File created: 10-Apr-2023 07:05:18 **{DSK}<home>larry>il>medley>sources>WINDOW.;2**

edit by: lmm

changes to: (VARS WINDOWCOMS)

previous date: 9-Jul-2022 11:10:09 {DSK}<home>larry>il>medley>sources>WINDOW.;1

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
;; Copyright (c) 1982-1988, 1990-1994, 1999-2000, 2021 by Venue & Xerox Corporation.

(RPAQQ WINDOWCOMS
       [(COMS (FNS WINDOWWORLD WINDOWWORLDP CHANGEBACKGROUND CHANGEBACKGROUNDBORDER TILE
                   \TTY.CREATING.DISPLAYSTREAM \CREATE.TTY.OUTCHARFN \CREATE.TTYDISPLAYSTREAM HASTTYWINDOWP
                   TTYINFOSTREAM CREATESCREEN \INSURESCREEN \BITMAPTOSCREEN MAINSCREEN)
              (VARS (\TTYREGIONOFFSETSPTR))
              (INITVARS [TTYREGIONOFFSETS '((0 . 0)
                                            (20 . -20)
                                            (40 . 0)
                                            (20 . 20]
                     (DEFAULTTTYREGION '(153 100 384 208))
                     (INITIAL-EXEC-REGION '(8 378 550 330))
                     (INITIAL-PROMPT-REGION '(8 719 550 89))
                     (\MAINSCREEN)
                     (\CURRENTBACKGROUNDBORDER)
                     (\SCREENS)
                     (\SCREENBITMAPS))
              (GLOBALVARS \TTYREGIONOFFSETSPTR TTYREGIONOFFSETS \DEFAULTTTYDISPLAYSTREAM)
              (VARIABLES \TopLevelTtyWindow))
        (COMS                                                    ; Window menu operations
              (FNS WINDOW.MOUSE.HANDLER \PROTECTED.APPLY DOWINDOWCOM DOBACKGROUNDCOM DEFAULT.BACKGROUND.COPYFN)
              (VARS (BackgroundCopyMenu))
              (INITVARS BackgroundCopyMenuCommands)
              (FNS BURYW CLEARW CLOSEW \CLOSEW1 \OKTOCLOSEW \INTERACTIVE.CLOSEW OPENW DOUSERFNS DOUSERFNS2
                   \USERFNISDON'T \OPENW1 CREATEW CREATEW1 \CREATEW1 OPENDISPLAYSTREAM MOVEW PPROMPT3
                   \ONSCREENCLIPPINGREGION RELMOVEW SHAPEW SHAPEW1 \SHAPEW2 RESHOWBORDER \RESHOWBORDER1 TRACKW
                   SNAPW WINDOWREGION)
              (FNS MINIMUMWINDOWSIZE)
              (INITVARS (BACKGROUNDCURSORINFN)
                     (BACKGROUNDBUTTONEVENTFN)
                     (BACKGROUNDCURSOROUTFN)
                     (BACKGROUNDCURSORMOVEDFN)
                     (BACKGROUNDCOPYBUTTONEVENTFN)
                     (BACKGROUNDCOPYRIGHTBUTTONEVENTFN (FUNCTION DEFAULT.BACKGROUND.COPYFN))
                     (BACKGROUNDCURSOREXITFN))
              (GLOBALVARS BACKGROUNDCURSORINFN BACKGROUNDBUTTONEVENTFN BACKGROUNDCURSOROUTFN
                     BACKGROUNDCURSORMOVEDFN BACKGROUNDCOPYBUTTONEVENTFN BACKGROUNDCOPYRIGHTBUTTONEVENTFN
                     \CARET.UP BACKGROUNDCURSOREXITFN)
              (EXPORT (MACROS .COPYKEYDOWNP. WSOP))
              (PROP ARGNAMES WSOP)
              (RECORDS WSOPS WSDATA))
        (COMS                                                    ; Window utilities
              (FNS ADVISEWDS SHOWWFRAME SHOWWTITLE \STRINGWIDTHGUESS RESHOWTITLE TOTOPW \INTERNALTOTOPW \TTW1
                   WHICHW)
              (INITVARS (WINDOWTITLEPRINTLEVEL '(2 . 5))
                     (WINDOWTITLESHADE BLACKSHADE)))
        [COMS                                                    ; Window vs non-window world
              (FNS WFROMDS NU\TOTOPWDS \COERCETODS)
              (DECLARE%: DONTCOPY (EXPORT (MACROS \COERCETODS .WHILE.ON.TOP.)))
              (P (MOVD 'NU\TOTOPWDS '\TOTOPWDS]
        (COMS                                                    ; User interface functions
              (FNS WINDOWP INSURE.WINDOW WINDOWPROP WINDOWADDPROP WINDOWDELPROP GETWINDOWPROP GETWINDOWUSERPROP
                   PUTWINDOWPROP REMWINDOWPROP WINDOWADDFNPROP)
                                                                 ; Compiled WINDOWPROP
              (PROP ARGNAMES WINDOWPROP)
              (OPTIMIZERS WINDOWPROP)
              (FNS CWINDOWPROP CGETWINDOWPROP \GETWINDOWHEIGHT \GETWINDOWWIDTH))
        (FNS WINDOW.BITMAP)
                                                                 ; lmm 4/23
        (COMS (FNS OPENWP TOPWP RESHAPEBYREPAINTFN \INBETWEENP DECODE/WINDOW/OR/DISPLAYSTREAM GROW/REGION
                   CLRPROMPT PROMPTPRINT OPENWINDOWS \INSUREWINDOW)
                                                                 ; these entries are left in for backward compatibility. They were
                                                                 ; dedocumented 6/83.  rrb
              (P (MOVD 'OPENWP 'ACTIVEWP))
              (FNS OVERLAPPINGWINDOWS WOVERLAPP ORDERFROMBOTTOMTOTOP)
                                                                 ; screen size changing functions.
              (FNS \ONSCREENW \PUTONSCREENW \UPDATECACHEDFIELDS \WWCHANGESCREENSIZE CREATEWFROMIMAGE
                   UPDATEWFROMIMAGE))
        [COMS ;; MEDLEY-NATIVE-WINDOWS INTERFACE FUNCTIONS
```

```
        (GLOBALVARS \SCREENS \SCREENTYPES)
        [INITVARS
                ;; \SCREENS is a list of all known screens.  The SCREEN-CREATE function for the screen type must register it there.  It's
                ;; used, e.g., by DSPCREATE to find the right screen given a screen bitmap.
                (\SCREENS)
                ;; \SCREENTYPES is used to interpret the values we get back from the query-for-screen-types function, and to look up the
                ;; methods for creating a screen and destroying one.
                (\SCREENTYPES '((1 MEDLEY OPEN-SCREEN CREATESCREEN CLOSE-SCREEN NILL)
                                (2 MEDLEY-COLOR-4)
                                (4 MEDLEY-COLOR-8)
                               y
                                (8 MEDLEY-COLOR-24)
                                (16 X-MONO)
                                (32 X-COLOR)
                                (64 MS-WINDOWS]
        ;; OLD-MEDLEY-SCREEN window management functions
        (FNS \MEDW.CREATEW \MEDW.OPENW \MEDW.CLOSEW \MEDW.MOVEW \MEDW.RELMOVEW \MEDW.SHRINKW \MEDW.EXPANDW
             \MEDW.SHAPEW \MEDW.REDISPLAYW \MEDW.BURYW \MEDW.TOTOPW \MEDW.DSPCREATE \GENERIC.DSPCREATE
             \GENERIC.DSPCREATE.DESTINATION.BITMAP? \MEDW.GETWINDOWPROP \MEDW.PUTWINDOWPROP \MEDW.CURSOR)
        (FNS \GENERIC.CURSOR)
        (DECLARE%: EVAL@COMPILE DONTCOPY (EXPORT (MACROS WINDOWOP)))
        (DECLARE%: DONTEVAL@COMPILE DONTEVAL@LOAD DOCOPY
                ;; Install the generic DSPCREATE over the simple one defined in LLDISPLAY.
                (P (MOVD '\GENERIC.DSPCREATE 'DSPCREATE)
                   (CL:UNLESS (EQUAL (GETD 'CURSOR)
                                     (GETD '\GENERIC.CURSOR))
                              (MOVD '\GENERIC.CURSOR 'CURSOR))]
(DECLARE%: EVAL@COMPILE DONTCOPY
       (GLOBALVARS \LastCursorPosition \LastInWindow WindowMenu BackgroundMenu BackgroundMenuCommands
              \LastWindowButtons WWFNS WindowMenuCommands WindowTitleDisplayStream WINDOWTITLEPRINTLEVEL
              WBorder \TOPWDS WINDOWBACKGROUNDSHADE BACKGROUNDFNS)
       (EXPORT (CONSTANTS (MinWindowWidth 26)
                          (MinWindowHeight 16))
               (RECORDS WINDOW SCREEN)))
(DECLARE%: EVAL@COMPILE (EXPORT (GLOBALVARS WINDOWUSERFORMS ENDOFWINDOWUSERFORMS PROMPTWINDOW)))
(SYSRECORDS WINDOW SCREEN)
(INITRECORDS WINDOW SCREEN)
(INITVARS (WindowMenu)
       (BackgroundMenu)
       (\LastCursorPosition (CREATEPOSITION))
       (\LastInWindow)
       (\LastWindowButtons 0)
       (WINDOWBACKGROUNDSHADE 34850)
       (WBorder 4)
       (HIGHLIGHTSHADE 32800)
       (WINDOWBACKGROUNDBORDER 34850))
(FILES PAINTW)
[ADDVARS (WindowMenuCommands (Close '\INTERACTIVE.CLOSEW "Closes a window")
                (Snap 'SNAPW "Saves a snapshot of a region of the screen.")
                (Paint 'PAINTW "Starts a painting mode in which the mouse can be
                       used to draw pictures or make notes on windows.")
                (Clear 'CLEARW "Clears a window to its gray.")
                (Bury 'BURYW "Puts a window on the bottom.")
                (Redisplay 'REDISPLAYW "Redisplays a window using its REPAINTFN.")
                (Hardcopy 'HARDCOPYIMAGEW "Prints a window using its HARDCOPYFN."
                       (SUBITEMS ("To a file" 'HARDCOPYIMAGEW.TOFILE "Puts image on a file; prompts for
                                         filename and format")
                                 ("To a printer" 'HARDCOPYIMAGEW.TOPRINTER "Sends image to a printer of
                                        your choosing")))
                (Move 'MOVEW "Moves a window by a corner.")
                (Shape 'SHAPEW "Gets a new region for a window.
                       Left button down marks fixed corner; sweep to other corner.
                       Middle button down moves closest corner.")
                (Shrink 'SHRINKW "Replaces this window with its icon (or title if it doesn't have an
                       icon."))
       (BackgroundMenuCommands (SaveVM '(SAVEVM)
                                       "Updates the virtual memory.")
              (Snap '(SNAPW)
                    "Saves a snapshot of a region of the screen.")
              (Hardcopy '(HARDCOPYW)
                     "Send hardcopy of screen region to printer."
                     (SUBITEMS ("To a file" '(HARDCOPYREGION.TOFILE)
                                       "Writes a region of screen to a file; prompts for filename and
                                       format")
                               ("To a printer" '(HARDCOPYREGION.TOPRINTER)
                                     "Sends a region of screen to a printer of your choosing"]
(ADDVARS (WINDOWUSERFORMS)
       (ENDOFWINDOWUSERFORMS))
(DECLARE%: DONTEVAL@LOAD DOCOPY (P (COND ((NULL \MAINSCREEN)
                                          (SETQ \MAINSCREEN (CREATESCREEN (SCREENBITMAP)))
                                          (SETQ \CURSORSCREEN \MAINSCREEN)
                                          (SETQ LASTSCREEN \MAINSCREEN)
                                          (WINDOWWORLD 'ON \MAINSCREEN T)))
```

```
                                        (MOVD? 'TRUE 'LISPWINDOWP))
        (VARS (\WINDOWWORLD T)))
```

;; Arrange for the proper compiler

```
(PROP FILETYPE WINDOW)
(DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                       (NLAML)
                                                                       (LAMA PROMPTPRINT WINDOWPROP
                                                                             DOWINDOWCOM])

(DEFINEQ
```

(**WINDOWWORLD**

```
[LAMBDA (ONOFF SCREEN MAINFLG)                          ; Edited 28-Feb-94 13:07 by sybalsky
```

;; ONOFF should be ON or OFF. SCREEN will generally be either \MAINSCREEN or \COLORSCREEN. MAINFLG = T if this is the first window
;; world being created (\MAINSCREEN), in which case we create the EXEC window, PROMPTWINDOW, and LOGOW.

```
(DECLARE (GLOBALVARS \TopLevelTtyWindow))
(PROG NIL
      (SETQ SCREEN (\INSURESCREEN SCREEN))
      (COND
         ((NULL ONOFF)
          (RETURN (fetch (SCREEN SCONOFF) of SCREEN)))
         ((EQ ONOFF (fetch (SCREEN SCONOFF) of SCREEN))      ; Already on or off. *

          )
         ((EQ ONOFF 'ON)
          (UNINTERRUPTABLY
              (\CLEARBM (fetch (SCREEN SCDESTINATION) of SCREEN)
                     WINDOWBACKGROUNDSHADE)                  ; Initially there are no windows. SCTOPW must be NIL before
                                                             ; any CREATEWs are done.
              (replace (SCREEN SCTOPW) of SCREEN with NIL)
              (CHANGEBACKGROUNDBORDER WINDOWBACKGROUNDBORDER)
              (SETQ \TOPWDS NIL)
              (CL:PUSHNEW (fetch (SCREEN SCDESTINATION) of SCREEN)
                     \SCREENBITMAPS)
              (CL:PUSHNEW SCREEN \SCREENS)
              (replace (SCREEN SCONOFF) of SCREEN with 'ON)
              [COND
                 (MAINFLG                                    ; creating the first window system
```

;; set up stream for displaying titles

```
                  (SETQ WindowTitleDisplayStream (fetch (SCREEN SCTITLEDS) of SCREEN))
                                                             ; Get TTY in shape. Region is only approx as user can change
                                                             ; it.
```

;; default display stream will create a window when it needs one

```
                  (SETQ \DEFAULTTTYDISPLAYSTREAM (\TTY.CREATING.DISPLAYSTREAM))
```

;; create the exec window

```
                  (TTYDISPLAYSTREAM (SETQ \TopLevelTtyWindow (CREATEW INITIAL-EXEC-REGION "Exec")))
                  (SETLINELENGTH)
                  (SETQ PROMPTWINDOW (CREATEW INITIAL-PROMPT-REGION "Prompt Window" 2))
                  (DSPTEXTURE BLACKSHADE PROMPTWINDOW)
                  (DSPOPERATION 'ERASE PROMPTWINDOW)
                  (DSPSCROLL 'ON PROMPTWINDOW)
                  (WINDOWPROP PROMPTWINDOW 'SHRINKFN 'DON'T)
                  (CLEARW PROMPTWINDOW)
                  (WINDOWPROP PROMPTWINDOW 'PAGEFULLFN (FUNCTION NILL))
                  (replace (SCREEN PROMPTW) OF SCREEN with PROMPTWINDOW)
```

;; window.mouse.handler variables?

```
                  (SETQ \LastInWindow NIL)
                  (SETQ \LastWindowButtons 0)
                  (SETQ \LastCursorPosition (create POSITION))
```

;; other things that happen at WINDOWWORLD time

```
                  (MAPC WINDOWUSERFORMS (FUNCTION EVAL])])
```

(**WINDOWWORLDP**

```
[LAMBDA (SCREEN)                                            (* kbr%: "30-Mar-85 14:28")
                                                            ; is the window system operating?
   (EQ (fetch (SCREEN SCONOFF) of (\INSURESCREEN SCREEN))
       'ON])
```

(**CHANGEBACKGROUND**

```
[LAMBDA (SHADE SCREEN)                                      ; Edited 6-Jul-88 11:39 by drc:
                                                            ; changes the window world background to SHADE
   (PROG (WINDOWS)
         (COND
            ((OR (NULL SHADE)
                 (EQ SHADE T))
             (SETQ SHADE WINDOWBACKGROUNDSHADE))
            ((NOT (OR (TEXTUREP SHADE)
                      (BITMAPP SHADE)))
```

```
            (\ILLEGAL.ARG SHADE)))
         (OR SCREEN (SETQ SCREEN \CURSORSCREEN))
         (SETQ WINDOWS (OPENWINDOWS SCREEN))
         (for W in WINDOWS do (\CLOSEW1 W))
         [COND
            ((TEXTUREP SHADE)
             (BLTSHADE SHADE (fetch (SCREEN SCDESTINATION) of SCREEN)))
            ((BITMAPP SHADE)
             (TILE SHADE (fetch (SCREEN SCDESTINATION) of SCREEN]
         (for W in (DREVERSE WINDOWS) do (\OPENW1 W])
```

(CHANGEBACKGROUNDBORDER

```
[LAMBDA (SHADE)                                                    (* lmm "25-Apr-86 15:48")
```

;; Changes the screen border on a Dandelion. SHADE is a 8x2 pattern

```
(PROG1 \CURRENTBACKGROUNDBORDER
    (COND
       ((SMALLP SHADE)
        (SETQ \CURRENTBACKGROUNDBORDER SHADE)
        (SELECTC \MACHINETYPE
            (\DANDELION (replace (IOPAGE DLDISPBORDER) of \IOPAGE with SHADE))
            (\DAYBREAK (\DoveDisplay.SetBorderPattern SHADE))
            NIL))))])
```

(TILE

```
[LAMBDA (SRC DST)                                                  (* kbr%: "10-Jul-85 23:51")
   (PROG (X Y W H DSTW DSTH)
         (SETQ X 0)
         (SETQ Y 0)
         (SETQ W (BITMAPWIDTH SRC))
         (SETQ H (BITMAPHEIGHT SRC))
         (SETQ DSTW (BITMAPWIDTH DST))
         (SETQ DSTH (BITMAPHEIGHT DST))
         (while (ILESSP X DSTW) do (SETQ Y 0)
                                   (while (ILESSP Y DSTH)
                                      do (BITBLT SRC 0 0 DST X Y W H NIL 'REPLACE)
                                         (add Y H))
                                   (add X W])
```

(\TTY.CREATING.DISPLAYSTREAM

```
[LAMBDA NIL                                                        ; Edited 13-Jun-2021 10:14 by rmk:
```

;; creates a displaystream that points to a stream that has a OUTCHARFN that creates a new displaystream. It is used as the default
;; TtyDisplayStream in a process.

```
(PROG [(DS (DSPCREATE (BITMAPCREATE 1 1]
      (replace (STREAM OUTCHARFN) of DS with (FUNCTION \CREATE.TTY.OUTCHARFN))
      (replace (STREAM FULLFILENAME) of DS with T)
      (RETURN DS])
```

(\CREATE.TTY.OUTCHARFN

```
[LAMBDA (STREAM CHAR)                                              ; Edited  8-Mar-87 14:58 by bvm:
```

;; outcharfn for \DEFAULTTTYDISPLAYSTREAM which creates a new window and then bouts to it.

```
(\OUTCHAR (\CREATE.TTYDISPLAYSTREAM)
       CHAR])
```

(\CREATE.TTYDISPLAYSTREAM

```
[LAMBDA NIL                                                        ; Edited  9-Mar-87 13:05 by bvm:
```

;; Called when system attempts input from or output to the "default tty stream", a dummy stream that is every new process's initial standard i/o. We
;; make a new window to be the ttydisplaystream, and return the stream.

