGE#)                                            ; the display microcode needs to have the display fall on
                                                               ; \DisplayWordAlign word boundaries.
         (COND
            ((IGREATERP (SETQ MAXPAGE# (SUB1 (FOLDHI (ITIMES RASTERWIDTH HEIGHT)
                                                     WORDSPERPAGE)))
                    \MaxScreenPage)

             ;; new screen size is larger, allocate more pages.  All pages are locked.  NOERROR is true in \NEWPAGE call in case pages are
             ;; already there, e.g. DLBOOT allocated them.

             (for I from (ADD1 \MaxScreenPage) to MAXPAGE# do (\NEWPAGE (\ADDBASE \DISPLAYREGION (UNFOLD I
                                                                                                    WORDSPERPAGE
                                                                                                     ))
                                                                      T T))
             (SETQ \MaxScreenPage MAXPAGE#)))
         (COND
            ((BITMAPP ScreenBitMap)                            ; reuse the same BITMAP ptr so that it will stay EQ to the one in
                                                               ; user datastructures.
             (replace BITMAPBASE of ScreenBitMap with \DISPLAYREGION)
             (replace BITMAPWIDTH of ScreenBitMap with WIDTH)
             (replace BITMAPRASTERWIDTH of ScreenBitMap with RASTERWIDTH)
             (replace BITMAPHEIGHT of ScreenBitMap with HEIGHT)
             ScreenBitMap)
            (T (create BITMAP
                      BITMAPBASE _ \DISPLAYREGION
                      BITMAPRASTERWIDTH _ RASTERWIDTH
                      BITMAPWIDTH _ WIDTH
                      BITMAPHEIGHT _ HEIGHT])
)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(UPDATESCREENDIMENSIONS)

(RPAQ? SCREENHEIGHT 808)

(RPAQ? SCREENWIDTH 1024)

(RPAQ? \OLDSCREENHEIGHT 808)

(RPAQ? \OLDSCREENWIDTH 1024)

(RPAQ? \MaxScreenPage -1)

(RPAQ? ScreenBitMap (\CreateScreenBitMap SCREENWIDTH SCREENHEIGHT))

(RPAQ? ColorScreenBitMap NIL)
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \OLDSCREENHEIGHT \OLDSCREENWIDTH \MaxScreenPage ScreenBitMap)
)

(DECLARE%: DONTEVAL@LOAD DOCOPY

(CURSOR.INIT)
)

;; initialization

(RPAQ? \DISPLAYINFOALIST )

(DEFINEQ
(\CoerceToDisplayDevice
  [LAMBDA (NameOrDevice)                                         (* hdj " 8-Mar-85 10:29")
    (DECLARE (GLOBALVARS LastCreatedDisplayDevice))
    (LET ((DEV (OR NameOrDevice LastCreatedDisplayDevice)))
         (COND
            ((type? FDEV DEV)
             DEV)
            (T (OR (\GETDEVICEFROMNAME DEV T T)
                   (ERROR "No color drivers have been loaded"])

(\CREATEDISPLAY
  [LAMBDA (DISPLAYNAME)                                          (* kbr%: " 1-Jul-85 15:23")

;;; create a new display device.  Mainly used by device-independent color code

    (PROG (FDEV)
          [SETQ FDEV (create FDEV
                                 DEVICENAME _ DISPLAYNAME
                                 RESETABLE _ NIL
                                 RANDOMACCESSP _ NIL
                                 PAGEMAPPED _ NIL
                                 CLOSEFILE _ (FUNCTION NILL)
                                 DELETEFILE _ (FUNCTION NILL)
                                 GETFILEINFO _ (FUNCTION NILL)
                                 OPENFILE _ [FUNCTION (LAMBDA (NAME ACCESS RECOG OTHERINFO FDEV)
                                                         NAME]
                                 READPAGES _ (FUNCTION \ILLEGAL.DEVICEOP)
                                 SETFILEINFO _ (FUNCTION NILL)
                                 GENERATEFILES _ (FUNCTION \GENERATENOFILES)
                                 TRUNCATEFILE _ (FUNCTION NILL)
                                 WRITEPAGES _ (FUNCTION \ILLEGAL.DEVICEOP)
                                 GETFILENAME _ [FUNCTION (LAMBDA (NAME RECOG FDEV)
                                                            NAME]
                                 REOPENFILE _ [FUNCTION (LAMBDA (NAME)
                                                           NAME]
                                 EVENTFN _ (FUNCTION NILL)
                                 DIRECTORYNAMEP _ (FUNCTION NILL)
                                 HOSTNAMEP _ (FUNCTION NILL)
                                 BIN _ (FUNCTION \ILLEGAL.DEVICEOP)
                                 BOUT _ (FUNCTION \DSPPRINTCHAR)
                                 PEEKBIN _ (FUNCTION \ILLEGAL.DEVICEOP)
                                 BACKFILEPTR _ (FUNCTION \PAGEDBACKFILEPTR)
                                 BLOCKIN _ (FUNCTION \ILLEGAL.DEVICEOP)
                                 BLOCKOUT _ (FUNCTION \NONPAGEDBOUTS)
                                 DEVICEINFO _ (create DISPLAYSTATE
                                                     ONOFF _ 'OFF]
          (\DEFINEDEVICE DISPLAYNAME FDEV)
          (RETURN FDEV])