```
[COND
   ((AND \WINDOWWORLD (NOT (HASTTYWINDOWP NIL)))
```

;; Check that the process does not yet have a tty window. We can get called even after one is created in the case where somebody explicitly
;; passed (TTYDISPLAYSTREAM) or *STANDARD-OUTPUT* as an argument to someone else (e.g., as stream arg to a printing fn that prints
;; more than one character). In this case, TTYDISPLAYSTREAM can't update the private variable holding the stream, so the dummy
;; outcharfn gets called again. So avoid creating a second window!

;; \windowworld check is to prevent error when loading window during loadup

```
    (COND
       ((NULL (SETQ \TTYREGIONOFFSETSPTR (CDR \TTYREGIONOFFSETSPTR)))
                                                                   ; the offsets distribute the break windows a little so many can be
                                                                   ; seen.
        (SETQ \TTYREGIONOFFSETSPTR TTYREGIONOFFSETS)))
    (SETQ \TTYWINDOW (CREATEW (CREATEREGION (IPLUS (fetch (REGION LEFT) of DEFAULTTTYREGION)
                                                   (CAR (CAR \TTYREGIONOFFSETSPTR)))
                                           (IPLUS (fetch (REGION BOTTOM) of DEFAULTTTYREGION)
                                                  (CDR (CAR \TTYREGIONOFFSETSPTR)))
                                           (fetch (REGION WIDTH) of DEFAULTTTYREGION)
                                           (fetch (REGION HEIGHT) of DEFAULTTTYREGION))
                              (CONCAT "TTY window for " (PROCESSPROP (THIS.PROCESS)
                                                                    'NAME))
```

```
                                NIL T))
```

;; \TTYWINDOW (bound at top of each process) saves the window so it won't get collected. This allows WFROMDS to find it even if it is
;; closed, which is how we create it initially (in case no output ever actually happens). In future, if windows become streams this can go away.

```
    (TTYDISPLAYSTREAM \TTYWINDOW))
   ((EQ *STANDARD-OUTPUT* \DEFAULTTTYDISPLAYSTREAM)          ; Somebody bound *STANDARD-OUTPUT* at the time the tty
                                                             ; window got created, so masked this binding. Fix it now to avoid
                                                             ; future calls here
    (SETQ *STANDARD-OUTPUT* (TTYDISPLAYSTREAM]
(TTYDISPLAYSTREAM])
```

(**HASTTYWINDOWP**

```
[LAMBDA (PROCESS)                                            (* lmm "17-Jan-86 20:31")
```

;; True if PROCESS has a tty window already.

```
(NEQ (OR (PROCESS.TTY PROCESS)
         \DEFAULTTTYDISPLAYSTREAM)
     \DEFAULTTTYDISPLAYSTREAM])
```

(**TTYINFOSTREAM**

```
[LAMBDA (PROCESS)                                            ; Edited 7-Mar-94 11:58 by sybalsky
```

;;; Returns a stream to which to print informative messages = PROCESS tty if PROCESS has one, else PROMPTWINDOW

```
(DECLARE (GLOBALVARS \DEFAULTTTYDISPLAYSTREAM))
(PROG ((STREAM (PROCESS.TTY PROCESS)))
      (RETURN (COND
                 ((AND STREAM (NEQ STREAM \DEFAULTTTYDISPLAYSTREAM))
                  STREAM)
                 (T (\GETSTREAM PROMPTWINDOW])
```

(**CREATESCREEN**

```
[LAMBDA (DESTINATION)                                        ; Edited 2-Mar-94 01:44 by sybalsky
```

;;; destination is the framebuffer for the screen you want created.e.g. (SCREENBITMAP) Creates a screen describing a medley regular window system.

```
(PROG (TITLEDS SCREEN)
      (SETQ TITLEDS (DSPCREATE DESTINATION))                 ; Create TITLEDS.
      (DSPOPERATION 'INVERT TITLEDS)
      (DSPFONT WINDOWTITLEFONT TITLEDS)
      (DSPRIGHTMARGIN MAX.SMALLP TITLEDS)                    ; Set right margin so title doesn't autoCR.
```

;; now create SCREEN.

```
      (SETQ SCREEN (create SCREEN
                           SCONOFF _ 'OFF
                           SCDESTINATION _ DESTINATION
                           SCWIDTH _ (BITMAPWIDTH DESTINATION)
                           SCHEIGHT _ (BITMAPHEIGHT DESTINATION)
                           SCDEPTH _ (BITSPERPIXEL DESTINATION)
                           SCTOPW _ NIL
                           SCTITLEDS _ TITLEDS
                           CREATEWFN _ (FUNCTION \MEDW.CREATEW)
                           OPENWFN _ (FUNCTION \MEDW.OPENW)
                           CLOSEWFN _ (FUNCTION \MEDW.CLOSEW)
                           MOVEWFN _ (FUNCTION \MEDW.MOVEW)
                           RELMOVEWFN _ (FUNCTION \MEDW.RELMOVEW)
                           SHRINKWFN _ (FUNCTION \MEDW.SHRINKW)
                           EXPANDWFN _ (FUNCTION \MEDW.EXPANDW)
                           SHAPEWFN _ (FUNCTION \MEDW.SHAPEW)
                           REDISPLAYFN _ (FUNCTION \MEDW.REDISPLAYW)
                           BURYWFN _ (FUNCTION \MEDW.BURYW)
                           TOTOPWFN _ (FUNCTION \MEDW.TOTOPW)
                           DSPCREATEFN _ (FUNCTION \MEDW.DSPCREATE)
                           GETWINDOWPROPFN _ (FUNCTION \MEDW.GETWINDOWPROP)
                           PUTWINDOWPROPFN _ (FUNCTION \MEDW.PUTWINDOWPROP)
                           SETCURSORFN _ (FUNCTION \MEDW.CURSOR)
                           WINIMAGEOPS _ \DISPLAYIMAGEOPS
                           WINFDEV _ DisplayFDEV
                           BBTTOWIN _ (FUNCTION \MEDW.BBTTOWIN)
                           BBTFROMWIN _ (FUNCTION \MEDW.BBTFROMWIN)
                           BBTWINWIN _ (FUNCTION \MEDW.BBTWINWIN)
                           SCCARETFLASH _ (FUNCTION \MEDW.CARET.SHOW)
                           SCGETSCREENPOSITION _ (FUNCTION \MEDW.GETSCREENPOSITION)
                           SCGETBOXSCREENPOSITION _ (FUNCTION \MEDW.GETBOXSCREENPOSITION)
                           SCGETSCREENREGION _ (FUNCTION \MEDW.GETSCREENREGION)))
      (CL:PUSHNEW SCREEN \SCREENS)                           ; Register this screen.
      (RETURN SCREEN])
```

(**\INSURESCREEN**

```
[LAMBDA (SCREEN)                                             (* kbr%: " 4-Aug-85 13:30")
  (COND
     ((type? SCREEN SCREEN)
      SCREEN)
```

```
        ((NULL SCREEN)
         \CURSORSCREEN)
        (T (\ILLEGAL.ARG SCREEN])
```

(\BITMAPTOSCREEN

```
  [LAMBDA (BITMAP)                                          (* gbn%: "25-Jan-86 16:44")
```

;;; returns the screen with this bitmap as its destination, NIL otherwise

```
    (for SCREEN in \SCREENS thereis (EQ (fetch (SCREEN SCDESTINATION) of SCREEN)
                                        BITMAP])
```

(MAINSCREEN

```
  [LAMBDA NIL                                               (* kbr%: " 2-Feb-86 14:55")
    \MAINSCREEN])

)

(RPAQQ \TTYREGIONOFFSETSPTR NIL)

(RPAQ? TTYREGIONOFFSETS '((0 . 0)
                          (20 . -20)
                          (40 . 0)
                          (20 . 20)))

(RPAQ? DEFAULTTTYREGION '(153 100 384 208))

(RPAQ? INITIAL-EXEC-REGION '(8 378 550 330))

(RPAQ? INITIAL-PROMPT-REGION '(8 719 550 89))

(RPAQ? \MAINSCREEN )

(RPAQ? \CURRENTBACKGROUNDBORDER )

(RPAQ? \SCREENS )

(RPAQ? \SCREENBITMAPS )

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \TTYREGIONOFFSETSPTR TTYREGIONOFFSETS \DEFAULTTTYDISPLAYSTREAM)
)

(DEFGLOBALVAR \TopLevelTtyWindow)

;; Window menu operations

(DEFINEQ
```

(WINDOW.MOUSE.HANDLER

```
  [LAMBDA NIL                                               ; Edited 22-Mar-94 13:31 by sybalsky
```

;;; Does user window operations if state of buttons has changed or mouse has changed windows

```
    (COND
       (\INTERRUPTABLE                                      ; don't do anything if uninterruptable
              (PROG (\MHCOM \MHPROCESS \MHWINDOW)
                    (GETMOUSESTATE)
                    [COND
                       ((OR (NEQ LASTMOUSEX (fetch XCOORD of \LastCursorPosition))
                            (NEQ LASTMOUSEY (fetch YCOORD of \LastCursorPosition))
                            (NEQ LASTMOUSEBUTTONS \LastWindowButtons))
                                                            ; Cursor has changed position or a button is down, see if it is in a
                                                            ; window or scroll area.
                        (PROG ((\MOUSEBUSY T))
                              (DECLARE (SPECVARS \MOUSEBUSY))   ; Indicates to others that the mouse process is doing something
                                                            ; 'interesting'
                              (replace XCOORD of \LastCursorPosition with LASTMOUSEX)
                              (replace YCOORD of \LastCursorPosition with LASTMOUSEY)
                              (SETQ \MHWINDOW (WHICHW LASTMOUSEX LASTMOUSEY \CURSORSCREEN))
                              [COND
                                 ((NEQ \MHWINDOW \LastInWindow)
                                  ;; Cursor has moved outside the current window, check to see if it moved into the scroll area and that the
                                  ;; scroll handler wants it.
                                  (COND
                                     ((AND \LastInWindow (LISPWINDOWP \LastInWindow)
                                           (IN/SCROLL/BAR? \LastInWindow LASTMOUSEX LASTMOUSEY)
                                           (PROGN                   ; SCROLL.HANDLER returns NIL if this window doesn't want to
                                                                    ; scroll.
                                                  (SCROLL.HANDLER \LastInWindow)))
                                      (replace XCOORD of \LastCursorPosition with -1)
                                      (GO RESETBUTTONS))
```

```
        [(OR (EQ LASTMOUSEBUTTONS 0)
             (NEQ LASTMOUSEBUTTONS \LastWindowButtons))
        ;; Cursor has changed windows, so call CURSOROUTFN of old window, CURSORINFN of new.  The
        ;; user enters another window by moving the cursor into it with no buttons pressed or by pressing a
        ;; button in the window.  This allows the user to go into a window with a button down, release it and
        ;; still be 'in' the window he came from.
         [COND
            ((NULL \LastInWindow)
             (AND BACKGROUNDCURSOROUTFN (GETD BACKGROUNDCURSOROUTFN)
                  (\PROTECTED.APPLY BACKGROUNDCURSOROUTFN)))
            ((SETQ \MHCOM (fetch (WINDOW CURSOROUTFN) of \LastInWindow))
             (ERSETQ (DOUSERFNS \MHCOM \LastInWindow]
         [COND
            ((NULL \MHWINDOW)
             (AND BACKGROUNDCURSORINFN (GETD BACKGROUNDCURSORINFN)
                  (\PROTECTED.APPLY BACKGROUNDCURSORINFN)))
            ((SETQ \MHCOM (fetch (WINDOW CURSORINFN) of \MHWINDOW))
             (ERSETQ (DOUSERFNS \MHCOM \MHWINDOW]
         (SETQ \LastInWindow \MHWINDOW)
         (COND
            ((EQ LASTMOUSEBUTTONS 0)
                                        ; Don't show transition to UP as we come out of another window
             (SETQ \LastWindowButtons LASTMOUSEBUTTONS)
             (RETURN]
        (T ;; Mouse is down and had not changed.  Nothing interesting to do -- act as if we are still in old
           ;; window
           (RETURN]

;;; We have now taken care of window changing stuff, and \MHWINDOW = \LastInWindow -- Now take care of button transitions

(COND
   ([AND (LASTMOUSESTATE (ONLY RIGHT))
         (NOT (AND \MHWINDOW (fetch (WINDOW RIGHTBUTTONFN) of \MHWINDOW]
                                        ; Right button is down.  This does window com unless overridden
                                        ; by RIGHTBUTTONFN
                                        ; this is separated out from the process stuff below so that
                                        ; window commands don't grab the tty.
    (COND
       ((AND (NULL \MHWINDOW)
              (.COPYKEYDOWNP.)
             BACKGROUNDCOPYRIGHTBUTTONEVENTFN
             (GETD BACKGROUNDCOPYRIGHTBUTTONEVENTFN))
                                        ; check for copy key.
        (\PROTECTED.APPLY BACKGROUNDCOPYRIGHTBUTTONEVENTFN))
       (T                               ; if \MHWINDOW is NIL, this does background menu stuff.
          (DOWINDOWCOM \MHWINDOW)))
    ;; this attempts to prevent the cursorout fn and scrolling fns from being called if the \LastInWindow was
    ;; closed.
    (OR (OPENWP \LastInWindow)
        (SETQ \LastInWindow NIL))
    (GO RESETBUTTONS))
   [\MHWINDOW                           ; Mouse is in a window, look for button change or cursor moving
                                        ; fn.
          (COND
             ((NEQ LASTMOUSEBUTTONS \LastWindowButtons)
                                        ; Button change within same window
              (COND
                 ((AND (LASTMOUSESTATE (NOT UP))
                       (SETQ \MHPROCESS (WINDOWPROP \MHWINDOW 'PROCESS))
                       (NOT (TTY.PROCESSP \MHPROCESS))
                       (NOT (.COPYKEYDOWNP.))
                       (SETQ \MHCOM (fetch (WINDOW WINDOWENTRYFN) of \MHWINDOW)))
                                        ; make sure that if this window has a process that that process
                                        ; has the tty.
                  (ERSETQ (DOUSERFNS \MHCOM \MHWINDOW))
                  (GO RESETBUTTONS))
                 ([SETQ \MHCOM (COND
                                  [(AND (.COPYKEYDOWNP.)
                                        (WINDOWPROP \MHWINDOW
                                              'COPYBUTTONEVENTFN]
                                  ((LASTMOUSESTATE (ONLY RIGHT))
                                   (fetch (WINDOW RIGHTBUTTONFN) of \MHWINDOW))
                                  (T (fetch (WINDOW BUTTONEVENTFN) of \MHWINDOW]
                  (\PROTECTED.APPLY \MHCOM \MHWINDOW)
                  (GO RESETBUTTONS)))
              (SETQ \LastWindowButtons LASTMOUSEBUTTONS))
             ((SETQ \MHCOM (fetch (WINDOW CURSORMOVEDFN) of \MHWINDOW))
                                        ; cursor must have moved.
              (ERSETQ (DOUSERFNS \MHCOM \MHWINDOW]
    (T                                  ; look for button change or cursor moving in background
       (COND
          ((NEQ LASTMOUSEBUTTONS \LastWindowButtons)
                                        ; Button change within background
           (COND
```

```
                                        ((AND (NULL \MHWINDOW)
                                              (.COPYKEYDOWNP.)
                                             BACKGROUNDCOPYBUTTONEVENTFN
                                              (GETD BACKGROUNDCOPYBUTTONEVENTFN))
                                         (\PROTECTED.APPLY BACKGROUNDCOPYBUTTONEVENTFN)
                                         (GO RESETBUTTONS))
                                        ((AND BACKGROUNDBUTTONEVENTFN (GETD BACKGROUNDBUTTONEVENTFN))
                                         (\PROTECTED.APPLY BACKGROUNDBUTTONEVENTFN)
                                         (GO RESETBUTTONS)))
                                       (SETQ \LastWindowButtons LASTMOUSEBUTTONS))
                                      ((AND BACKGROUNDCURSORMOVEDFN (GETD BACKGROUNDCURSORMOVEDFN))
                                                                    ; cursor must have moved.
                                       (\PROTECTED.APPLY BACKGROUNDCURSORMOVEDFN)
                                       (GO RESETBUTTONS))
                                      ([AND BACKGROUNDCURSOREXITFN (OR (EQ LASTMOUSEX 0)
                                                                       (EQ LASTMOUSEX (SUB1 \CURSORDESTWIDTH]
                                                                    ; cursor must have moved.
                                       (ERSETQ (APPLY* BACKGROUNDCURSOREXITFN))
                                       (GETMOUSESTATE)
                                       (replace (POSITION XCOORD) of \LastCursorPosition with LASTMOUSEX)
                                       (replace (POSITION YCOORD) of \LastCursorPosition with LASTMOUSEY]
                    (RETURN)
                RESETBUTTONS
                                                                    ; Look at mouse again, since user fn may have waited for mouse
                                                                    ; to come up
                    (GETMOUSESTATE)
                    (SETQ \LastWindowButtons LASTMOUSEBUTTONS)
                    (RETURN])
```

(\PROTECTED.APPLY

```
  [LAMBDA (FN WINDOW)                                   (* bvm%: "20-Apr-84 16:20")
    (DECLARE (LOCALVARS . T))

;;; Apply FN to WINDOW under an errorset to trap errors

    (ERSETQ (APPLY* FN WINDOW])
```

(DOWINDOWCOM

```
  [LAMBDA ARGS                                          ; Edited 25-Nov-86 17:30 by hdj
    ;; the button handler for the window system.  if no arg, just return.
    (if (NEQ ARGS 0)
        then (LET ((WINDOW (ARG ARGS 1)))
                  (COND
                     [(type? WINDOW WINDOW)
                      (PROG ($$VAR)
                            (COND
                               ((SETQ $$VAR (WINDOWPROP WINDOW 'DOWINDOWCOMFN))
                                (RETURN (APPLY* $$VAR WINDOW)))
                               (T (\CHECKCARET WINDOW)
                                  (TOTOPW WINDOW)
                                  (RETURN (COND
                                            ([SETQ $$VAR
                                              (MENU (COND
                                                       ((type? MENU WindowMenu)
                                                        WindowMenu)
                                                       (T (SETQ WindowMenu
                                                            (create MENU
                                                                    ITEMS _ WindowMenuCommands
                                                                    CHANGEOFFSETFLG _ 'Y
                                                                    MENUOFFSET _
                                                                    (create POSITION
                                                                            XCOORD _ -1
                                                                            YCOORD _ 0)
                                                                    WHENHELDFN _ (FUNCTION PPROMPT3)
                                                                    WHENUNHELDFN _ (FUNCTION CLRPROMPT)
                                                                    CENTERFLG _ T]
                                             (APPLY* $$VAR WINDOW)
                                             T]
                     ((NULL WINDOW)
                      (DOBACKGROUNDCOM])
```

(DOBACKGROUNDCOM

```
  [LAMBDA NIL                                           ; Edited 10-Mar-92 15:48 by jds
    ;; Bring up the background menu.
    (PROG (FORM)
          (AND (OR BackgroundMenu BackgroundMenuCommands)
               [SETQ FORM (MENU (COND
                                  ((type? MENU BackgroundMenu)
                                   BackgroundMenu)
                                  (T (SETQ BackgroundMenu (create MENU
                                                                  ITEMS _ BackgroundMenuCommands
                                                                  CHANGEOFFSETFLG _ 'Y
```

```
                                                                MENUOFFSET _
                                                                (create POSITION
                                                                       XCOORD _ -1
                                                                       YCOORD _ 0)
                                                                WHENHELDFN _ (FUNCTION PPROMPT3)
                                                                WHENUNHELDFN _ (FUNCTION CLRPROMPT)
                                                                CENTERFLG _ T]
              (ERSETQ (EVAL FORM])
```

(DEFAULT.BACKGROUND.COPYFN

```
  [LAMBDA NIL                                                   (* bvm%: "17-Oct-85 00:02")
```

;;; the default function called when the right button goes down in the background and the copy key is held down.

```
    (COND
       ((AND (MOUSESTATE (NOT UP))
             BackgroundCopyMenuCommands)
        (LET [(FORM (MENU (COND
                          ((type? MENU BackgroundCopyMenu)
                           BackgroundCopyMenu)
                          (T (SETQ BackgroundCopyMenu (create MENU
                                                              ITEMS _ BackgroundCopyMenuCommands
                                                              CHANGEOFFSETFLG _ 'Y
                                                              MENUOFFSET _
                                                              (create POSITION
                                                                     XCOORD _ -1
                                                                     YCOORD _ 0)
                                                              CENTERFLG _ T]
             (AND FORM (ERSETQ (EVAL FORM])

)

(RPAQQ BackgroundCopyMenu NIL)

(RPAQ? BackgroundCopyMenuCommands NIL)

(DEFINEQ
```

(BURYW

```
  [LAMBDA (WINDOW)                                              ; Edited 2-Feb-94 13:13 by sybalsky:mv:envos
    (WINDOWOP 'BURYWFN (fetch (WINDOW SCREEN) of (SETQ WINDOW (\INSUREWINDOW WINDOW)))
           WINDOW])
```

(CLEARW

```
  [LAMBDA (WINDOW)                                              ; Edited 8-Dec-93 18:10 by nilsson
```

;; clears a window to its background shade, resets its offsets to 0,0 in the lower left corner and resets the position to the upper left {first line of text}.

```
    (SETQ WINDOW (\INSUREWINDOW WINDOW))
    (DSPRESET (fetch (WINDOW DSP) of WINDOW])
```

(CLOSEW

```
  [LAMBDA (WINDOW)                                              ; Edited 25-Apr-94 10:08 by sybalsky
```

;; closes a window. saves the current state in the WINDOW and allow it to be reOPENWed.

;; Returns T if the window closed OK (and was previously open), as a signal to \INTERACTIVE.CLOSEW.

```
    (SETQ WINDOW (\INSUREWINDOW WINDOW))
    (COND
       ((OPENWP WINDOW)
        (COND
           ((AND (\OKTOCLOSEW WINDOW)
                 (OPENWP WINDOW))                               ; one of the CLOSEFNs may have closed the window. If so,
                                                                ; don't reopen it.
            (WINDOWOP 'CLOSEWFN (fetch (WINDOW SCREEN) of WINDOW)
                   WINDOW)
            T])
```

(\CLOSEW1

```
  [LAMBDA (WINDOW)                                              ; Edited 25-Apr-94 10:08 by sybalsky
```

;; actually does the closing operation. Is used by SHRINKW to avoid the CLOSEFN mechanism.

```
    (WINDOWOP 'CLOSEWFN (fetch (WINDOW SCREEN) of (SETQ WINDOW (\INSUREWINDOW WINDOW)))
           WINDOW])
```

(\OKTOCLOSEW

```
  [LAMBDA (WINDOW)                                              (* rrb "14-JUN-82 12:40")
```

;; calls the windows closefns. Returns T if it is ok to close the window.

```
    (COND
       ((EQ (DOUSERFNS (fetch (WINDOW CLOSEFN) of WINDOW)
                   WINDOW T)
            'DON'T)
        NIL)
```

```
       (T WINDOW])
```

(\INTERACTIVE.CLOSEW

```
[LAMBDA (WINDOW)                                        ; Edited  4-Mar-88 09:52 by jds
```

;; Interactive version of CLOSEW -- used by the window-command menu. If the window can't be closed, this function prints a message saying so.

```
(LET ((CLOSEFN (fetch (WINDOW CLOSEFN) of WINDOW)))
     (COND
        ((OR (EQ 'DON'T CLOSEFN)
             (AND (LISTP CLOSEFN)
                  (FMEMB 'DON'T CLOSEFN)))          ; The window has DON'T as one of its CLOSEFNs.  Tell the guy
                                                    ; the window isn't closeable.
         (PROMPTPRINT "This window cannot be closed."))
        (T                                          ; Try closing it.
           (CLOSEW WINDOW)
           T])
```

(OPENW

```
[LAMBDA (WINDOW)                                        ; Edited 25-Apr-94 10:12 by sybalsky
```

;; Generic OPENW method

```
(SETQ WINDOW (\INSUREWINDOW WINDOW))
(COND
   ((OPENWP WINDOW)                                     ; used to bring the window to top but doesn't since TOTOPW has
                                                        ; been documented.
    NIL)
   (T (PROG [(USEROPENFN (WINDOWPROP WINDOW 'OPENFN]
            (COND
               ((\USERFNISDON'T USEROPENFN)             ; one of the OPENFNs is DON'T
                NIL)
               (T                                       ; open it by putting it on top and swapping its bits in
                  (\OPENW1 WINDOW)                      ; call the openfns after the window has been opened.
                  (DOUSERFNS USEROPENFN WINDOW)
                  (RETURN WINDOW])
```

(DOUSERFNS

```
[LAMBDA (FNLST WINDOW CHECKFORDON'TFLG)                 (* rrb "20-Mar-84 16:18")
```

;; applys a list of user functins and If CHECKFORDON'TFLG is non-NIL, it stops if don't is returned as one of the values and returns DON'T

```
(DECLARE (GLOBALVARS LAMBDASPLST))
(COND
   [(OR (NLISTP FNLST)
        (FMEMB (CAR FNLST)
               LAMBDASPLST))
    (COND
       ((AND CHECKFORDON'TFLG (EQ FNLST 'DON'T))
        'DON'T)
       (FNLST (AND (EQ (APPLY* FNLST WINDOW)
                       'DON'T)
                   'DON'T]
   ((AND CHECKFORDON'TFLG (FMEMB 'DON'T FNLST))
    'DON'T)
   ((for USERFN in FNLST when (EQ (APPLY* USERFN WINDOW)
                                  'DON'T)
       do                                               ; return if any of the openfns says don't
          (AND CHECKFORDON'TFLG (RETURN 'DON'T])
```

(DOUSERFNS2

```
[LAMBDA (FNLST WINDOW ARG1 ARG2 ARG3)                   (* rrb " 3-Jul-84 15:59")
```

;; applys a list of user functions to two arguments. This is used by SHAPEW.

```
(DECLARE (GLOBALVARS LAMBDASPLST))
(COND
   [(OR (NLISTP FNLST)
        (FMEMB (CAR FNLST)
               LAMBDASPLST))
    (COND
       (FNLST (APPLY* FNLST WINDOW ARG1 ARG2 ARG3]
   ((for USERFN in FNLST do (APPLY* USERFN WINDOW ARG1 ARG2 ARG3])
```

(\USERFNISDON'T

```
[LAMBDA (USERFN)                                        (* rrb "18-JUN-82 12:16")
```

;; determines if one of the userfunction is DON'T

```
(COND
   ((NLISTP USERFN)
    (EQ USERFN 'DON'T))
   (T (FMEMB 'DON'T USERFN])
```

(\OPENW1

```
[LAMBDA (WINDOW)                                        ; Edited 25-Apr-94 10:12 by sybalsky
```

```
;;; Open a closed window without running the OPENW methods.
    ;; If already open, punt.
    (WINDOWOP 'OPENWFN (fetch (WINDOW SCREEN) of (SETQ WINDOW (\INSUREWINDOW WINDOW)))
          WINDOW])
```

(CREATEW

```
[LAMBDA (REGION TITLE BORDERSIZE NOOPENFLG PROPS)            ; Edited  7-Jan-94 11:16 by nilsson
    ;; Generic CREATEW function.
    (LET (SCREEN REG)
         (COND
            [(NULL REGION)
             (PROMPTPRINT "Specify region for window")
             (COND
                (TITLE (PROMPTPRINT " %"" TITLE "%"")))
             (SETQ REGION (GETSCREENREGION MinWindowWidth MinWindowHeight))
             (SETQ SCREEN (fetch (SCREENREGION SCREEN) of REGION))
             (SETQ REG (COPY (fetch (SCREENREGION REGION) of REGION]
            ((type? REGION REGION)
             (SETQ SCREEN \CURSORSCREEN)                     ; Protect against user smashing REGION later on.
             (SETQ REG (COPY REGION)))
            [(type? SCREENREGION REGION)
             (SETQ SCREEN (fetch (SCREENREGION SCREEN) of REGION))
             (SETQ REG (COPY (fetch (SCREENREGION REGION) of REGION]
            ((DISPLAYSTREAMP REGION)
             (HELP "DISPLAYSTREAMs as REGIONS no longer supported."))
            (T (ERROR "Not a region" REG)))
         (\CREATEW1 SCREEN REG TITLE BORDERSIZE NOOPENFLG PROPS])
```

(CREATEW1

```
[LAMBDA (REGION TITLE BORDERSIZE NOOPENFLG PROPS OLDWINDOW)    ; Edited 27-Dec-93 18:41 by nilsson
    ;; To reuse an old window structure, you have to specify the REGION and OLDWINDOW
    (LET [SCREEN REG DSP DISPLAYDATA TITLEHEIGHT WINDOW (BORDERSIZE (COND
                                                                       ((NUMBERP BORDERSIZE)
                                                                        (ABS BORDERSIZE))
                                                                       ((NUMBERP WBorder)
                                                                        (ABS WBorder))
                                                                       (T 2]
         (COND
            [(NULL REGION)
             (PROMPTPRINT "Specify region for window")
             (COND
                (TITLE (PROMPTPRINT " %"" TITLE "%"")))
             (SETQ REGION (GETSCREENREGION MinWindowWidth MinWindowHeight))
             (SETQ SCREEN (fetch (SCREENREGION SCREEN) of REGION))
             (SETQ REG (COPY (fetch (SCREENREGION REGION) of REGION]
            ((type? REGION REGION)
             (SETQ SCREEN \CURSORSCREEN)                     ; Protect against user smashing REGION later on.
             (SETQ REG (COPY REGION)))
            [(type? SCREENREGION REGION)
             (SETQ SCREEN (fetch (SCREENREGION SCREEN) of REGION))
             (SETQ REG (COPY (fetch (SCREENREGION REGION) of REGION]
            ((DISPLAYSTREAMP REGION)
             (HELP "DISPLAYSTREAMs as REGIONS no longer supported."))
            (T (ERROR "Not a region" REG)))
         (COND
            ((NOT (IGREATERP (IMIN (fetch (REGION WIDTH) of REG)
                                   (fetch (REGION HEIGHT) of REG))
                             (UNFOLD BORDERSIZE 2)))
             (ERROR "Region too small to use as a window" REGION)))
         (SETQ WINDOW (WINDOWOP 'CREATEWFN SCREEN REG TITLE BORDERSIZE NOOPENFLG PROPS OLDWINDOW))
         (COND
            ((NOT NOOPENFLG)
             (OPENW WINDOW)))
        WINDOW])
```

(\CREATEW1

```
[LAMBDA (SCREEN REGION TITLE BORDERSIZE NOOPENFLG PROPS OLDWINDOW)
                                                             ; Edited  7-Jan-94 10:57 by nilsson
    ;; To reuse an old window structure, you have to specify the REGION and OLDWINDOW
    (LET [DSP DISPLAYDATA TITLEHEIGHT WINDOW (BORDERSIZE (COND
                                                            ((NUMBERP BORDERSIZE)
                                                             (ABS BORDERSIZE))
                                                            ((NUMBERP WBorder)
                                                             (ABS WBorder))
                                                            (T 2]
         (COND
            ((NOT (IGREATERP (IMIN (fetch (REGION WIDTH) of REGION)
                                   (fetch (REGION HEIGHT) of REGION))
                             (UNFOLD BORDERSIZE 2)))
```

```
            (ERROR "Region too small to use as a window" REGION)))
         (SETQ WINDOW (WINDOWOP 'CREATEWFN SCREEN REGION TITLE BORDERSIZE NOOPENFLG PROPS OLDWINDOW))
         (COND
            ((NOT NOOPENFLG)
             (OPENW WINDOW)))
         WINDOW])
```

(OPENDISPLAYSTREAM

```
[LAMBDA (FILE OPTIONS)                                          (* hdj "17-Jan-86 14:47")
  (GETSTREAM (CREATEW (LISTGET OPTIONS 'REGION)
                      (COND
                         ((EQ FILE '{LPT})
                          "Display image stream")
                         (T FILE])
```

(MOVEW

```
[LAMBDA (WINDOW POSorX Y FORCE)                                 ; Edited  5-Jan-94 16:08 by nilsson
  (WINDOWOP 'MOVEWFN (fetch (WINDOW SCREEN) of WINDOW)
         WINDOW POSorX Y FORCE])
```

(PPROMPT3

```
[LAMBDA (ITEM)                                                  (* rrb "17-NOV-81 12:15")
```

;; prints the third element of ITEM in the prompt window. This is the default WHENHELDFN for MENUs.

```
(COND
   ((AND (LISTP ITEM)
         (CADDR ITEM))
    (PROMPTPRINT (CADDR ITEM])
```

(\ONSCREENCLIPPINGREGION

```
[LAMBDA (WIN)                                                   (* kbr%: "26-Mar-85 23:34")
```

;; returns a region which is the part of the windows clipping region that is on the screen.

```
(INTERSECTREGIONS (DSPCLIPPINGREGION NIL WIN)
       (\DSPUNTRANSFORMREGION (fetch (SCREEN SCREGION) of (fetch (WINDOW SCREEN) of WIN))
              (fetch (STREAM IMAGEDATA) of (WINDOWPROP WIN 'DSP])
```

(RELMOVEW

```
[LAMBDA (WINDOW POS)                                            ; Edited  2-Feb-94 13:12 by sybalsky:mv:envos
  (WINDOWOP 'RELMOVEWFN (fetch (WINDOW SCREEN) of (SETQ WINDOW (\INSUREWINDOW WINDOW)))
         WINDOW POS])
```

(SHAPEW

```
[LAMBDA (WINDOW NEWREGION MAINONLYFLG)                          ; Edited 24-Jan-97 10:53 by rmk:
                                                                ; Edited 24-Sep-92 12:30 by jds
```

;; entry that shapes a window checking the userfns for DON'T and interacting to get a region if necessary. This also checks for a user function to
;; do the actual reshaping. look for a function on windowprop INITCORNERSFN, which will take the window and return the initcorners for the
;; window, to be passed to getregion. MAINONLYFLG is a flag passed to any DOSHAPEFN (especially for RESHAPEALLWINDOWS in
;; ATTACHEDWINDOW). It indicates that the new region is to be allocated entirely to the main window.

```
(SETQ WINDOW (\INSUREWINDOW WINDOW))
(PROG ((OLDSIZE (WINDOWPROP WINDOW 'REGION))
       NEWSIZE)
      (COND
         ((\USERFNISDON'T (fetch (WINDOW RESHAPEFN) of WINDOW))
                                                                ; don't allow the window to be reshaped.
          (PROMPTPRINT "This window cannot be reshaped.")
          (RETURN NIL)))
      (SETQ NEWSIZE (MINIMUMWINDOWSIZE WINDOW))                 ; Start with the minimum allowable size.
      [SETQ NEWSIZE (COND
                       (NEWREGION                               ; An explicit new region was specified; make sure it's big enough.
                              (COND
                                 [(OR (LESSP (fetch (REGION WIDTH) of NEWREGION)
                                             (CAR NEWSIZE))
                                      (LESSP (fetch (REGION HEIGHT) of NEWREGION)
                                             (CDR NEWSIZE)))    ; given a region that is too small, so expand the width and height
                                                                ; to at least the minima.
                                  (CREATEREGION (fetch (REGION LEFT) of NEWREGION)
                                         (fetch (REGION BOTTOM) of NEWREGION)
                                         (IMAX (CAR NEWSIZE)
                                               (fetch (REGION WIDTH) of NEWREGION))
                                         (IMAX (CDR NEWSIZE)
                                               (fetch (REGION HEIGHT) of NEWREGION]
                                  (T NEWREGION)))
                       ((WINDOWPROP WINDOW 'INITCORNERSFN)      ; There's an INITCORNERSFN.  Fire it up and prompt the user
                                                                ; for a  new shape.
                        (GETREGION (CAR NEWSIZE)
                               (CDR NEWSIZE)
                               (WINDOWREGION WINDOW 'SHAPEW)
                               (fetch (WINDOW NEWREGIONFN) of WINDOW)
                               WINDOW
```

```
                                (APPLY* (WINDOWPROP WINDOW 'INITCORNERSFN)
                                       WINDOW)))
                            (T                                          ; Just go prompt the user for a new shape.
                               (GETREGION (CAR NEWSIZE)
                                      (CDR NEWSIZE)
                                      (WINDOWREGION WINDOW 'SHAPEW)
                                      (fetch (WINDOW NEWREGIONFN) of WINDOW)
                                      WINDOW]
        (RETURN (COND
                   ((EQUAL NEWSIZE OLDSIZE)
                    ;; if same size and place as before, do nothing
                    NIL)
                   ((AND (EQ (fetch (REGION WIDTH) of NEWSIZE)
                             (fetch (REGION WIDTH) of OLDSIZE))
                         (EQ (fetch (REGION HEIGHT) of NEWSIZE)
                             (fetch (REGION HEIGHT) of OLDSIZE)))
                    ;; if same width and height, then optimize to a move
                    (MOVEW WINDOW (fetch (REGION LEFT) of NEWSIZE)
                           (fetch (REGION BOTTOM) of NEWSIZE)))
                   (T ;; do the shape, checking for a doshapefn
                      (APPLY* (OR (WINDOWPROP WINDOW 'DOSHAPEFN)
                                  'SHAPEW1)
                              WINDOW
                              (COPYALL NEWSIZE)
                              MAINONLYFLG])
```

(SHAPEW1

```
  [LAMBDA (WINDOW REGION)                                               (* kbr%: "25-Jan-86 15:08")
    ;; entry for shaping a window that does the reshape without checking for a user function.
    (DECLARE (LOCALVARS . T))
    (SETQ WINDOW (\INSUREWINDOW WINDOW))
    (OR (REGIONP REGION)
        (\ILLEGAL.ARG REGION))
    (PROG ((OLDREGION (fetch (WINDOW REG) of WINDOW))
           (OLDCLIPREG (DSPCLIPPINGREGION NIL (fetch (WINDOW DSP) of WINDOW)))
           (WBORDER (fetch (WINDOW WBORDER) of WINDOW))
           SCREEN NUSAV NOWOPEN?)
          (SETQ SCREEN (fetch (WINDOW SCREEN) of WINDOW))
          [SETQ NUSAV (BITMAPCREATE (fetch (REGION WIDTH) of REGION)
                                    (fetch (REGION HEIGHT) of REGION)
                                    (fetch (BITMAP BITMAPBITSPERPIXEL) of (fetch (SCREEN SCDESTINATION) of SCREEN]
          (UNINTERRUPTABLY
              (COND
                 ((OPENWP WINDOW)                                       ; notice whether window is open or not to call OPENFNs only if
                                                                        ; not now open.
                  (SETQ NOWOPEN? T)
                  (\CLOSEW1 WINDOW)))                                   ; Save window image
              (replace (WINDOW REG) of WINDOW with REGION)
              [replace (WINDOW SAVE) of WINDOW with (PROG1 NUSAV
                                                           (SETQ NUSAV (fetch (WINDOW SAVE) of WINDOW)))]
              (ADVISEWDS WINDOW OLDREGION)
              (SHOWWFRAME WINDOW)
              (COND
                 (NOWOPEN? (\OPENW1 WINDOW))
                 (T (OPENW WINDOW))))
          (DOUSERFNS2 (OR (fetch (WINDOW RESHAPEFN) of WINDOW)
                          (FUNCTION RESHAPEBYREPAINTFN))
                 WINDOW NUSAV (create REGION
                                      LEFT _ WBORDER
                                      BOTTOM _ WBORDER
                                      WIDTH _ (fetch (REGION WIDTH) of OLDCLIPREG)
                                      HEIGHT _ (fetch (REGION HEIGHT) of OLDCLIPREG))
                 OLDREGION)
          (RETURN WINDOW])
```

(\SHAPEW2

```
  [LAMBDA (WINDOW REGION)                                               ; Edited  6-Jan-87 13:56 by woz
```

;;; entry for shaping a window that does the reshape without checking for a user function, and without running the openfn.

```
    (DECLARE (LOCALVARS . T))
    (SETQ WINDOW (\INSUREWINDOW WINDOW))
    (OR (REGIONP REGION)
        (\ILLEGAL.ARG REGION))
    (PROG ((OLDREGION (fetch (WINDOW REG) of WINDOW))
           (OLDCLIPREG (DSPCLIPPINGREGION NIL (fetch (WINDOW DSP) of WINDOW)))
           (WBORDER (fetch (WINDOW WBORDER) of WINDOW))
           SCREEN NUSAV)
          (SETQ SCREEN (fetch (WINDOW SCREEN) of WINDOW))
          [SETQ NUSAV (BITMAPCREATE (fetch (REGION WIDTH) of REGION)
                                    (fetch (REGION HEIGHT) of REGION)
```

```
                        (fetch (BITMAP BITMAPBITSPERPIXEL) of (fetch (SCREEN SCDESTINATION) of SCREEN]
    (UNINTERRUPTABLY
        (COND
           ((OPENWP WINDOW)                                    ; close open window before changing region
            (\CLOSEW1 WINDOW)))

        ;; Save window image

        (replace (WINDOW REG) of WINDOW with REGION)
        [replace (WINDOW SAVE) of WINDOW with (PROG1 NUSAV
                                                    (SETQ NUSAV (fetch (WINDOW SAVE) of WINDOW)))]
        (ADVISEWDS WINDOW OLDREGION)
        (SHOWWFRAME WINDOW)                                    ; open without openfn
        (\OPENW1 WINDOW))
    (DOUSERFNS2 (OR (fetch (WINDOW RESHAPEFN) of WINDOW)
                    (FUNCTION RESHAPEBYREPAINTFN))
          WINDOW NUSAV (CREATEREGION WBORDER WBORDER (fetch (REGION WIDTH) of OLDCLIPREG)
                                (fetch (REGION HEIGHT) of OLDCLIPREG))
          OLDREGION)
    (RETURN WINDOW])
```

(RESHOWBORDER

```
[LAMBDA (BORDER WINDOW)                                        (* rrb "15-JUN-83 14:46")

    ;; updates a windows display with a new border            ; if the border is the same, don't change anything.
    (OR (EQ BORDER (fetch (WINDOW WBORDER) of WINDOW))
        (\RESHOWBORDER1 BORDER (fetch (WINDOW WBORDER) of WINDOW)
              WINDOW])
```

(\RESHOWBORDER1

```
[LAMBDA (NEWBORDER OLDBORDER WINDOW)                           (* kbr%: "25-Jan-86 15:13")

    ;; redisplays the border of a window.  Is called by RESHOWBORDER and RESHOWTITLE.  It doesn't check for equality between the new and old
    ;; borders because it is also used when a title is added or deleted.