(DISPLAYSTREAMINIT
  [LAMBDA (N)                                                    (* kbr%: "24-Feb-86 12:53")
    (DECLARE (GLOBALVARS \LastTTYLines \TopLevelTtyWindow))

    ;; starts display and sets N lines for tty at top

    (\STARTDISPLAY)
    (SETQ TtyDisplayStream (DSPCREATE))
    (PROG (TTYHEIGHT TTYFONTHEIGHT (TTYFONT (DSPFONT NIL TtyDisplayStream)))
          (SETQ TTYFONTHEIGHT (FONTHEIGHT TTYFONT))
          (DSPDESTINATION ScreenBitMap TtyDisplayStream)

     ;; this is done here so that processes that are created before window world is turned on have an acceptable binding for their tty.

          (TERMINAL-OUTPUT (SETQ \TopLevelTtyWindow (SETQ \DEFAULTTTYDISPLAYSTREAM TtyDisplayStream)))
          (RETURN (PROG1 \LastTTYLines
                      (SETQ TTYHEIGHT (ITIMES (COND
                                                 [(NUMBERP N)
                                                  (SETQ \LastTTYLines (COND
                                                                         ((IGREATERP (ITIMES N TTYFONTHEIGHT)
                                                                                 SCREENHEIGHT)
                                                                          ; too many lines, reduce to fit leaving two lines bottom margin.
                                                                          (IDIFFERENCE (IQUOTIENT SCREENHEIGHT
                                                                                              TTYFONTHEIGHT)
                                                                                 2))
                                                                         (T N]
                                                 (T \LastTTYLines))
                                             TTYFONTHEIGHT))      ; put TTY region on top
                      (DSPYOFFSET (IDIFFERENCE SCREENHEIGHT TTYHEIGHT)
                             TtyDisplayStream)
                      (DSPYPOSITION (FONTDESCENT TTYFONT)
                             TtyDisplayStream)
                      (DSPXOFFSET 0 TtyDisplayStream)
                      (DSPCLIPPINGREGION (create REGION
                                                LEFT _ 0
                                                BOTTOM _ 0
                                                WIDTH _ SCREENWIDTH
                                                HEIGHT _ TTYHEIGHT)
```

```
                              TtyDisplayStream)
                        ;; called after clipping region for TTYDISPLAYSTREAM has been set so that \#DISPLAYLINES will get set correctly.
                        (DSPRIGHTMARGIN SCREENWIDTH TtyDisplayStream))])

(\STARTDISPLAY
  [LAMBDA NIL                                                   (* kbr%: "19-Jan-86 14:52")
    (PROG (OLDWINDOWS)
          (UPDATESCREENDIMENSIONS)
          [COND
             ((AND (OR (NOT (EQ SCREENWIDTH \OLDSCREENWIDTH))
                       (NOT (EQ SCREENHEIGHT \OLDSCREENHEIGHT)))
                   \WINDOWWORLD)

              ;; Need to move windows around so that they remain on screen, and/or fix the display to account for new raster width

              (SETQ OLDWINDOWS (REVERSE (OPENWINDOWS)))         ; Returns bottom window first
              (COND
                 ((OR (LESSP SCREENWIDTH \OLDSCREENWIDTH)
                      (LESSP SCREENHEIGHT \OLDSCREENHEIGHT))    ; Screen shrank, movement needed
                  (\MOVE.WINDOWS.ONTO.SCREEN OLDWINDOWS)))