    (PROG ((REGION (fetch (WINDOW REG) of WINDOW))
           (OLDSAVE (fetch (WINDOW SAVE) of WINDOW))
           NUSAV DELTA NUWIDTH NUHEIGHT)
          (SETQ DELTA (IDIFFERENCE NEWBORDER OLDBORDER))
          (SETQ NUWIDTH (IPLUS (fetch (REGION WIDTH) of REGION)
                               (ITIMES DELTA 2)))
          [SETQ NUHEIGHT (IDIFFERENCE (IPLUS (fetch (REGION HEIGHT) of (DSPCLIPPINGREGION NIL (fetch (WINDOW DSP)
                                                                                                    of WINDOW)))
                                             (ITIMES NEWBORDER 2))
                                      (COND
                                         [(fetch (WINDOW WTITLE) of WINDOW)
                                          (DSPLINEFEED NIL (fetch (SCREEN SCTITLEDS) of (fetch (WINDOW SCREEN)
                                                                                               of WINDOW]
                                      (T 0]
          (SETQ NUSAV (BITMAPCREATE NUWIDTH NUHEIGHT (fetch (BITMAP BITMAPBITSPERPIXEL) of OLDSAVE)))
          (.WHILE.TOP.DS. WINDOW                               ; Save window image
                 (\SW2BM (fetch (SCREEN SCDESTINATION) of (fetch (WINDOW SCREEN) of WINDOW))
                        REGION
                        (fetch (WINDOW SAVE) of WINDOW)
                        NIL)                                   ; put new save image into window
                 (replace (WINDOW SAVE) of WINDOW with NUSAV)
                 (replace (WINDOW WBORDER) of WINDOW with NEWBORDER)
                                                               ; create a region that coresponds to the old region with the new
                                                               ; border.
                 (replace (WINDOW REG) of WINDOW with (create REGION
                                                              LEFT _ (IDIFFERENCE (fetch (REGION LEFT) of REGION)
                                                                                  DELTA)
                                                              BOTTOM _ (IDIFFERENCE (fetch (REGION BOTTOM)
                                                                                       of REGION)
                                                                                    DELTA)
                                                              WIDTH _ NUWIDTH
                                                              HEIGHT _ NUHEIGHT))
                 (UPDATE/SCROLL/REG WINDOW)                    ; draw border in the new image.
                 (SHOWWFRAME WINDOW)                           ; copy the visible part from the old image into the new one.
                 (BITBLT OLDSAVE OLDBORDER OLDBORDER NUSAV NEWBORDER NEWBORDER (IDIFFERENCE (fetch (BITMAP
                                                                                                   BITMAPWIDTH
                                                                                                   )
                                                                                              of OLDSAVE)
                                                                                       (ITIMES 2 OLDBORDER))
                        (fetch (REGION HEIGHT) of (DSPCLIPPINGREGION NIL (fetch (WINDOW DSP) of WINDOW)))
                        'INPUT
                        'REPLACE)                              ; put the new image up on the screen.
                 (\SW2BM (fetch (SCREEN SCDESTINATION) of (fetch (WINDOW SCREEN) of WINDOW))
                        (fetch (WINDOW REG) of WINDOW)
                        (fetch (WINDOW SAVE) of WINDOW)
                        NIL])
```

(TRACKW

```
[LAMBDA (WINDOW)                                               (* rrb " 9-MAR-82 14:28")

    ;; causes a window to follow the cursor.  found to be not useful but very pretty for small windows.

    (SETQ WINDOW (\INSUREWINDOW WINDOW))
```

```
    (RESETFORM (CURSOR CROSSHAIRS)
           (TOTOPW WINDOW)
           (until (MOUSESTATE (NOT UP)))
           (CURSOR LOCKEDSPOT)
           (bind (DX _ (IDIFFERENCE (fetch (REGION LEFT) of (fetch (WINDOW REG) of WINDOW))
                                    LASTMOUSEX))
                 (DY _ (IDIFFERENCE (fetch (REGION BOTTOM) of (fetch (WINDOW REG) of WINDOW))
                                    LASTMOUSEY))
              until (MOUSESTATE UP) do (MOVEW WINDOW (create POSITION
                                                             XCOORD _ (IPLUS LASTMOUSEX DX)
                                                             YCOORD _ (IPLUS LASTMOUSEY DY])
```

(SNAPW

```
  [LAMBDA NIL                                                    ; Edited 21-Jul-92 17:12 by jds
```

;; makes a new window which is a copy of the bits underneath the REGION read from the user.

```
    (PROG (SCREENREGION SCREEN REGION NEWWINDOW)
          (SETQ SCREENREGION (GETSCREENREGION 30 20))
          (SETQ SCREEN (fetch (SCREENREGION SCREEN) of SCREENREGION))
          (SETQ REGION (fetch (SCREENREGION REGION) of SCREENREGION))
          (SETQ NEWWINDOW (CREATEW (create SCREENREGION
                                           SCREEN _ SCREEN
                                           REGION _ (GROW/REGION REGION WBorder))
                                   NIL NIL T))                   ; keep it closed so it doesn't cover any of the bits it is to copy.
                                                                 ; put existing screen bits from SAVE.
          (BITBLT (fetch (SCREEN SCDESTINATION) of SCREEN)
                  (fetch (REGION LEFT) of REGION)
                  (fetch (REGION BOTTOM) of REGION)
                  (fetch (WINDOW SAVE) of NEWWINDOW)
                  WBorder WBorder (fetch (REGION WIDTH) of REGION)
                  (fetch (REGION HEIGHT) of REGION)
                  'INPUT
                  'REPLACE)
          (WINDOWPROP NEWWINDOW 'TYPE :SNAP)                     ; MARK THIS AS A SNAP WINDOW.
          (OPENW NEWWINDOW)
          (MOVEW NEWWINDOW)
          (RETURN NEWWINDOW])
```

(WINDOWREGION

```
  [LAMBDA (WINDOW COM)                                           (* jow "26-Aug-85 13:48")
```

;; gets the region that a window wants to consider to be its. COM can be a window com used to help calculate the region, ie for shaping or
;; moving...

```
    (PROG (FN)
          (RETURN (COND
                     ((SETQ FN (WINDOWPROP WINDOW 'CALCULATEREGIONFN))
                      (APPLY* FN WINDOW COM))
                     (T (WINDOWPROP WINDOW 'REGION])

)

(DEFINEQ
```

(MINIMUMWINDOWSIZE

```
  [LAMBDA (WINDOW)                                               (* rrb "20-NOV-83 12:06")
```

;; returns the minimum extent of a window

```
    (PROG [(EXT (WINDOWPROP WINDOW 'MINSIZE]
          [COND
             [(NULL EXT)
              (SETQ EXT (CONS MinWindowWidth (HEIGHTIFWINDOW (FONTPROP WINDOW 'HEIGHT)
                                                             (WINDOWPROP WINDOW 'TITLE]
             ((LITATOM EXT)
              (SETQ EXT (APPLY* EXT WINDOW]
          [COND
             [(AND (NUMBERP (CAR EXT))
                   (NUMBERP (CDR EXT]
             (T (SETQ EXT (ERROR "Illegal extent property" EXT]
          (RETURN EXT])

)

(RPAQ? BACKGROUNDCURSORINFN )

(RPAQ? BACKGROUNDBUTTONEVENTFN )

(RPAQ? BACKGROUNDCURSOROUTFN )

(RPAQ? BACKGROUNDCURSORMOVEDFN )

(RPAQ? BACKGROUNDCOPYBUTTONEVENTFN )

(RPAQ? BACKGROUNDCOPYRIGHTBUTTONEVENTFN (FUNCTION DEFAULT.BACKGROUND.COPYFN))

(RPAQ? BACKGROUNDCURSOREXITFN )
```

```
(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS BACKGROUNDCURSORINFN BACKGROUNDBUTTONEVENTFN BACKGROUNDCURSOROUTFN BACKGROUNDCURSORMOVEDFN
       BACKGROUNDCOPYBUTTONEVENTFN BACKGROUNDCOPYRIGHTBUTTONEVENTFN \CARET.UP BACKGROUNDCURSOREXITFN)
)

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(PUTPROPS .COPYKEYDOWNP. MACRO [NIL (OR (KEYDOWNP 'LSHIFT)
                                        (KEYDOWNP 'RSHIFT)
                                        (KEYDOWNP 'COPY])

(PUTPROPS WSOP MACRO [ARGS (LET ((METHOD (CADR (CAR ARGS)))
                                 (DISPLAY (CADR ARGS))
                                 (OTHERARGS (CDDR ARGS)))
                              `(SPREADAPPLY* (fetch (WSOPS ,METHOD) of (fetch (FDEV WINDOWOPS)
                                                                              of ,DISPLAY))
                                     ,DISPLAY
                                     ,@OTHERARGS])
)

;; END EXPORTED DEFINITIONS

(PUTPROPS WSOP ARGNAMES (METHOD DISPLAY . OTHERARGS))

(DECLARE%: EVAL@COMPILE

(RECORD WSOPS (STARTBOARD STARTCOLOR STOPCOLOR EVENTFN SENDCOLORMAPENTRY SENDPAGE PILOTBITBLT))

(RECORD WSDATA (WSDESTINATION WSREGION WSBACKGROUND WSCOLORMAP)
       (SYSTEM))
)

;; Window utilities

(DEFINEQ

(ADVISEWDS
  [LAMBDA (WINDOW OLDREG MOVEONLYFLG)                                    (* kbr%: "29-Mar-85 14:01")
    (DECLARE (LOCALVARS . T))

    ;; called whenever the dimensions of a guaranteed WINDOW change.  Updates the dependent fields in the associated DisplayStream.  Also
    ;; updates dependent fields in the WINDOW such as Scroll region.

    ;; OLDREG if given, is the region this window used to have and is used to maintain the relationship between the WINDOW coordinates and the
    ;; displaystreams when the WINDOW moves.

    ;; MOVEONLYFLG indicates that the dimensions of the region haven't changed.

    (PROG (R D WBORDERSIZE CLIPREG TWICEBORDER PROC)
          (SETQ R (fetch (WINDOW REG) of WINDOW))
          (SETQ D (fetch (WINDOW DSP) of WINDOW))
          (SETQ WBORDERSIZE (fetch (WINDOW WBORDER) of WINDOW))
          (SETQ TWICEBORDER (UNFOLD WBORDERSIZE 2))
          (COND
             (OLDREG (RELDSPXOFFSET (IDIFFERENCE (fetch (REGION LEFT) of R)
                                                 (fetch (REGION LEFT) of OLDREG))
                                    D)
                     (RELDSPYOFFSET (IDIFFERENCE (fetch (REGION BOTTOM) of R)
                                                 (fetch (REGION BOTTOM) of OLDREG))
                                    D)

                     ;; if only moving, the clipping region remains the same.  This is checked for because the height of the window title may have
                     ;; changed and this calculation results in the wrong answer.  All other calls to ADVISEWDS should repaint the border.

                     (OR MOVEONLYFLG (DSPCLIPPINGREGION [create REGION
                                                                LEFT _ (fetch (REGION LEFT)
                                                                          of (SETQ CLIPREG (DSPCLIPPINGREGION NIL D
                                                                                                              )))
                                                                BOTTOM _ (fetch (REGION BOTTOM) of CLIPREG)
                                                                WIDTH _ (IDIFFERENCE (fetch (REGION WIDTH)
                                                                                        of R)
                                                                                     TWICEBORDER)
                                                                HEIGHT _
                                                                (IPLUS (IDIFFERENCE (fetch (REGION HEIGHT)
                                                                                       of R)
                                                                                    TWICEBORDER)
                                                                       (COND
                                                                          [(fetch (WINDOW WTITLE) of WINDOW)
                                                                           (DSPLINEFEED NIL
                                                                                  (fetch (SCREEN SCTITLEDS)
                                                                                     of (fetch (WINDOW SCREEN)
                                                                                           of WINDOW]
                                                                          (T 0]
                                                        D)))
             (T (DSPXOFFSET (IPLUS (fetch (REGION LEFT) of R)
```

```
                                WBORDERSIZE)
                        D)
                (DSPYOFFSET (IPLUS (fetch (REGION BOTTOM) of R)
                                WBORDERSIZE)
                        D)
                (DSPCLIPPINGREGION [create REGION
                                           LEFT _ 0
                                           BOTTOM _ 0
                                           WIDTH _ (IDIFFERENCE (fetch (REGION WIDTH) of R)
                                                                TWICEBORDER)
                                           HEIGHT _ (IPLUS (IDIFFERENCE (fetch (REGION HEIGHT) of R)
                                                                        TWICEBORDER)
                                                           (COND
                                                              [(fetch (WINDOW WTITLE) of WINDOW)
                                                               (DSPLINEFEED NIL (fetch (SCREEN SCTITLEDS)
                                                                                       of (fetch (WINDOW SCREEN)
                                                                                                 of WINDOW]
                                                              (T 0]
                        D)))
        [COND
           ((NULL MOVEONLYFLG)                                  ; if the previous right margin was the default, change it.
            (AND (OR (NOT OLDREG)
                     (EQ (DSPRIGHTMARGIN NIL D)
                         (IDIFFERENCE (fetch (REGION WIDTH) of OLDREG)
                                      TWICEBORDER)))
                 (DSPRIGHTMARGIN (IDIFFERENCE (fetch (REGION WIDTH) of R)
                                              TWICEBORDER)
                                 D))
            (COND
               ((AND (SETQ PROC (WINDOWPROP WINDOW 'PROCESS))
                     (EQ D (PROCESS.TTY PROC)))                 ; if the window changing is a tty, set its linelength.
                [PROCESS.EVAL PROC (LIST (FUNCTION PAGEHEIGHT)
                                         (IQUOTIENT (fetch (REGION HEIGHT) of (SETQ CLIPREG (DSPCLIPPINGREGION
                                                                                                NIL D)))
                                                    (IMINUS (DSPLINEFEED NIL D]
               (PROCESS.EVAL PROC '(SETLINELENGTH))
               (IF NIL
                   THEN                                         ; try it without this.
                        (COND
                           ((EQ (PROCESSPROP PROC 'NAME)
                                'EXEC)
                            ;; in the exec process, make sure the current position is inside the new shape.  reuse variables R and
                            ;; TWICEBORDER to save binding.
                            (COND
                               ((ILESSP (SETQ R (DSPYPOSITION NIL D))
                                        (SETQ TWICEBORDER (fetch (REGION BOTTOM) of CLIPREG)))
                                (DSPYPOSITION TWICEBORDER D))
                               ((IGREATERP R (SETQ TWICEBORDER (IPLUS (fetch (REGION HEIGHT) of CLIPREG)
                                                                      TWICEBORDER)))
                                (DSPYPOSITION (IDIFFERENCE TWICEBORDER (FONTPROP D 'ASCENT))
                                              D]
        (UPDATE/SCROLL/REG WINDOW))
    WINDOW])
```

(**SHOWWFRAME**

```
[LAMBDA (WIN)                                                   ; Edited 24-Sep-92 12:31 by jds
```

;; Displays the border and title in the save image of a window

```
(PROG ((TITLE (fetch (WINDOW WTITLE) of WIN))
       (BORDER (fetch (WINDOW WBORDER) of WIN))
       (DSP (fetch (WINDOW DSP) of WIN))
       (SAVEIMAGE (fetch (WINDOW SAVE) of WIN))
       WINWDTH WINHGHT BLACKPART WHITEPART)
      [SETQ WINHGHT (fetch (REGION HEIGHT) of (SETQ WINWDTH (fetch (WINDOW REG) of WIN]
      (SETQ WINWDTH (fetch (REGION WIDTH) of WINWDTH))          ; make most of the border black
      (SETQ BLACKPART (IMAX (FOLDHI BORDER 2)
                            (IDIFFERENCE BORDER 2)))
      (SETQ WHITEPART (IDIFFERENCE BORDER BLACKPART))           ; Fill in frame in save image
      (BITBLT NIL NIL NIL SAVEIMAGE 0 0 WINWDTH WINHGHT 'TEXTURE 'REPLACE BLACKSHADE)
                                                                ; White out the frame in the saved image
      (BITBLT NIL NIL NIL SAVEIMAGE BLACKPART BLACKPART (IDIFFERENCE WINWDTH (ITIMES 2 BLACKPART))
             (IPLUS (fetch (REGION HEIGHT) of (DSPCLIPPINGREGION NIL DSP))
                    (ITIMES 2 WHITEPART))
             'TEXTURE
             'REPLACE WHITESHADE)
      (AND TITLE (SHOWWTITLE TITLE SAVEIMAGE BORDER NIL WIN)))
WIN])
```

(**SHOWWTITLE**

```
[LAMBDA (TITLE BM BORDER CENTERFLG WINDOW)                      (* kbr%: "25-Jan-86 15:21")
```

;; prints a title in a window.

```
(PROG (TITLEDS FONT BLACKPART TITLESHADE BMWIDTH HEIGHT BOTTOM X LEFTMARGIN)
      (SETQ TITLEDS (fetch (SCREEN SCTITLEDS) of (fetch (WINDOW SCREEN) of WINDOW)))
```

```
          (SETQ FONT (DSPFONT NIL TITLEDS))
          (SETQ BLACKPART (SELECTQ BORDER
                                (0 0)
                                ((1 2)
                                    1)
                                (3 2)
                                (IDIFFERENCE BORDER 2)))
          (SETQ TITLESHADE (OR (TEXTUREP (OR (WINDOWPROP WINDOW 'WINDOWTITLESHADE)
                                             WINDOWTITLESHADE))
                               BLACKSHADE))
          (DSPDESTINATION BM TITLEDS)
          (DSPCLIPPINGREGION (create REGION
                                     LEFT _ 0
                                     BOTTOM _ [SETQ BOTTOM (IDIFFERENCE (IPLUS (BITMAPHEIGHT BM)
                                                                               (COND
                                                                                  ((ZEROP BORDER)
                                                                                   0)
                                                                                  (T
                                                                     ; if room, leave a line of the border at the top of the title.
                                                                                     -1)))
                                                                      (SETQ HEIGHT (FONTPROP FONT 'HEIGHT)]
                                     WIDTH _ (SETQ BMWIDTH (BITMAPWIDTH BM))
                                     HEIGHT _ HEIGHT)
                TITLEDS)
          (MOVETO (COND
                     [CENTERFLG                                    ; save left margin for later shading.
                            (SETQ LEFTMARGIN (IMAX BORDER (IQUOTIENT (IDIFFERENCE BMWIDTH (\STRINGWIDTHGUESS
                                                                                            TITLE FONT))
                                                                      2]
                     (T BORDER))
                  (IPLUS BOTTOM (FONTPROP FONT 'DESCENT))
                  TITLEDS)
          (RESETFORM (PRINTLEVEL WINDOWTITLEPRINTLEVEL)
                  (PROG ((PLVLFILEFLG T))
                        (PRIN3 TITLE TITLEDS)))
          (BITBLT NIL NIL NIL TITLEDS (SETQ X (IPLUS (IMAX 2 BLACKPART)
                                                     (DSPXPOSITION NIL TITLEDS)))
                  (COND
                     ((EQ BLACKPART 1)
                      (ADD1 BOTTOM))
                     (T BOTTOM))
                  (IDIFFERENCE BMWIDTH (IPLUS X BLACKPART))
                  NIL
                  'TEXTURE
                  'REPLACE TITLESHADE)                           ; shade stuff before title if centered.
          (AND CENTERFLG (BITBLT NIL NIL NIL TITLEDS BORDER (COND
                                                              ((EQ BLACKPART 1)
                                                               (ADD1 BOTTOM))
                                                              (T BOTTOM))
                                 (IDIFFERENCE LEFTMARGIN (IPLUS (IMAX 2 BLACKPART)
                                                                BORDER))
                                 NIL
                                 'TEXTURE
                                 'REPLACE TITLESHADE])
```

(\STRINGWIDTHGUESS

```
[LAMBDA (X FONT)                                                 ; Edited 3-Apr-87 13:44 by jop
```

;; returns a guess as to the string width of X. It goes one level so works on circular structures. It is used as a heuristic by functions who are going ;; to print something with printlevel.

```
(STRINGWIDTH X FONT T])
```

(RESHOWTITLE

```
[LAMBDA (TITLE WINDOW JUSTDISPLAYFLG)                            (* kbr%: "25-Jan-86 15:26")
```

;; updates a windows display with a new title

```
(PROG* ((WREG (fetch (WINDOW REG) of WINDOW))
        (TITLEDS (fetch (SCREEN SCTITLEDS) of (fetch (WINDOW SCREEN) of WINDOW)))
        (TITLEHEIGHT (IMINUS (DSPLINEFEED NIL TITLEDS)))
        (OLDTITLE (fetch (WINDOW WTITLE) of WINDOW))
        (BORDER (fetch (WINDOW WBORDER) of WINDOW))
        BM BMBTM HGHT)
       [COND
          (JUSTDISPLAYFLG)
          ((EQ TITLE (fetch (WINDOW WTITLE) of WINDOW))
           (RETURN))
          (T (replace (WINDOW WTITLE) of WINDOW with TITLE)
             (COND
                ([OR (NULL OLDTITLE)
                     (NULL TITLE)
                     (NEQ TITLEHEIGHT (IDIFFERENCE (fetch (REGION HEIGHT) of WREG)
                                                   (IPLUS (fetch (REGION HEIGHT) of (DSPCLIPPINGREGION
                                                                                      NIL
                                                                                      (fetch (WINDOW DSP)
                                                                                         of WINDOW)))
```

```
                                                    (ITIMES 2 BORDER]
                                                           ; Previously no title, so make space for one
                                                           ; Have to remove title
                                                           ; or title height changed.
                                                           ; so windows region on the screen has to be made larger.
                    (\RESHOWBORDER1 (fetch (WINDOW WBORDER) of WINDOW)
                           (fetch (WINDOW WBORDER) of WINDOW)
                           WINDOW)
                    (RETURN]                                ; code from here is to reprint the title in place to avoid creating
                                                           ; any large bitmaps.
             [SETQ BM (BITMAPCREATE (fetch (REGION WIDTH) of WREG)
                           (SETQ TITLEHEIGHT (ADD1 TITLEHEIGHT))
                           (BITSPERPIXEL (fetch (SCREEN SCDESTINATION) of (fetch (WINDOW SCREEN) of WINDOW]
             (BITBLT NIL NIL NIL BM 0 0 NIL NIL 'TEXTURE 'REPLACE BLACKSHADE)
                                                           ; use SHOWWTITLE to put the image of the title into the
                                                           ; auxilliary bitmap.
             (SHOWWTITLE TITLE BM BORDER NIL WINDOW)
             [COND
                ((IGREATERP TITLEHEIGHT (SETQ HGHT (fetch (REGION HEIGHT) of WREG)))
                 (SETQ BMBTM (IDIFFERENCE (SUB1 TITLEHEIGHT)
                                   HGHT]
             (UNINTERRUPTABLY
                 (TOTOPW WINDOW)
                 (BITBLT BM 0 (COND
                                 (BMBTM)
                                 ((IGREATERP BORDER 0)
                                  ;; if there is a border, the title was printed in the scratch bitmap so to leave one point of the border on top
                                  0)
                                 (T 1))
                          (fetch (SCREEN SCDESTINATION) of (fetch (WINDOW SCREEN) of WINDOW))
                          (fetch (REGION LEFT) of WREG)
                          [IDIFFERENCE (fetch (REGION PTOP) of WREG)
                                 (COND
                                    (BMBTM HGHT)
                                    (T (IPLUS TITLEHEIGHT (COND
                                                             ((IGREATERP BORDER 0)
                                                              ;; if there is a border, the title was printed in the scratch bitmap so to leave
                                                              ;; one point of the border on top
                                                              0)
                                                             (T -1]
                          NIL
                          (COND
                             (BMBTM HGHT)))))])
```

(TOTOPW

```
  [LAMBDA (WINDOW NOCALLTOTOPFNFLG)                        ; Edited 21-Feb-94 12:57 by sybalsky
    (WINDOWOP 'TOTOPWFN (fetch (WINDOW SCREEN) of (\INSUREWINDOW WINDOW))
           WINDOW NOCALLTOTOPFNFLG])
```

(\INTERNALTOTOPW

```
  [LAMBDA (W1 RPT)                                         (* gbn%: "25-Jan-86 15:36")
    (PROG (SCREEN SCREENTOPW)
          (SETQ W1 (\INSUREWINDOW W1))
          (SETQ SCREEN (fetch (WINDOW SCREEN) of W1))
          (SETQ SCREENTOPW (fetch (SCREEN SCTOPW) of SCREEN))
          (OR (EQ W1 SCREENTOPW)
              (COND
                 ((NULL SCREENTOPW)                        ; all windows are closed open this one.
                  (OPENW W1))
                 (T (UNINTERRUPTABLY
                        (\TTW1 W1 SCREENTOPW)
                        ;; N.B. \TTW1 can side effect the screen
                        (COND
                           ((EQ W1 (fetch (SCREEN SCTOPW) of SCREEN)))
                           ((NOT RPT)                      ; GC msgs or other glitches can cause W1 not to make it.  Check
                                                           ; and try ONCE more
                            (\INTERNALTOTOPW W1 T))))])
```

(\TTW1

```
  [LAMBDA (WINDOW WS)                                      ; Edited 31-Jul-92 10:06 by jds
```

;;; This seems to swap the intersection of bitmaps.

```
    (COND
       [(fetch (WINDOW NEXTW) of WS)
        (PROG (ISECT SCREEN)
              (SETQ SCREEN (fetch (WINDOW SCREEN) of WINDOW))
              (.WHILE.TOP.DS. \TOPWDS (SETQ ISECT (INTERSECTREGIONS (fetch (WINDOW REG) of WINDOW)
                                                                    (fetch (WINDOW REG) of WS)
                                                                    (fetch (SCREEN SCREGION) of SCREEN)))
                     [AND ISECT (\SW2BM (fetch (SCREEN SCDESTINATION) of SCREEN)
```

```
                                    ISECT
                                    (fetch (WINDOW SAVE) of WS)
                                    (TRANSLATEREG ISECT (fetch (WINDOW REG) of WS]
                    [COND
                       ((EQ WINDOW (fetch (WINDOW NEXTW) of WS)) ; doesn't have to be uninterruptable here because TOTOPW is.
                        (replace (WINDOW NEXTW) of WS with (fetch (WINDOW NEXTW) of WINDOW))
                        (replace (WINDOW NEXTW) of WINDOW with (fetch (SCREEN SCTOPW) of SCREEN))
                        (replace (SCREEN SCTOPW) of SCREEN with WINDOW)
                        (SETQ \TOPWDS (fetch (WINDOW DSP) of WINDOW)))
                       (T (\TTW1 WINDOW (fetch (WINDOW NEXTW) of WS]
                    (AND ISECT (\SW2BM (fetch (WINDOW SAVE) of WINDOW)
                                    (TRANSLATEREG ISECT (fetch (WINDOW REG) of WINDOW))
                                    (fetch (WINDOW SAVE) of WS)
                                    (TRANSLATEREG ISECT (fetch (WINDOW REG) of WS]
       ((type? WINDOW WINDOW)                                   ; must be closed window; reopen it
        (OPENW WINDOW])
```

(WHICHW

```
  [LAMBDA (X Y SCREEN)                                          (* gbn%: "25-Jan-86 15:47")
    (SETQ SCREEN (\INSURESCREEN SCREEN))
    (COND
       ((POSITIONP X)
        (WHICHW (fetch (POSITION XCOORD) of X)
                (fetch (POSITION YCOORD) of X)
                SCREEN))
       (T (for (WINDOW _ (fetch (SCREEN SCTOPW) of SCREEN)) by (fetch (WINDOW NEXTW) of WINDOW) while WINDOW
             thereis (INSIDE? (fetch (WINDOW REG) of WINDOW)
                              X Y])

)

(RPAQ? WINDOWTITLEPRINTLEVEL '(2 . 5))

(RPAQ? WINDOWTITLESHADE BLACKSHADE)
```

;; Window vs non-window world

```
(DEFINEQ
```

(WFROMDS

```
  [LAMBDA (DS DONTCREATE)                                       ; Edited  7-Jan-94 12:12 by nilsson
```

;; Finds or creates a window for a display stream

;; uses an XPointer from the displaystream as a hint. This means that the window might have been garbage collected, hence all the confirmation.

```
    (DECLARE (GLOBALVARS \DEFAULTTTYDISPLAYSTREAM))
    (COND
       ((WINDOWP DS)
        DS)
       ((IMAGESTREAMP DS)
        (PROG (DD HINTW)
              [COND
                 ((IMAGESTREAMTYPEP DS 'TEXT)
```

;; generalize this mess!!!

```
                  (RETURN (CAR (fetch (TEXTOBJ \WINDOW) of (TEXTOBJ DS]
              (SETQ DD (\GETDISPLAYDATA DS DS))
              (RETURN (COND
                         ((AND (SETQ HINTW (fetch (\DISPLAYDATA XWINDOWHINT) of DD))
                               (EQ (fetch (WINDOW DSP) of HINTW)
                                   DS))
                          HINTW)
                         [(AND (EQ DS \DEFAULTTTYDISPLAYSTREAM)
                               (EQ (TTYDISPLAYSTREAM)
                                   \DEFAULTTTYDISPLAYSTREAM))     ; assume this process is doing something with T.
                          (COND
                             ((NOT DONTCREATE)
                              (\CREATE.TTYDISPLAYSTREAM)
                              (WFROMDS (TTYDISPLAYSTREAM]
                         ([SETQ HINTW (for WINDOW in (OPENWINDOWS T) thereis (EQ DS (fetch (WINDOW DSP)
                                                                                           of WINDOW]
                                                                    ; (OPENWINDOWS T) returns all windows on all screens
                          HINTW)
                         ((NOT DONTCREATE)
                          (CREATEW NIL NIL NIL T])
```

(NU\TOTOPWDS

```
  [LAMBDA (DS NOTOTOPFNFLG)                                     ; Edited 17-Aug-88 19:37 by jds
                                                                ; Moves the window of displaystream DS to the top
    (AND (FMEMB (DSPDESTINATION NIL DS)
                \SCREENBITMAPS)
         (TOTOPW (WFROMDS DS)
                 NOTOTOPFNFLG])
```

(\COERCETODS

```
  [LAMBDA (X)                                                    (* rrb "23-OCT-81 13:29")
    ;; Called from \SFInsureDisplayStream macro.  Compiles open in system code, closed call in user code, and equivalent to \ILLEGAL.ARG if no
    ;; window package.
    (COND
       ((type? WINDOW X)
        (fetch (WINDOW DSP) of X))
       (T (\ILLEGAL.ARG X])

)

(DECLARE%: DONTCOPY

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: EVAL@COMPILE

(PUTPROPS \COERCETODS MACRO [OPENLAMBDA (X)
                                (COND
                                   ((type? WINDOW X)
                                    (fetch (WINDOW DSP) of X))
                                   (T (\ILLEGAL.ARG X])

(PUTPROPS .WHILE.ON.TOP. MACRO ((FIRST . REST)
                                (UNINTERRUPTABLY
                                    (\INTERNALTOTOPW FIRST)
                                     . REST)))
)
)

;; END EXPORTED DEFINITIONS

(MOVD 'NU\TOTOPWDS '\TOTOPWDS)

;; User interface functions

(DEFINEQ

(WINDOWP
  [LAMBDA (X)                                                    (* rrb "20-NOV-81 07:30")
    (AND (type? WINDOW X)
         X])

(INSURE.WINDOW
  [LAMBDA (WIN? NOERRORFLG)                                      (* rrb "17-Mar-86 15:39")

;;; coerces WIN?  to a window.

    (COND
       ((type? WINDOW WIN?)
        WIN?)
       ((DISPLAYSTREAMP (\OUTSTREAMARG WIN? T))
        (WFROMDS WIN?))
       ((NULL NOERRORFLG)
        (\ILLEGAL.ARG WIN?])

(WINDOWPROP
  [LAMBDA X                                                      (* rrb "26-AUG-82 17:36")
    ;; general top level entry for both fetching and setting window properties.
    (COND
       ((IGREATERP X 2)
        (PUTWINDOWPROP (ARG X 1)
               (ARG X 2)
               (ARG X 3)))
       ((EQ X 2)
        (GETWINDOWPROP (ARG X 1)
               (ARG X 2)))
       (T (\ILLEGAL.ARG NIL])

(WINDOWADDPROP
  [LAMBDA (WINDOW PROP ITEMTOADD FIRSTFLG)                       (* rrb "20-Mar-84 16:07")
    ;; adds an element to a window property.
    (PROG ((CURRENT (WINDOWPROP WINDOW PROP)))
          (RETURN (WINDOWPROP WINDOW PROP (COND
                                             ((NULL CURRENT)
                                              (LIST ITEMTOADD))
                                             [(NLISTP CURRENT)
                                              (COND
                                                 ((EQ CURRENT ITEMTOADD)
                                                  (LIST ITEMTOADD))
```

```
                                        (FIRSTFLG (LIST ITEMTOADD CURRENT))
                                        (T (LIST CURRENT ITEMTOADD]
                                    ((FMEMB ITEMTOADD CURRENT)
                                                    ; don't put things on twice.
                                     (COND
                                        ((AND FIRSTFLG (NEQ (CAR CURRENT)
                                                            ITEMTOADD))
                                                    ; make it first
                                         (CONS ITEMTOADD (REMOVE ITEMTOADD CURRENT)))
                                        (T CURRENT)))
                                    (FIRSTFLG (CONS ITEMTOADD CURRENT))
                                    (T (NCONC1 (APPEND CURRENT)
                                               ITEMTOADD])
```

(**WINDOWDELPROP**

```
[LAMBDA (WINDOW PROP ITEMTODELETE)                          (* rrb "13-JUN-82 17:58")
```

;; deletes a property from a window property.

```
(PROG ((CURRENT (WINDOWPROP WINDOW PROP)))
      (RETURN (COND
                 ((LISTP CURRENT)
                  (AND (FMEMB ITEMTODELETE CURRENT)
                       (WINDOWPROP WINDOW PROP (REMOVE ITEMTODELETE CURRENT])
```

(**GETWINDOWPROP**

```
[LAMBDA (WINDOW PROP)                                       ; Edited 27-Dec-93 11:46 by sybalsky:mv:envos
```

;; gets values from a window. Called by the macro for WINDOWPROP.

```
[OR (type? WINDOW WINDOW)
    (COND
       ((DISPLAYSTREAMP (\OUTSTREAMARG WINDOW T))
        (SETQ WINDOW (WFROMDS WINDOW)))
       (T (\ILLEGAL.ARG WINDOW]
(WINDOWOP 'GETWINDOWPROPFN (fetch (WINDOW SCREEN) of WINDOW)
       WINDOW PROP VALUE])
```

(**GETWINDOWUSERPROP**

```
[LAMBDA (WINDOW USERPROP)                                   (* rrb "28-OCT-83 11:00")
```

;; gets a property from the USERDATA property list of a window. This is the function called by the macro for GETWINDOWPROP which result
;; from a call to WINDOWPROP that doesn't have a third argument.

```
(LISTGET (fetch (WINDOW USERDATA) of (\INSUREWINDOW WINDOW))
       USERPROP])
```

(**PUTWINDOWPROP**

```
[LAMBDA (WINDOW PROP VALUE)                                 ; Edited 27-Dec-93 11:46 by sybalsky:mv:envos
  [OR (type? WINDOW WINDOW)
      (COND
         ((DISPLAYSTREAMP (\OUTSTREAMARG WINDOW))
          (SETQ WINDOW (WFROMDS WINDOW)))
         (T (\ILLEGAL.ARG WINDOW]
  (WINDOWOP 'PUTWINDOWPROPFN (fetch (WINDOW SCREEN) of WINDOW)
       WINDOW PROP VALUE])
```

(**REMWINDOWPROP**

```
[LAMBDA (WINDOW PROP)                                       (* rmk%: "31-AUG-83 16:42")
  [OR (type? WINDOW WINDOW)
      (COND
         ((DISPLAYSTREAMP (\OUTSTREAMARG WINDOW))
          (SETQ WINDOW (WFROMDS WINDOW)))
         (T (LISPERROR "ILLEGAL ARG" WINDOW]
  (PROG (DATA)
        (SETQ DATA (fetch (WINDOW USERDATA) of WINDOW))
        (RETURN (for TAIL on DATA by (CDDR TAIL) bind PREV do (COND
                                                                 ((EQ (CAR TAIL)
                                                                      PROP)
                                                                  (COND
                                                                     (PREV (RPLACD (CDR PREV)
                                                                                   (CDDR TAIL)))
                                                                     ((CDDR TAIL)
                                                                      (FRPLNODE2 TAIL (CDDR TAIL)))
                                                                     (T (replace (WINDOW USERDATA) of WINDOW
                                                                           with NIL)))
                                                                  (RETURN PROP)))
                                                              (SETQ PREV TAIL]))
```

(**WINDOWADDFNPROP**

```
[LAMBDA (WINDOW PROP ITEMTOADD)                             (* rrb "18-JUN-82 16:30")
```

;; adds A functional element to a window property. This is different from WINDOWADDTOPROP because is checks for LAMBDA expressions as a
;; single element.