              ;; Finally, close the windows to save their images.  Do this in separate pass from the moving, in case somebody's MOVEFN tried to
              ;; do something with a window we had closed

              (for W in OLDWINDOWS do (\CLOSEW1 W))
              (COND
                 ((AND NIL (NOT (EQ SCREENWIDTH \OLDSCREENWIDTH)))
                  (\UPDATE.PBT.RASTERWIDTHS]
          (UNINTERRUPTABLY
              (SETQ ScreenBitMap (\CreateScreenBitMap SCREENWIDTH SCREENHEIGHT))
              (SHOWDISPLAY (fetch (BITMAP BITMAPBASE) of ScreenBitMap)
                     (fetch (BITMAP BITMAPRASTERWIDTH) of ScreenBitMap))
              (SETQ \DisplayStarted T))
          (SETQ WHOLESCREEN (SETQ WHOLEDISPLAY
                             (create REGION
                                    LEFT _ 0
                                    BOTTOM _ 0
                                    WIDTH _ SCREENWIDTH
                                    HEIGHT _ SCREENHEIGHT)))
          (COND
             (\MAINSCREEN (replace (SCREEN SCDESTINATION) of \MAINSCREEN with ScreenBitMap)
                    (replace (SCREEN SCWIDTH) of \MAINSCREEN with SCREENWIDTH)
                    (replace (SCREEN SCHEIGHT) of \MAINSCREEN with SCREENHEIGHT)))
          (SETQ \CURSORDESTINATION ScreenBitMap)
          (SETQ \CURSORDESTWIDTH SCREENWIDTH)
          (SETQ \CURSORDESTHEIGHT SCREENHEIGHT)
          (SETQ \CURSORDESTRASTERWIDTH (fetch (BITMAP BITMAPRASTERWIDTH) of ScreenBitMap))
          [COND
             (OLDWINDOWS                                        ; Now that we've created ScreenBitMap with the right raster
                                                                ; width, put the windows back up
                    (CHANGEBACKGROUND WINDOWBACKGROUNDSHADE)
                    (for W in (REVERSE OLDWINDOWS) do (\OPENW1 W]
          (SETQ \OLDSCREENHEIGHT SCREENHEIGHT)
          (SETQ \OLDSCREENWIDTH SCREENWIDTH])

(\MOVE.WINDOWS.ONTO.SCREEN
  [LAMBDA (WINDOWS)                                             (* bvm%: "15-Aug-85 15:08")
    (COND
       ([for W in WINDOWS thereis (LET ((REG (fetch (WINDOW REG) of W)))
                                       (OR (GREATERP (fetch (REGION RIGHT) of REG)
                                                  SCREENWIDTH)
                                           (GREATERP (fetch (REGION TOP) of REG)
                                                  SCREENHEIGHT]    ; Move all windows some if any are off screen
        (LET (XFACTOR YFACTOR REG)
             (SETQ XFACTOR (FQUOTIENT SCREENWIDTH \OLDSCREENWIDTH))
             (SETQ YFACTOR (FQUOTIENT SCREENHEIGHT \OLDSCREENHEIGHT))
             (for W in WINDOWS unless (NEQ W (MAINWINDOW W))
                do ;; In the case of attached windows, move only the main one, so that attached windows are properly dragged along

                   (MOVEW (SETQ W (MAINWINDOW W T))
                          (IMAX 0 (IDIFFERENCE [FIXR (FTIMES XFACTOR (fetch (REGION RIGHT)
                                                                        of (SETQ REG (fetch (WINDOW REG)
                                                                                        of W]
                                         (fetch (REGION WIDTH) of REG)))
                          (IMAX 0 (IDIFFERENCE (FIXR (FTIMES YFACTOR (fetch (REGION TOP) of REG)))
                                         (fetch (REGION HEIGHT) of REG])

(\UPDATE.PBT.RASTERWIDTHS
  [LAMBDA NIL                                                   (* bvm%: "11-Aug-85 00:12")

;;; Fix all the cached bitblt tables that think they know what the screen width is