```
    (PROG ((CURRENT (WINDOWPROP WINDOW PROP)))
          (RETURN (WINDOWPROP WINDOW PROP (COND
                                             ((NULL CURRENT)
                                              (LIST ITEMTOADD))
                                             ((OR (NLISTP CURRENT)
                                                  (FMEMB (CAR CURRENT)
                                                         LAMBDASPLST))
                                              (LIST CURRENT ITEMTOADD))
                                             ((FMEMB ITEMTOADD CURRENT)
                                                                  ; don't put things on twice.
                                              CURRENT)
                                             (T (NCONC1 (APPEND CURRENT)
                                                        ITEMTOADD])

)

;; Compiled WINDOWPROP

(PUTPROPS WINDOWPROP ARGNAMES (NIL (WINDOW PROP {NEWVALUE}) . U))


(DEFOPTIMIZER WINDOWPROP (&REST ARGS)
                         (CWINDOWPROP ARGS))

(DEFINEQ

(CWINDOWPROP
  [LAMBDA (FORMTAIL)                                          (* rrb "28-OCT-83 10:51")
    ;; compiles calls to WINDOWPROP
    (COND
       ((NULL (CDR FORMTAIL))                                 ; less than 2 args
        (printout T "Possible error in call to WINDOWPROP: less than 2 args" T (LIST 'WINDOWPROP FORMTAIL)
               T)
        (CGETWINDOWPROP (CAR FORMTAIL)
               NIL))
       ((NOT (EQ (CAADR FORMTAIL)
                 'QUOTE))                                     ; property is not quoted.
        'IGNOREMACRO)
       [(NULL (CDDR FORMTAIL))                                ; fetching a window property.
        (CGETWINDOWPROP (CAR FORMTAIL)
               (CADR (CADR FORMTAIL]
       (T                                                     ; storing a window property
          (CONS 'PUTWINDOWPROP FORMTAIL])


(CGETWINDOWPROP
  [LAMBDA (WINFORM PROP)                                      (* kbr%: "17-Feb-86 10:43")
    ;; compiles calls on WINDOWPROP that are fetching values.  This needs to be changed whenever GETWINDOWPROP is changed.
    (PROG NIL
          (RETURN (SUBST (LIST '\INSUREWINDOW WINFORM)
                         'DATUM
                         (SELECTQ PROP
                              (RIGHTBUTTONFN
                                   (CONSTANT (RECORDACCESSFORM '(WINDOW RIGHTBUTTONFN)
                                                  'DATUM
                                                  'ffetch)))
                              (BUTTONEVENTFN
                                   (CONSTANT (RECORDACCESSFORM '(WINDOW BUTTONEVENTFN)
                                                  'DATUM
                                                  'ffetch)))
                              (CURSORINFN (CONSTANT (RECORDACCESSFORM '(WINDOW CURSORINFN)
                                                         'DATUM
                                                         'ffetch)))
                              (CURSOROUTFN (CONSTANT (RECORDACCESSFORM '(WINDOW CURSOROUTFN)
                                                          'DATUM
                                                          'ffetch)))
                              (CURSORMOVEDFN
                                   (CONSTANT (RECORDACCESSFORM '(WINDOW CURSORMOVEDFN)
                                                  'DATUM
                                                  'ffetch)))
                              (DSP (CONSTANT (RECORDACCESSFORM '(WINDOW DSP)
                                                 'DATUM
                                                 'ffetch)))
                              (SCREEN (CONSTANT (RECORDACCESSFORM '(WINDOW SCREEN)
                                                     'DATUM
                                                     'ffetch)))
                              (SCROLLFN (CONSTANT (RECORDACCESSFORM '(WINDOW SCROLLFN)
                                                       'DATUM
                                                       'ffetch)))
                              (MOVEFN (CONSTANT (RECORDACCESSFORM '(WINDOW MOVEFN)
                                                     'DATUM
                                                     'ffetch)))
                              (RESHAPEFN (CONSTANT (RECORDACCESSFORM '(WINDOW RESHAPEFN)
                                                        'DATUM
                                                        'ffetch)))
```

```
                                (EXTENT (CONSTANT (RECORDACCESSFORM '(WINDOW EXTENT)
                                                            'DATUM
                                                            'ffetch)))
                                (REPAINTFN (CONSTANT (RECORDACCESSFORM '(WINDOW REPAINTFN)
                                                              'DATUM
                                                              'ffetch)))
                                (CLOSEFN (CONSTANT (RECORDACCESSFORM '(WINDOW CLOSEFN)
                                                             'DATUM
                                                             'ffetch)))
                                (WINDOWENTRYFN
                                    (CONSTANT (RECORDACCESSFORM '(WINDOW WINDOWENTRYFN)
                                                        'DATUM
                                                        'ffetch)))
                                (PROCESS (CONSTANT (RECORDACCESSFORM '(WINDOW PROCESS)
                                                             'DATUM
                                                             'ffetch)))
                                (REGION (CONSTANT (RECORDACCESSFORM '(WINDOW REG)
                                                            'DATUM
                                                            'ffetch)))
                                (NEWREGIONFN (CONSTANT (RECORDACCESSFORM '(WINDOW NEWREGIONFN)
                                                                 'DATUM
                                                                 'ffetch)))
                                (TITLE (CONSTANT (RECORDACCESSFORM '(WINDOW WTITLE)
                                                           'DATUM
                                                           'ffetch)))
                                (BORDER (CONSTANT (RECORDACCESSFORM '(WINDOW WBORDER)
                                                            'DATUM
                                                            'ffetch)))
                                (IMAGECOVERED (CONSTANT (RECORDACCESSFORM '(WINDOW SAVE)
                                                                 'DATUM
                                                                 'ffetch)))
                                (HEIGHT (LIST 'GETWINDOWPROP WINFORM ''HEIGHT))
                                (WIDTH (LIST 'GETWINDOWPROP WINFORM ''WIDTH))
                                (RETURN (PROGN ;; return around SUBST.  GETWINDOWUSERPROP will perform the window check and this
                                               ;; avoids compiling code for it at every call.
                                               (LIST 'GETWINDOWUSERPROP WINFORM (KWOTE PROP])
```

(\GETWINDOWHEIGHT

```
  [LAMBDA (WINDOW)                                                    (* gbn%: "25-Jan-86 15:45")
    ;; calculate the height from the REGION in case user has changed the clipping region.  This won't work if the height of the title display stream has
    ;; changed.
    (SETQ WINDOW (\INSUREWINDOW WINDOW))
    (DIFFERENCE (fetch (REGION HEIGHT) of (fetch (WINDOW REG) of WINDOW))
           (DIFFERENCE (ITIMES 2 (fetch (WINDOW WBORDER) of WINDOW))
                  (COND
                     [(fetch (WINDOW WTITLE) of WINDOW)
                      (DSPLINEFEED NIL (fetch (SCREEN SCTITLEDS) of (fetch (WINDOW SCREEN) of WINDOW]
                     (T 0])
```

(\GETWINDOWWIDTH

```
  [LAMBDA (WINDOW)                                                    (* rrb " 4-Jun-84 18:03")
    ;; calculate the width from the REGION in case the user has changed the clipping region.
    (SETQ WINDOW (\INSUREWINDOW WINDOW))
    (DIFFERENCE (fetch (REGION WIDTH) of (fetch (WINDOW REG) of WINDOW))
           (ITIMES 2 (fetch (WINDOW WBORDER) of WINDOW])

)

(DEFINEQ
```

(WINDOW.BITMAP

```
  [LAMBDA (W)                                                         ; Edited 12-Jun-90 10:38 by mitani
                                                                      (* Returns all of the bitmap of the window)
    (PROG [BM (REGION (WINDOWPROP W 'REGION]
          (CLOSEW W)
          (SETQ BM (BITMAPCREATE (fetch (REGION WIDTH) of REGION)
                                 (fetch (REGION HEIGHT) of REGION)))
          (BITBLT (WINDOWPROP W 'IMAGECOVERED)
                 NIL NIL BM)
          (OPENW W)
          (RETURN BM])

)

;; lmm 4/23

(DEFINEQ
```

(OPENWP

```
  [LAMBDA (WINDOW)                                                    (* rrb "26-OCT-83 15:01")
    ;; is WINDOW an open window?
```

```
    (AND (type? WINDOW WINDOW)
         (NEQ (fetch (WINDOW NEXTW) of WINDOW)
              'CLOSED)
         WINDOW])
```

(TOPWP

```
  [LAMBDA (WINDOW)                                                    (* kbr%: "17-Feb-86 10:37")
```

;;; A function user's can use to test if WINDOW is the TOPW of it's screen.

```
    (EQ WINDOW (fetch (SCREEN SCTOPW) of (fetch (WINDOW SCREEN) of WINDOW])
```

(RESHAPEBYREPAINTFN

```
  [LAMBDA (WINDOW OLDIMAGE IMAGEREGION OLDSCREENREGION)               (* rrb "11-Oct-84 17:22")
```

;; default reshaping function that copies the lower left portion of the old image into the new image and calls the repaint function on the newly ;; exposed portions.

;; if IMAGEREGION shares a corner with the current region, the excess is added in the opposite directions. Also the newly exposed region will be ;; a subset of the EXTENT property if the window has one.

```
    (PROG ((NEWSCREENREGION (WINDOWPROP WINDOW 'REGION))
           (EXTENT (WINDOWPROP WINDOW 'EXTENT))
           (DSP (WINDOWPROP WINDOW 'DSP))
           (OLDWIDTH (fetch (REGION WIDTH) of IMAGEREGION))
           (OLDHEIGHT (fetch (REGION HEIGHT) of IMAGEREGION))
           NEWWID NEWHGHT WREGION OLDCRLFT OLDCRBTM NEWCRLFT NEWCRBTM DELTAWID DELTAHGHT NEWPTOP OLDPTOP
           NEWPRIGHT OLDPRIGHT YPOS)
          (SETQ WREGION (DSPCLIPPINGREGION NIL DSP))
          (SETQ OLDCRLFT (fetch (REGION LEFT) of WREGION))
          (SETQ OLDCRBTM (fetch (REGION BOTTOM) of WREGION))          ; calculate the position of the new clipping region.
          (SETQ NEWWID (fetch (REGION WIDTH) of WREGION))
          (SETQ DELTAWID (IDIFFERENCE NEWWID OLDWIDTH))
          (SETQ NEWHGHT (fetch (REGION HEIGHT) of WREGION))
          (SETQ DELTAHGHT (IDIFFERENCE NEWHGHT OLDHEIGHT))
          [COND
             [(AND OLDSCREENREGION EXTENT (EQ (fetch (REGION PRIGHT) of NEWSCREENREGION)
                                              (fetch (REGION PRIGHT) of OLDSCREENREGION)))
                                                                      ; right edges match, move the left one
              (SETQ NEWCRLFT (IDIFFERENCE OLDCRLFT DELTAWID))
              (COND
                 ((AND (IGREATERP DELTAWID 0)
                       (IGREATERP (fetch (REGION LEFT) of EXTENT)
                              NEWCRLFT))
                  ;; this would be extending the window onto parts of the extent that don't have anything in them, reset the left so that it gets the
                  ;; entire extent
                  (SETQ NEWCRLFT (IMIN (fetch (REGION LEFT) of EXTENT)
                                       (IDIFFERENCE (fetch (REGION RIGHT) of EXTENT)
                                              NEWWID]
             (T                                                       ; otherwise move the right edge.
                (COND
                   [(AND (IGREATERP DELTAWID 0)
                         EXTENT
                         (IGREATERP (IPLUS OLDCRLFT NEWWID)
                                (fetch (REGION RIGHT) of EXTENT)))
                    ;; this would be extending the window onto parts of the extent that don't have anything in them, reset the left so that it gets the
                    ;; entire extent
                    (SETQ NEWCRLFT (IMAX (IMIN (fetch (REGION LEFT) of EXTENT)
                                               OLDCRLFT)
                                         (IDIFFERENCE OLDCRLFT DELTAWID]
                    (T (SETQ NEWCRLFT OLDCRLFT]
          [COND
             [(AND OLDSCREENREGION (EQ (fetch (REGION PTOP) of NEWSCREENREGION)
                                       (fetch (REGION PTOP) of OLDSCREENREGION)))
                                                                      ; top edges match, move the bottom one
              (SETQ NEWCRBTM (IDIFFERENCE OLDCRBTM DELTAHGHT))
              (COND
                 ((AND (IGREATERP DELTAHGHT 0)
                       EXTENT
                       (IGREATERP (fetch (REGION BOTTOM) of EXTENT)
                              NEWCRBTM))
                  ;; this would be extending the window onto parts of the extent that don't have anything in them, reset the bottom so that it gets
                  ;; the entire extent
                  (SETQ NEWCRBTM (IMIN (fetch (REGION BOTTOM) of EXTENT)
                                       (IDIFFERENCE (fetch (REGION TOP) of EXTENT)
                                              NEWHGHT]
             (T                                                       ; otherwise move the top edge.
                (COND
                   [(AND (IGREATERP DELTAHGHT 0)
                         EXTENT
                         (IGREATERP (IPLUS OLDCRBTM OLDHEIGHT DELTAHGHT)
                                (fetch (REGION PTOP) of EXTENT)))
                    ;; this would be extending the window onto parts of the extent that don't have anything in them, reset the bottom so that it
                    ;; gets the entire extent
```

```
                (SETQ NEWCRBTM (IMAX (IDIFFERENCE OLDCRBTM DELTAHGHT)
                                     (fetch (REGION BOTTOM) of EXTENT)
                                     (IDIFFERENCE (fetch (REGION PTOP) of EXTENT)
                                            NEWHGHT]
               (T (SETQ NEWCRBTM OLDCRBTM]                    ; scroll the window so that the new left bottom is the left bottom
                                                              ; of the clipping region.
      [COND
         ((AND (NULL EXTENT)
               (\INBETWEENP (DSPXPOSITION NIL WINDOW)
                      OLDCRLFT
                      (IPLUS OLDCRLFT OLDWIDTH))
               (\INBETWEENP (SETQ YPOS (DSPYPOSITION NIL WINDOW))
                      OLDCRBTM
                      (IPLUS OLDCRBTM OLDHEIGHT)))
          ;; if the window doesn't have any EXTENT and its position is visible, make sure its Y position is visible at the end of the scroll.
          (COND
             [(ILESSP YPOS NEWCRBTM)                          ; make sure the entire line of text being printed is visible.
              (SETQ NEWCRBTM (DIFFERENCE YPOS (FONTPROP WINDOW 'DESCENT]
             ([IGREATERP YPOS (DIFFERENCE (IPLUS NEWCRBTM NEWHGHT)
                                          (FONTPROP WINDOW 'ASCENT]
              (SETQ NEWCRBTM (IPLUS (IDIFFERENCE YPOS NEWHGHT)
                                    (FONTPROP WINDOW 'ASCENT]
      [COND
         ((NEQ OLDCRLFT NEWCRLFT)
          (COND
             ((EQ (DSPSCROLL NIL WINDOW)
                  'ON)                                        ; if scrolling is turned on, don't change the coordinates.
              NIL)
             (T (WXOFFSET (DIFFERENCE OLDCRLFT NEWCRLFT)
                       WINDOW]
      [COND
         ((NEQ OLDCRBTM NEWCRBTM)
          (COND
             ((EQ (DSPSCROLL NIL WINDOW)
                  'ON)                                        ; if scrolling is turned on, change the Y rather than the
                                                              ; coordinates.
              (DSPYPOSITION (PLUS (DIFFERENCE OLDCRBTM NEWCRBTM)
                                  YPOS)
                     WINDOW))
             (T (WYOFFSET (DIFFERENCE OLDCRBTM NEWCRBTM)
                       WINDOW]                                ; call the redisplay function on the four possible areas and blt the
                                                              ; middle one.
      (COND
         ((IGREATERP (SETQ NEWPTOP (IPLUS NEWCRBTM NEWHGHT))
                  (SETQ OLDPTOP (IPLUS OLDCRBTM OLDHEIGHT)))  ; call the display function on the newly exposed top area.
          (REDISPLAYW WINDOW (create REGION
                                     LEFT _ NEWCRLFT
                                     BOTTOM _ OLDPTOP
                                     WIDTH _ NEWWID
                                     HEIGHT _ (IDIFFERENCE NEWPTOP OLDPTOP))
                      T)))
      (COND
         ((IGREATERP OLDCRLFT NEWCRLFT)                       ; call the display function on the newly exposed LEFT area.
          (REDISPLAYW WINDOW (create REGION
                                     LEFT _ NEWCRLFT
                                     BOTTOM _ OLDCRBTM
                                     WIDTH _ (IDIFFERENCE OLDCRLFT NEWCRLFT)
                                     HEIGHT _ OLDHEIGHT)
                      T)))                                    ; blt center region.
      (BITBLT OLDIMAGE (fetch (REGION LEFT) of IMAGEREGION)
             (fetch (REGION BOTTOM) of IMAGEREGION)
             DSP OLDCRLFT OLDCRBTM OLDWIDTH OLDHEIGHT NIL 'REPLACE)
      (COND
         ((IGREATERP (SETQ NEWPRIGHT (IPLUS NEWCRLFT NEWWID))
                  (SETQ OLDPRIGHT (IPLUS OLDCRLFT OLDWIDTH)))  ; call the display function on the newly exposed right area.
          (REDISPLAYW WINDOW (create REGION
                                     LEFT _ OLDPRIGHT
                                     BOTTOM _ OLDCRBTM
                                     WIDTH _ (IDIFFERENCE NEWPRIGHT OLDPRIGHT)
                                     HEIGHT _ OLDHEIGHT)
                      T)))
      (COND
         ((IGREATERP OLDCRBTM NEWCRBTM)                       ; call the display function on the newly exposed LEFT area.
          (REDISPLAYW WINDOW (create REGION
                                     LEFT _ NEWCRLFT
                                     BOTTOM _ NEWCRBTM
                                     WIDTH _ NEWWID
                                     HEIGHT _ (IDIFFERENCE OLDCRBTM NEWCRBTM))
                      T)))
      (RETURN WINDOW])
```

(\INBETWEENP

```
  [LAMBDA (X LFT RGHT)                                        (* rrb "11-Oct-84 17:07")
```

;; returns T if X is between LEFT and RIGHT

```
    (AND (GEQ X LFT)
         (GREATERP RGHT X])
```

(DECODE/WINDOW/OR/DISPLAYSTREAM

```
[LAMBDA (DSORW WINDOWVAR TITLE BORDER)                    ; Edited 24-Sep-92 12:32 by jds
```

;; provides a defaulting mechanism for display-streams that uses windows too. If DSORW is NIL, it uses the value of WINDOWVAR and if DSORW ;; is NEW, it creates a new one.

```
(COND
   ((DISPLAYSTREAMP DSORW))
   ((WINDOWP DSORW)
    (OPENW DSORW)
    (AND TITLE (NOT (EQUAL TITLE (fetch (WINDOW WTITLE) of DSORW)))
         (WINDOWPROP DSORW 'TITLE TITLE))
    (AND BORDER (WINDOWPROP DSORW 'BORDER BORDER))
    (fetch (WINDOW DSP) of DSORW))
   [(NULL DSORW)
    (fetch (WINDOW DSP) of (PROG ((WINDOW (EVALV WINDOWVAR)))
                                 (RETURN (COND
                                            ((WINDOWP WINDOW)
                                             (OPENW WINDOW)
                                             (AND TITLE (NOT (EQUAL TITLE (fetch (WINDOW WTITLE) of WINDOW)))
                                                  (WINDOWPROP WINDOW 'TITLE TITLE))
                                             (AND BORDER (WINDOWPROP WINDOW 'BORDER BORDER))
                                            WINDOW)
                                           (T (SET WINDOWVAR (CREATEW NIL TITLE BORDER]
   [(EQ DSORW 'NEW)
    (fetch (WINDOW DSP) of (SET WINDOWVAR (CREATEW NIL TITLE BORDER]
   (T (ERROR "Illegal args" (LIST DSORW WINDOWVAR])
```

(GROW/REGION

```
[LAMBDA (REGION AMOUNT)                                   (* rrb "19-OCT-83 11:18")
```

;; increase REGION by amount in all directions

```
(CREATEREGION (IDIFFERENCE (fetch (REGION LEFT) of REGION)
                    AMOUNT)
       (IDIFFERENCE (fetch (REGION BOTTOM) of REGION)
              AMOUNT)
       (IPLUS (fetch (REGION WIDTH) of REGION)
              (SETQ AMOUNT (ITIMES AMOUNT 2)))
       (IPLUS (fetch (REGION HEIGHT) of REGION)
              AMOUNT])
```

(CLRPROMPT

```
[LAMBDA NIL                                               ; Edited 7-Mar-94 11:55 by sybalsky
```

;; clears the prompt window

```
(LET ((P PROMPTWINDOW))
     (if P
         then (COND
                 ((type? WINDOW P)
                  (DSPRESET P))
                 (T (TERPRI P)
                    (TERPRI P])
```

(PROMPTPRINT

```
[LAMBDA N                                                 ; Edited 7-Mar-94 11:55 by sybalsky
   (CLRPROMPT)
   (for I from 1 to N do (PRIN1 (ARG N I)
                                PROMPTWINDOW])
```

(OPENWINDOWS

```
[LAMBDA (SCREEN)                                          (* kbr%: " 4-Aug-85 16:34")
```

;; returns a list of all open windows

```
(PROG (WINDOW WINDOWS)
      (COND
         ((EQ SCREEN T)                                   ; Return all open windows.
          (SETQ WINDOWS (for SCREEN in \SCREENS join (OPENWINDOWS SCREEN)))
          (RETURN WINDOWS)))
      (SETQ SCREEN (\INSURESCREEN SCREEN))
      (SETQ WINDOW (fetch (SCREEN SCTOPW) of SCREEN))
      (while WINDOW do (SETQ WINDOWS (CONS WINDOW WINDOWS))
                       (SETQ WINDOW (fetch (WINDOW NEXTW) of WINDOW)))
      (SETQ WINDOWS (DREVERSE WINDOWS))
      (RETURN WINDOWS])
```

(\INSUREWINDOW

```
[LAMBDA (WINDOW)                                          (* rmk%: " 1-SEP-83 10:25")
```

;; coerces to a window

```
    (COND
       ((type? WINDOW WINDOW)
        WINDOW)
       ((AND (DISPLAYSTREAMP (\OUTSTREAMARG WINDOW T))
             (WFROMDS WINDOW)))
       (T (\ILLEGAL.ARG WINDOW])

)

;; these entries are left in for backward compatibility. They were dedocumented 6/83.  rrb

(MOVD 'OPENWP 'ACTIVEWP)

(DEFINEQ

(OVERLAPPINGWINDOWS
  [LAMBDA (WINDOW)                                          (* gbn%: "25-Jan-86 15:52")
    ;; returns all windows that overlap with WINDOW or that overlap a window that is in the OVERLAPPINGWINDOWS of WINDOW.
    (PROG (WPTR OVERLAPS DONTS)
          (SETQ WPTR (fetch (SCREEN SCTOPW) of (fetch (WINDOW SCREEN) of WINDOW)))
          (SETQ OVERLAPS (CONS WINDOW (ALLATTACHEDWINDOWS WINDOW)))
      LP  [COND
             ((NULL WPTR)
              (RETURN OVERLAPS))
             ((MEMB WPTR OVERLAPS)                          ; skip the window itself
              NIL)
             ([SOME OVERLAPS (FUNCTION (LAMBDA (X)
                                       (WOVERLAPP WPTR X]   ; this window overlaps a member of the interesting ones.
              (SETQ OVERLAPS (CONS WPTR OVERLAPS))
              ;; find all members of donts that overlap this new window and move them {and ones that overlap them} to OVERLAPS.
              (PROG ((ADDS (CONS WPTR))
                     OVERLAPPED)
                NWLP
                    (COND
                       ((for old OVERLAPPED in DONTS thereis (WOVERLAPP (CAR ADDS)
                                                                        OVERLAPPED))
                                                           ; the window that was added overlaps one of the previously
                                                           ; looked at windows that was untouched.
                        (SETQ ADDS (CONS OVERLAPPED ADDS))
                        (SETQ OVERLAPS (CONS OVERLAPPED OVERLAPS))
                        (SETQ DONTS (REMOVE OVERLAPPED DONTS))
                        (GO NWLP))
                       ((SETQ ADDS (CDR ADDS))          ; there are more windows that were added.
                        (GO NWLP)))
                    (RETURN)))
             (T (SETQ DONTS (CONS WPTR DONTS]
          (SETQ WPTR (fetch (WINDOW NEXTW) of WPTR))
          (GO LP])

(WOVERLAPP
  [LAMBDA (W1 W2)                                           (* rrb "16-AUG-81 08:30")
    ;; do these windows overlap?
    (REGIONSINTERSECTP (fetch (WINDOW REG) of W1)
           (fetch (WINDOW REG) of W2])

(ORDERFROMBOTTOMTOTOP
  [LAMBDA (WLST)                                            (* gbn%: "25-Jan-86 15:56")
    ;; returns a list of windows in order from bottom to top
    (PROG (ANS WPTR)
          (COND
             ((NULL WLST)
              (RETURN NIL)))
          [SETQ WPTR (fetch (SCREEN SCTOPW) of (fetch (WINDOW SCREEN) of (CAR WLST]
                                                           ; start at the topw
      LP  [COND
             ((NULL WPTR)
              (RETURN ANS))
             ((FMEMB WPTR WLST)
              (SETQ ANS (CONS WPTR ANS]
          (SETQ WPTR (fetch (WINDOW NEXTW) of WPTR))
          (GO LP])

)

;; screen size changing functions.

(DEFINEQ

(\ONSCREENW
```

```
  [LAMBDA (W)                                                    (* kbr%: "18-Jan-86 18:40")
    ;; does W have any part on the screen?
    ;; for now only consider that it might be too far to the right as this is the wide to narrow screen case.
    ;; HARDCURSORWIDTH is to make sure the cursor can be set in the window.  It can be taken out when cursor hotspot can go anywhere.
    (IGREATERP (IDIFFERENCE (fetch (SCREEN SCWIDTH) of (fetch (WINDOW SCREEN) of W))
                      HARDCURSORWIDTH)
           (fetch (REGION LEFT) of (WINDOWPROP W 'REGION])
```

(\PUTONSCREENW

```
  [LAMBDA (W)                                                    (* kbr%: "26-Mar-85 23:29")
    ;; moves W so that it will be on the screen.  For now, moves it to the left by screenwidth
    (MOVEW W (create POSITION
                     XCOORD _ (IDIFFERENCE (fetch (REGION LEFT) of (fetch (WINDOW REG) of W))
                                           (fetch (SCREEN SCWIDTH) of (fetch (WINDOW SCREEN) of W)))
                     YCOORD _ (fetch (REGION BOTTOM) of (WINDOWPROP W 'REGION])
```

(\UPDATECACHEDFIELDS

```
  [LAMBDA (DS)                                                   (* rrb "14-OCT-81 16:53")
    ;; updates the cached fields of a displaystream for the fact that the screen bitmap changed sizes
    (\SFFixDestination DS])
```

(\WWCHANGESCREENSIZE

```
  [LAMBDA (SCREEN)                                               (* lmm "16-Nov-86 05:04")
    ;; the sysout has been moved to a screen of a different size.  All windows are closed, the screenbitmap is updated to correct new size and the
    ;; windows are reopened so that at least part of each is visible.
    (PROG (WINDOWS)
          (SETQ SCREEN (\INSURESCREEN SCREEN))
          (SETQ WINDOWS (DREVERSE (OPENWINDOWS SCREEN)))       ; OPENWINDOWS returns the windows with bottom window first.
          (for W in WINDOWS do (\CLOSEW1 W))
          (\STARTDISPLAY)
          (\CLEARBM (fetch (SCREEN SCDESTINATION) of SCREEN)
                 WINDOWBACKGROUNDSHADE)                        ; update cached bitmap width information that is in the display
                                                               ; streams
          [for W in WINDOWS do (\UPDATECACHEDFIELDS (WINDOWPROP W 'DSP]
                                                               ; bring back windows
          (for W in (REVERSE WINDOWS) do (COND
                                             ((NOT (\ONSCREENW W))
                                              (\PUTONSCREENW W)))
                                         (OPENW W])
```

(CREATEWFROMIMAGE

```
  [LAMBDA (IMAGE SCREEN)                                         (* gbn%: "25-Jan-86 16:05")
    ;; creates a window that has IMAGE (a bitmap) as an image.  It is initially closed and can be opened.
    (PROG (WINDOW)
          (SETQ WINDOW (CREATEW (create SCREENREGION
                                        SCREEN _ (\INSURESCREEN SCREEN)
                                        LEFT _ 0
                                        BOTTOM _ 0
                                        WIDTH _ (BITMAPWIDTH IMAGE)
                                        HEIGHT _ (BITMAPHEIGHT IMAGE))
                                NIL 0 T))
          [WINDOWPROP WINDOW 'MINSIZE (CONS (IMIN MinWindowWidth (BITMAPWIDTH IMAGE))
                                            (IMIN MinWindowWidth (BITMAPHEIGHT IMAGE]
          (BITBLT IMAGE 0 0 (fetch (WINDOW SAVE) of WINDOW))
          (RETURN WINDOW])
```

(UPDATEWFROMIMAGE

```
  [LAMBDA (WINDOW)                                               ; Edited 20-Aug-91 18:05 by jds
    ;; makes the fields of a window consistent with its image.
    (PROG ((REGION (fetch (WINDOW REG) of WINDOW))
           (IMAGE (fetch (WINDOW SAVE) of WINDOW)))
          (replace (REGION LEFT) of REGION with 0)
          (replace (REGION BOTTOM) of REGION with 0)
          (replace (REGION WIDTH) of REGION with (BITMAPWIDTH IMAGE))
          (replace (REGION HEIGHT) of REGION with (BITMAPHEIGHT IMAGE])

)

;; MEDLEY-NATIVE-WINDOWS INTERFACE FUNCTIONS

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \SCREENS \SCREENTYPES)
)
```

```
(RPAQ? \SCREENS )

(RPAQ? \SCREENTYPES '((1 MEDLEY OPEN-SCREEN CREATESCREEN CLOSE-SCREEN NILL)
                      (2 MEDLEY-COLOR-4)
                      (4 MEDLEY-COLOR-8)
                      y
                      (8 MEDLEY-COLOR-24)
                      (16 X-MONO)
                      (32 X-COLOR)
                      (64 MS-WINDOWS)))
```

;; OLD-MEDLEY-SCREEN window management functions

```
(DEFINEQ
```

(\MEDW.CREATEW

```
  [LAMBDA (SCREEN REGION TITLE BORDERSIZE NOOPENFLG PROPS OLDWINDOW)
                                                        ; Edited 28-Dec-93 15:12 by nilsson
    ;; creates and returns a window. If OLDWINDOW is defined this method has to reuse OLDWINDOW. This helps us open old windows on new
    ;; screens.
    (LET ((DSP (if OLDWINDOW
                   then (DSPCREATE SCREEN (fetch (WINDOW DSP) of OLDWINDOW))
                 else (DSPCREATE SCREEN)))
          DISPLAYDATA WINDOW)
         (SETQ DISPLAYDATA (fetch (STREAM IMAGEDATA) of DSP))
         [SETQ WINDOW (OR OLDWINDOW (create WINDOW
                                            REG _ REGION
                                            SAVE _ (BITMAPCREATE (fetch (REGION WIDTH) of REGION)
                                                                 (fetch (REGION HEIGHT) of REGION)
                                                                 (BITSPERPIXEL (fetch (SCREEN SCDESTINATION) of SCREEN)))
                                            WTITLE _ TITLE
                                            WBORDER _ BORDERSIZE
                                            NEXTW _ 'CLOSED]
         (replace (WINDOW SCREEN) of WINDOW with SCREEN)
         (replace (WINDOW DSP) of WINDOW with DSP)
         (replace (\DISPLAYDATA XWINDOWHINT) of DISPLAYDATA with WINDOW)
                                                        ; make the display stream and window agree about dimensions.
         (if OLDWINDOW
             then (LET ((R (fetch (WINDOW REG) of OLDWINDOW))
                        (TWICEBORDER (UNFOLD BORDERSIZE 2)))
                    ;; OLDWINDOW was defined. We have to recalculate the clippingregion since some screens (notably X) uses the clipping
                    ;; region relative to the window instead of relative to the screen
                    (DSPXOFFSET (IPLUS (fetch (REGION LEFT) of R)
                                       BORDERSIZE)
                                DSP)
                    (DSPYOFFSET (IPLUS (fetch (REGION BOTTOM) of R)
                                       BORDERSIZE)
                                DSP)
                    (DSPCLIPPINGREGION [create REGION
                                               LEFT _ 0
                                               BOTTOM _ 0
                                               WIDTH _ (IDIFFERENCE (fetch (REGION WIDTH) of R)
                                                                    TWICEBORDER)
                                               HEIGHT _ (IPLUS (IDIFFERENCE (fetch (REGION HEIGHT) of R)
                                                                            TWICEBORDER)
                                                               (COND
                                                                  [(fetch (WINDOW WTITLE) of OLDWINDOW)
                                                                   (DSPLINEFEED NIL (fetch (SCREEN SCTITLEDS)
                                                                                           of (fetch (WINDOW SCREEN)
                                                                                                     of OLDWINDOW]
                                                                  (T 0]
                                DSP))
           else (ADVISEWDS WINDOW)
                (MOVETOUPPERLEFT WINDOW)
                (SHOWWFRAME WINDOW))
         (COND
            ((NOT NOOPENFLG)
             (OPENW WINDOW)))
         WINDOW])
```

(\MEDW.OPENW

```
  [LAMBDA (SCREEN WINDOW)                               ; Edited 25-Apr-94 10:12 by sybalsky
    ;; opens a window by putting on the window stack and putting its bits on the screen.  Returns the window if it was actually opened.
    ;; If already open, punt.
    (if (EQ (fetch (WINDOW NEXTW) of WINDOW)
            'CLOSED)
        then (LET (DD)
                  (UNINTERRUPTABLY
                      (replace (WINDOW NEXTW) of WINDOW with (fetch (SCREEN SCTOPW) of SCREEN))
                      (replace (SCREEN SCTOPW) of SCREEN with WINDOW)
                      (SETQ \TOPWDS (fetch (WINDOW DSP) of WINDOW))
                                                        ; DSP of a window is guaranteed to be a display-stream
```

```
                    (SETQ DD (fetch (STREAM IMAGEDATA) of \TOPWDS))
                                                        ; Just in case screen width has changed.
                    (replace (PILOTBBT PBTDESTBPL) of (fetch (\DISPLAYDATA DDPILOTBBT) of DD)
                       with (UNFOLD (fetch (BITMAP BITMAPRASTERWIDTH) of (fetch (SCREEN SCDESTINATION)
                                                                            of SCREEN))
                                    BITSPERWORD))
                    (.WHILE.TOP.DS. \TOPWDS (\SW2BM (fetch (\DISPLAYDATA DDDestination) of DD)
                                                    (fetch (WINDOW REG) of WINDOW)
                                                    (fetch (WINDOW SAVE) of WINDOW)
                                                    NIL)))])
```

(**MEDW.CLOSEW**

```
[LAMBDA (SCREEN WINDOW)                                 ; Edited 25-Apr-94 10:07 by sybalsky
```

;; Do the actual closing operation for Medley windows.

```
(LET (NEXTW)
     (COND
        ((NOT (EQ \TOPWDS (FETCH (WINDOW DSP) OF WINDOW)))  ; This window isn't on top, so we want to bring it there WITHOUT
                                                            ; running topfns.
         (\TOTOPWDS (FETCH (WINDOW DSP) OF WINDOW)
                T)))
     (.WHILE.TOP.DS. \TOPWDS (\SW2BM (fetch (SCREEN SCDESTINATION) of SCREEN)
                                     (fetch (WINDOW REG) of WINDOW)
                                     (fetch (WINDOW SAVE) of WINDOW)
                                     NIL)
          (SETQ NEXTW (fetch (WINDOW NEXTW) of WINDOW))
          (replace (SCREEN SCTOPW) of SCREEN with NEXTW)
          [SETQ \TOPWDS (COND
                           (NEXTW (fetch (WINDOW DSP) of NEXTW]
                                                            ; smash the window's link to other's in the chain.
          (replace (WINDOW NEXTW) of WINDOW with 'CLOSED])
```

(**MEDW.MOVEW**

```
[LAMBDA (SCREEN WINDOW POSorX Y)                        ; Edited 27-Sep-93 10:23 by jds
```

;; moves a window. If window is closed and position is given, it won't open the window. It also calls the window's MOVEFN property.

```
(SETQ WINDOW (\INSUREWINDOW WINDOW))
(PROG ((OLDREGION (fetch (WINDOW REG) of WINDOW))
       (USERMOVEFN (fetch (WINDOW MOVEFN) of WINDOW))
       (OPEN? (OPENWP WINDOW))
       OLDSCREEN POS NEWREGION OLDLEFT OLDBOTTOM OLDWIDTH OLDHEIGHT OLDCLIPREGION LFT BTM REG FN)
      (SETQ OLDSCREEN (fetch (WINDOW SCREEN) of WINDOW))
      (COND
         ([COND
             ((LISTP USERMOVEFN)
              (FMEMB 'DON'T USERMOVEFN))
             (T (EQ USERMOVEFN 'DON'T]
          (PROMPTPRINT "This window cannot be moved.")
          (RETURN)))
      [COND
         ((NOT (SUBREGIONP OLDREGION (fetch (SCREEN SCREGION) of OLDSCREEN)))
                                                        ; use T as an indication that the window was completely off
                                                        ; screen.
          (SETQ OLDCLIPREGION (OR (\ONSCREENCLIPPINGREGION WINDOW)
                                  T]
      (SETQ OLDLEFT (fetch (REGION LEFT) of OLDREGION))
      (SETQ OLDBOTTOM (ffetch (REGION BOTTOM) of OLDREGION))
      (SETQ OLDWIDTH (ffetch (REGION WIDTH) of OLDREGION))
      (SETQ OLDHEIGHT (ffetch (REGION HEIGHT) of OLDREGION))
      (COND
         ([AND POSorX (SETQ POS (COND
                                   ((POSITIONP POSorX)
                                    POSorX)
                                   [(NUMBERP POSorX)
                                    (COND
                                       ((NUMBERP Y)
                                        (create POSITION
                                                XCOORD _ POSorX
                                                YCOORD _ Y))
                                       (T (\ILLEGAL.ARG Y]
                                   ((REGIONP POSorX)
                                    (create POSITION
                                            XCOORD _ (fetch (REGION LEFT) of POSorX)
                                            YCOORD _ (fetch (REGION BOTTOM) of POSorX)))
                                   (T (\ILLEGAL.ARG POSorX]  ; if not aready open, don't
          (AND OPEN? (TOTOPW WINDOW)))
          (T                                            ; no position to move to has been given, ask user for one.
             (TOTOPW WINDOW)                            ; TOTOPW opens the window if it is not already.
             [COND
                [[AND (SETQ FN (WINDOWPROP WINDOW 'CALCULATEREGIONFN))
                      (SETQ REG (APPLY* FN WINDOW '\MEDW.MOVEW]
                                                        ; prompt with a region that is calculated by the window
                 [SETQ POS (GETBOXPOSITION (fetch (REGION WIDTH) of REG)
                                           (ffetch (REGION HEIGHT) of REG)
                                           (SETQ LFT (ffetch (REGION LEFT) of REG))
```

```
                                    (SETQ BTM (ffetch (REGION BOTTOM) of REG]
                ;; use a position that is offset by the same amount as the calculated region was from the window's region.
                (SETQ POS (create POSITION
                                  XCOORD _ (IPLUS (fetch (POSITION XCOORD) of POS)
                                                  (IDIFFERENCE OLDLEFT LFT))
                                  YCOORD _ (IPLUS (ffetch (POSITION YCOORD) of POS)
                                                  (IDIFFERENCE OLDBOTTOM BTM]
             (T (SETQ POS (GETBOXPOSITION OLDWIDTH OLDHEIGHT OLDLEFT OLDBOTTOM]
          (SETQ OPEN? T)))
      [COND
         ((AND (LISTP USERMOVEFN)
               (NOT (FMEMB (CAR USERMOVEFN)
                           LAMBDASPLST)))
          (AND (EQ [for MFN in USERMOVEFN do (SETQ NEWREGION (APPLY* MFN WINDOW POS))
                                            (COND
                                               ((EQ NEWREGION 'DON'T)
                                                (RETURN 'DON'T))
                                               ((POSITIONP NEWREGION)
                                                (SETQ POS NEWREGION]
                   'DON'T)
               (RETURN)))
         (USERMOVEFN (SETQ NEWREGION (APPLY* USERMOVEFN WINDOW POS))
                (COND
                   ((EQ NEWREGION 'DON'T)
                    (RETURN))
                   ((POSITIONP NEWREGION)
                    (SETQ POS NEWREGION]
      (COND
         ((OR (NOT (EQ (fetch (POSITION XCOORD) of POS)
                       OLDLEFT))
              (NOT (EQ (ffetch (POSITION YCOORD) of POS)
                       OLDBOTTOM)))
          (SETQ NEWREGION (create REGION
                                  LEFT _ (ffetch (POSITION XCOORD) of POS)
                                  BOTTOM _ (ffetch (POSITION YCOORD) of POS)
                                  WIDTH _ OLDWIDTH
                                  HEIGHT _ OLDHEIGHT))
          (UNINTERRUPTABLY
              [COND
                 (OPEN? ;; if window is open, move it to top as its MOVEFN may have changed things and swap its bits to its new location
                        (.WHILE.TOP.DS. WINDOW (\SW2BM (fetch (SCREEN SCDESTINATION) of OLDSCREEN)
                                                       OLDREGION
                                                       (fetch (WINDOW SAVE) of WINDOW)
                                                       NIL)
                                        (\SW2BM (ffetch (WINDOW SAVE) of WINDOW)
                                                NIL
                                                (ffetch (SCREEN SCDESTINATION) of OLDSCREEN)
                                                NEWREGION]
              (replace (WINDOW REG) of WINDOW with NEWREGION)
              (ADVISEWDS WINDOW OLDREGION T))
          [COND
             ((AND OPEN? (WINDOWPROP WINDOW 'REPAINTFN)
                   OLDCLIPREGION)                          ; redisplay those parts that were off the screen.
              (COND
                 ((EQ OLDCLIPREGION T)                     ; whole window was off.
                  (REDISPLAYW WINDOW NIL T))
                 (T (PROG (NEWCLIPPINGREGION NCL OCL NCB OCB OCR NCR OCW NCW OCH NCH OCT NCT)
                          (SETQ NEWCLIPPINGREGION (\ONSCREENCLIPPINGREGION WINDOW))
                                                           ; the title may be the only thing now on the screen.
                          (OR NEWCLIPPINGREGION (RETURN))
                          (SETQ NCB (fetch (REGION BOTTOM) of NEWCLIPPINGREGION))
                          (SETQ OCB (fetch (REGION BOTTOM) of OLDCLIPREGION))
                          (SETQ OCW (ffetch (REGION WIDTH) of OLDCLIPREGION))
                          (SETQ NCW (ffetch (REGION WIDTH) of NEWCLIPPINGREGION))
                          (SETQ OCH (ffetch (REGION HEIGHT) of OLDCLIPREGION))
                          (SETQ NCH (ffetch (REGION HEIGHT) of NEWCLIPPINGREGION))
                          [COND
                             ((ILESSP (SETQ NCL (ffetch (REGION LEFT) of NEWCLIPPINGREGION))
                                      (SETQ OCL (ffetch (REGION LEFT) of OLDCLIPREGION)))
                              (REDISPLAYW WINDOW (CREATEREGION NCL OCB (IDIFFERENCE OCL NCL)
                                                               OCH]
                          [COND
                             ((ILESSP (SETQ OCR (IPLUS OCL OCW))
                                      (SETQ NCR (IPLUS NCL NCW)))
                                                           ; some stuff appeared from the right.
                              (REDISPLAYW WINDOW (CREATEREGION OCR OCB (IDIFFERENCE NCR OCR)
                                                               OCH]
                          [COND
                             ((ILESSP NCB OCB)
                              (REDISPLAYW WINDOW (CREATEREGION NCL NCB NCW (IDIFFERENCE OCB NCB]
                          [COND
                             ((ILESSP (SETQ OCT (IPLUS OCB OCH))
                                      (SETQ NCT (IPLUS NCB NCH)))
                                                           ; some stuff appeared from the top
                              (REDISPLAYW WINDOW (CREATEREGION NCL OCT NCW (IDIFFERENCE NCT OCT]
```