    (\MAPMDS 'PILOTBBT (FUNCTION (LAMBDA (PAGENO)
                                  (to (FOLDLO \MDSIncrement 16) bind (PBT _ (create POINTER
                                                                               PAGE# _ PAGENO))
```

```
                                                    do

                                                   ;; NOTE: We are depending on PILOTBBT structures being 16-word units, and that the first 32-bit
                                                   ;; field is NOT the one we are smashing.  That's so we don't trash links in the free list.  In fact, since
                                                   ;; PBTDESTLO and PBTDESTHI are in the first 32-bit field, we are actually guaranteed by the AND
                                                   ;; below not to touch any free PILOTBBT structures

                                                       (COND
                                                          ((AND (EQ (fetch (PILOTBBT PBTDESTHI) of PBT)
                                                                    (FOLDLO \VP.DISPLAY PAGESPERSEGMENT))
                                                                (EQ (fetch (PILOTBBT PBTDESTLO) of PBT)
                                                                    0))           ; Destination is screen
                                                           (replace (PILOTBBT PBTDESTBPL) of PBT with SCREENWIDTH)))
                                                       (SETQ PBT (\ADDBASE PBT 16])

(\STOPDISPLAY
  [LAMBDA NIL                                                          (* lmm " 7-Jan-86 17:59")
    (DECLARE (GLOBALVARS \MaxScreenPage))

    ;; Turn off Lisp display, go back to bcpl display.  Exists only for emergency use

    (UNINTERRUPTABLY
        (SHOWDISPLAY)
        (\UNLOCKPAGES (fetch BITMAPBASE of ScreenBitMap)
               (ADD1 \MaxScreenPage))
        (SETQ \MaxScreenPage -1)
        (SETQ \DisplayStarted NIL))
    (PAGEHEIGHT 58])

(\DEFINEDISPLAYINFO
  [LAMBDA (DISPLAYINFO)                                                (* kbr%: " 1-Jul-85 17:39")
    (PROG (BUCKET)
          (SETQ BUCKET (ASSOC (CAR DISPLAYINFO)
                              \DISPLAYINFOALIST))
          (COND
             (BUCKET (DREMOVE BUCKET \DISPLAYINFOALIST)))
          (push \DISPLAYINFOALIST DISPLAYINFO])
)

(DECLARE%: EVAL@COMPILE DONTCOPY

(ADDTOVAR DONTCOMPILEFNS \UPDATE.PBT.RASTERWIDTHS)
)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(PUTPROPS DISPLAYINITIALIZEDP MACRO (NIL                               (* always initialized now)
                                          T))

(PUTPROPS DISPLAYSTARTEDP MACRO (NIL \DisplayStarted))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \DisplayStarted \DisplayStreamsInitialized \DisplayInitialed WHOLEDISPLAY WHOLESCREEN SCREENWIDTH
       SCREENHEIGHT)
)

;; END EXPORTED DEFINITIONS

(ADDTOVAR GLOBALVARS WHOLESCREEN)

(DEFINEQ

(INITIALIZEDISPLAYSTREAMS
  [LAMBDA NIL                                                          (* lmm " 7-Jan-86 16:51")
    (SETQ WHOLEDISPLAY (create REGION))
    (SETQ \SYSPILOTBBT (create PILOTBBT))                              ; For BITBLT
    (SETQ \SYSBBTEXTURE (BITMAPCREATE 16 16))                          ; For texture handling in \BITBLTSUB
                                                                       ; A guaranteed display font is initialized here after pup, font, and
                                                                       ; bitmap code has been loaded.
    (SETQ \GUARANTEEDDISPLAYFONT (FONTCREATE 'GACHA 10 NIL NIL 'DISPLAY))
    (SETQ DEFAULTFONT (FONTCLASS 'DEFAULTFONT (LIST 1 \GUARANTEEDDISPLAYFONT])

)

(DECLARE%: DOCOPY DONTEVAL@LOAD

(RPAQQ \DisplayStarted NIL)

(RPAQQ \LastTTYLines 12)

(INITIALIZEDISPLAYSTREAMS)
(DISPLAYSTREAMINIT 1000)
)

(PUTPROPS LLDISPLAY FILETYPE COMPILE-FILE)

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA )
)
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX

CONSTANT INDEX

OPTIMIZER INDEX

RECORD INDEX

PROPERTY INDEX