```
                              (COND
                                 ((IGREATERP (IPLUS OLDBOTTOM OLDHEIGHT)
                                         (fetch (SCREEN SCHEIGHT) of OLDSCREEN))
                                                                  ; should reshow the title but don't have any entry for that.
                                  NIL]
                        (DOUSERFNS (WINDOWPROP WINDOW 'AFTERMOVEFN)
                                   WINDOW)))
                  (RETURN POS])
```

(\MEDW.RELMOVEW

```
[LAMBDA (SCREEN WINDOW POS)                                       ; Edited 18-Nov-94 13:51 by jds
```

;; Move WINDOW by relative DX DY

```
(PROG [(WINREG (WINDOWPROP WINDOW 'REGION]
      (MOVEW WINDOW (create POSITION
                           XCOORD _ (IPLUS (fetch (REGION LEFT) of WINREG)
                                           (fetch (POSITION XCOORD) of POS))
                           YCOORD _ (IPLUS (fetch (REGION BOTTOM) of WINREG)
                                           (fetch (POSITION YCOORD) of POS])
```

(\MEDW.SHRINKW

```
[LAMBDA (SCREEN WINDOW TOWHAT ICONPOSITION EXPANDFN)              ; Edited 27-Sep-93 10:24 by jds
```

;; Create a small WINDOW which acts as an Icon of window. This 'icon window' provides a popup menu which will open the main WINDOW again,
;; and run the function EXPANDFN. TOWHAT can be a BITMAP which will be used to make a WINDOW image, an existing window, or a string
;; which will be printed in TITLE only icon window, or can be an existing window. If TOWHAT is NIL, the TITLE of the main WINDOW is used as
;; the TOWHAT for the icon.

```
(SETQ WINDOW (\INSUREWINDOW WINDOW))
(COND
   ((NOT (OPENWP WINDOW))
```

;; if it is not currently open, don't do anything. Maybe something should happen here but I don't understand what --- rrb

```
    NIL)
   ((WINDOWPROP WINDOW 'ICONFOR)                                  ; This is already an icon!
    NIL)
   ((EQ (DOUSERFNS (WINDOWPROP WINDOW 'SHRINKFN)
               WINDOW T)
        'DON'T)                                                   ; one of the shrinkfns disallowed the shrinkage.
    NIL)
   (T (LET (TITLE ICONW FN ICONISBITMAP)                          ; get the icon specification from the window if none is given.
           [SETQ ICONW (COND
                       ((type? BITMAP TOWHAT)                     ; use bitMap to create a WINDOW
                        [WINDOWPROP WINDOW 'ICON (SETQ TOWHAT (CREATEWFROMIMAGE (BITMAPCOPY
                                                                                  (SETQ ICONISBITMAP
                                                                                   TOWHAT))
                                                                                 (fetch (WINDOW SCREEN) of WINDOW]
                                                                  ; save the icon on the window so that next time it will shrink to
                                                                  ; the same thing.
                        TOWHAT)
                       ((WINDOWP TOWHAT)                          ; use given WINDOW as icon
                        (WINDOWPROP WINDOW 'ICON TOWHAT)
                                                                  ; save the icon on the window so that next time it will shrink to
                                                                  ; the same thing.
                        TOWHAT)
                       ((STRINGP TOWHAT)
                        [WINDOWPROP WINDOW 'ICON (SETQ TOWHAT (\DTEST (APPLY* DEFAULTICONFN WINDOW
                                                                              TOWHAT)
                                                                      'WINDOW]
                        TOWHAT)
                       (T                                         ; current call doesn't specify an icon window. Look for something
                                                                  ; on the window.
                          [SETQ TOWHAT (COND
                                          ((SETQ FN (WINDOWPROP WINDOW 'ICONFN))
                                                                  ; User fn to create an icon. Can return cached value
                                           (APPLY* FN WINDOW (WINDOWPROP WINDOW 'ICONWINDOW)
                                                  (POSITIONP ICONPOSITION)))
                                          (T (WINDOWPROP WINDOW 'ICON]
                          (COND
                             ((WINDOWP TOWHAT)                    ; use given WINDOW as icon
                              TOWHAT)
                             ((type? BITMAP TOWHAT)               ; use bitMap to create a WINDOW
                              (CREATEWFROMIMAGE (BITMAPCOPY (SETQ ICONISBITMAP TOWHAT))
                                     (fetch (WINDOW SCREEN) of WINDOW)))
                             (T
                               ;; Call default icon maker.  Note: don't store this as the ICON property, because we want it to be
                               ;; recomputed each time, because, for example, the window's title, from which the icon text is derived,
                               ;; might change.  Not a problem for windows that have an ICONFN because then the ICONFN is
                               ;; responsible for keeping it up to date
                                (\DTEST (APPLY* DEFAULTICONFN WINDOW TOWHAT)
                                       'WINDOW]
                (WINDOWPROP WINDOW 'ICONWINDOW ICONW)
                (WINDOWPROP ICONW 'ICONFOR WINDOW)                ; set up so that if icon is closed, main window will be also.
                (WINDOWADDFNPROP ICONW 'CLOSEFN (FUNCTION CLOSEMAINWINDOW))
                                                                  ; set up so that if main window is opened, icon is closed.
```

```
            [COND
               ((EQ (WINDOWPROP ICONW 'BUTTONEVENTFN)
                    'TOTOPW)                                    ; if the iconw doesn't have a buttoneventfn, give it one that the
                                                                ; middle expands it.
                (WINDOWPROP ICONW 'BUTTONEVENTFN (FUNCTION ICONBUTTONEVENTFN]
            (WINDOWADDFNPROP WINDOW 'OPENFN (FUNCTION CLOSEICONWINDOW))
            (WINDOWADDFNPROP ICONW 'MOVEFN (FUNCTION \NOTENEWICONPOSITION))
            (AND EXPANDFN (WINDOWADDFNPROP WINDOW 'EXPANDFN EXPANDFN))
            (WINDOWPROP ICONW 'DOWINDOWCOMFN (FUNCTION DOICONWINDOWCOM))
            [COND
               [(AND (NEQ ICONPOSITION 'SAME)
                     (OR ICONISBITMAP (POSITIONP ICONPOSITION)))
                ;; If ICONPOSITION given explicitly, or we derived the icon as a bitmap, need to move it into new position
                (MOVEW ICONW (COND
                                 ((POSITIONP ICONPOSITION)
                                  ICONPOSITION)
                                 ((PROG1 [POSITIONP (SETQ ICONPOSITION (WINDOWPROP WINDOW 'ICONPOSITION]
                                                                ; leave it in its current location.

                                         ))
                                 (T (SETQ ICONPOSITION (ICONPOSITION.FROM.WINDOW WINDOW (WINDOWPROP
                                                                                         ICONW
                                                                                         'REGION]
               (T (SETQ ICONPOSITION (LET [(REG (WINDOWPROP ICONW 'REGION]
                                          (create POSITION
                                                  XCOORD _ (fetch (REGION LEFT) of REG)
                                                  YCOORD _ (fetch (REGION BOTTOM) of REG]
            (WINDOWPROP WINDOW 'ICONPOSITION ICONPOSITION)
            (TOTOPW WINDOW T)
            ;; bring it to the top without callings its totopfns in case the shrinkfns brought another window to the top.
            (\CLOSEW1 WINDOW)
            (OPENW ICONW)
            ICONW])
```

(\MEDW.EXPANDW

```
  [LAMBDA (SCREEN ICONW)                                        ; Edited 27-Sep-93 10:24 by jds
```

;;; expands an icon window into its main window.

```
    (PROG ((IW ICONW)
           MAINWINDOW USEREXPANDFN EXPANDREGION)
          [COND
             [(SETQ MAINWINDOW (WINDOWPROP IW 'ICONFOR]
             ((SETQ IW (WINDOWPROP IW 'ICONWINDOW))             ; user has passed in the window to expand, not its icon.
              (COND
                 ((OPENWP (SETQ MAINWINDOW ICONW))              (* make sure the window is shrunken.)
                  (RETURN ICONW]
          (COND
             ([AND MAINWINDOW (NULL (\USERFNISDON'T (SETQ USEREXPANDFN (WINDOWPROP MAINWINDOW 'EXPANDFN]
              ;; if the main window will open and none of the expandfns stop it, open the main window and Close icon Window
              (if (AND (WINDOWPROP MAINWINDOW 'EXPANDREGIONFN)
                       (SETQ EXPANDREGION (APPLY* (WINDOWPROP MAINWINDOW 'EXPANDREGIONFN)
                                                  MAINWINDOW)))
                  then ;; there is an EXPANDREGIONFN to calculate a new region to expand into, and it didn't return NIL, so assume
                       ;; EXPANDREGION is a valid region.  SHAPE instead of just openning.  SHAPEW2 will open the window, ignoring an
                       ;; openfn or doshapefn, but allowing the reshapefns to run.
                       (\SHAPEW2 MAINWINDOW EXPANDREGION)
                else (\OPENW1 MAINWINDOW))
              (\CLOSEW1 IW)
              (WINDOWDELPROP MAINWINDOW 'OPENFN 'CLOSEICONWINDOW)
              (WINDOWDELPROP IW 'CLOSEFN 'CLOSEMAINWINDOW)      ; call the expand functions after the window has been opened.
              (DOUSERFNS USEREXPANDFN MAINWINDOW)               ; break link from icon to window.
              (RETURN (WINDOWPROP IW 'ICONFOR NIL])
```

(\MEDW.SHAPEW

```
  [LAMBDA (SCREEN WINDOW NEWREGION)                             ; Edited 27-Sep-93 10:25 by jds
```

```
    ;; entry that shapes a window checking the userfns for DON'T and interacting to get a region if necessary.  This also checks for a user function to
    ;; do the actual reshaping.  look for a function on windowprop INITCORNERSFN, which will take the window and return the initcorners for the
    ;; window, to be passed to getregion.

    (SETQ WINDOW (\INSUREWINDOW WINDOW))
    (PROG ((OLDSIZE (WINDOWPROP WINDOW 'REGION))
           NEWSIZE)
          (COND
             ((\USERFNISDON'T (fetch (WINDOW RESHAPEFN) of WINDOW))
                                                                ; don't allow the window to be reshaped.
              (PROMPTPRINT "This window cannot be reshaped.")
              (RETURN NIL)))
          (SETQ NEWSIZE (MINIMUMWINDOWSIZE WINDOW))             ; Start with the minimum allowable size.
          [SETQ NEWSIZE (COND
                           (NEWREGION                           ; An explicit new region was specified; make sure it's big enough.
```

```
                              (COND
                                 [(OR (LESSP (fetch (REGION WIDTH) of NEWREGION)
                                             (CAR NEWSIZE))
                                      (LESSP (fetch (REGION HEIGHT) of NEWREGION)
                                             (CDR NEWSIZE))) ; given a region that is too small, so expand the width and height
                                                             ; to at least the minima.
                                  (CREATEREGION (fetch (REGION LEFT) of NEWREGION)
                                         (fetch (REGION BOTTOM) of NEWREGION)
                                         (IMAX (CAR NEWSIZE)
                                               (fetch (REGION WIDTH) of NEWREGION))
                                         (IMAX (CDR NEWSIZE)
                                               (fetch (REGION HEIGHT) of NEWREGION]
                                  (T NEWREGION)))
                           ((WINDOWPROP WINDOW 'INITCORNERSFN)   ; There's an INITCORNERSFN.  Fire it up and prompt the user
                                                                 ; for a  new shape.
                            (GETREGION (CAR NEWSIZE)
                                   (CDR NEWSIZE)
                                   (WINDOWREGION WINDOW '\MEDW.SHAPEW)
                                   (fetch (WINDOW NEWREGIONFN) of WINDOW)
                                   WINDOW
                                   (APPLY* (WINDOWPROP WINDOW 'INITCORNERSFN)
                                          WINDOW)))
                           (T                                    ; Just go prompt the user for a new shape.
                              (GETREGION (CAR NEWSIZE)
                                     (CDR NEWSIZE)
                                     (WINDOWREGION WINDOW '\MEDW.SHAPEW)
                                     (fetch (WINDOW NEWREGIONFN) of WINDOW)
                                     WINDOW]
        (RETURN (COND
                   ((EQUAL NEWSIZE OLDSIZE)
                    ;; if same size and place as before, do nothing
                    NIL)
                   ((AND (EQ (fetch (REGION WIDTH) of NEWSIZE)
                             (fetch (REGION WIDTH) of OLDSIZE))
                         (EQ (fetch (REGION HEIGHT) of NEWSIZE)
                             (fetch (REGION HEIGHT) of OLDSIZE)))
                    ;; if same width and height, then optimize to a move
                    (MOVEW WINDOW (fetch (REGION LEFT) of NEWSIZE)
                           (fetch (REGION BOTTOM) of NEWSIZE)))
                   (T ;; do the shape, checking for a doshapefn
                      (APPLY* (OR (WINDOWPROP WINDOW 'DOSHAPEFN)
                                  'SHAPEW1)
                              WINDOW
                              (COPYALL NEWSIZE])
```

(\MEDW.REDISPLAYW

```
[LAMBDA (SCREEN WINDOW REGION ALWAYSFLG)                         ; Edited 27-Sep-93 10:26 by jds
```

;; calls a repaint function after setting the clipping region of the window to it. If ALWAYSFLG is NIL, it won't redisplay unless there is a window ;; repaintfn.

```
(PROG ((DSP (fetch (WINDOW DSP) of WINDOW))
       REPAINTFN CLIPREG)
      (COND
         [(SETQ REPAINTFN (WINDOWPROP WINDOW 'REPAINTFN]
         (ALWAYSFLG (SETQ REPAINTFN (FUNCTION NILL)))
         (T (PROMPTPRINT "Window has no REPAINTFN.  Can't redisplay.")
            (RETURN)))
      (SETQ CLIPREG (DSPCLIPPINGREGION NIL DSP))
      (RETURN (COND
                 (REGION [COND
                            ((NOT (SUBREGIONP CLIPREG REGION))
                                                             ; reduce REGION so that it is within the clipping region of the
                                                             ; window
                             (OR (SETQ REGION (INTERSECTREGIONS REGION CLIPREG))
                                 (RETURN]
                  (RESETLST
                      (RESETSAVE NIL (LIST 'DSPCLIPPINGREGION (DSPCLIPPINGREGION REGION DSP)
                                           DSP))
                      (RESETSAVE NIL (LIST 'DSPXOFFSET (DSPXOFFSET NIL DSP)
                                           DSP))
                      (RESETSAVE NIL (LIST 'DSPYOFFSET (DSPYOFFSET NIL DSP)
                                           DSP))
                      (FILLWITHBACKGROUND WINDOW REGION)
                      (DOUSERFNS2 REPAINTFN WINDOW REGION)))
                 (T (FILLWITHBACKGROUND WINDOW REGION)
                    (DOUSERFNS2 REPAINTFN WINDOW CLIPREG])
```

(\MEDW.BURYW

```
[LAMBDA (SCREEN WINDOW)                                          ; Edited 27-Sep-93 10:26 by jds
```

;; HACK: Puts WINDOW at the bottom by putting everything that touches it to the top!

```
(SETQ WINDOW (\INSUREWINDOW WINDOW))
```

```
(PROG ((OVERLAPPINGWINDOWS (ORDERFROMBOTTOMTOTOP (OVERLAPPINGWINDOWS WINDOW)))
       ABOVEWINDOWS ATWINS)
      [SETQ ABOVEWINDOWS (REMOVE WINDOW (LDIFFERENCE OVERLAPPINGWINDOWS (SETQ ATWINS (ALLATTACHEDWINDOWS
                                                                                      WINDOW]
```

;; close them in order from the top. This should be the fastest since they would have to come to the top to be closed anyway.

```
      (for W in (REVERSE OVERLAPPINGWINDOWS) do (\CLOSEW1 W))
      (\OPENW1 WINDOW)                                    ; put attached windows below the other windows.
      (for W in ATWINS do (\OPENW1 W))                    ; finally open the other windows.
      (for W in ABOVEWINDOWS do (\OPENW1 W))
      (RETURN WINDOW])
```

(\MEDW.TOTOPW

```
[LAMBDA (SCREEN WINDOW NOCALLTOTOPFNFLG)                  ; Edited 27-Sep-93 10:27 by jds
```

;; user entry to bring a window to the top. Unless NOCALLTOTOPFNFLG is non-NIL, it will call the windows TOTOPFN

```
(SETQ WINDOW (\INSUREWINDOW WINDOW))
(COND
   ((EQ WINDOW (fetch (SCREEN SCTOPW) of (fetch (WINDOW SCREEN) of WINDOW)))
    (PROGN                                                (* (SETQ \TOPWDS (fetch (WINDOW DSP) of WINDOW)))
           NIL))
   ((OPENWP WINDOW)
    (OR NOCALLTOTOPFNFLG (DOUSERFNS (WINDOWPROP WINDOW 'TOTOPFN)
                                    WINDOW))
    (\INTERNALTOTOPW WINDOW))
   ((OPENW WINDOW)                                        ; if it is not open, open it and then call the TOTOPFN
    (OR NOCALLTOTOPFNFLG (DOUSERFNS (WINDOWPROP WINDOW 'TOTOPFN)
                                    WINDOW)))
   (T                                                     ; window won't open probably because of DON'T OPENFN
      (ERROR "Window won't open; Can't be bring to top." WINDOW)))
WINDOW])
```

(\MEDW.DSPCREATE

```
[LAMBDA (SCREEN DESTINATION OLDDSP)                       ; Edited 9-Jul-2022 10:48 by rmk
                                                          ; Edited 2-Aug-2021 00:44 by rmk:
```

;; MEDLEY-WINDOW-SPECIFIC version of DSPCREATE. This is what gets called by dispatch from \GENERIC.DSPCREATE. If provided, ;; OLDDSP can be created on a new screen.

;; Creates a stream-of-type-display on the DESTINATION bitmap or display device

```
(\COMMON.DSPCREATE (OR (BITMAPP (fetch (SCREEN SCDESTINATION) of SCREEN))
                       (BITMAPP DESTINATION)
                       ScreenBitMap)
       (fetch (SCREEN WINFDEV) of SCREEN)
       (fetch (SCREEN WINIMAGEOPS) of SCREEN])
```

(\GENERIC.DSPCREATE

```
[LAMBDA (DESTINATION OLDDSP)                              ; Edited 9-Jul-2022 10:47 by rmk
                                                          ; Edited 8-Jul-2022 21:16 by rmk
                                                          ; Edited 27-Dec-93 13:18 by nilsson
```

;; This generic version is installed as DSPCREATE when WINDOW is loaded, overriding the simpler version \SIMPLE.DSPCREATE in ;; LLDISPLAY. We now branch on screens.

;; This adds the undocumented OLDDSP argument, provided for calls from \MEDW.CREATEW to recreate an old window on a new screen.

```
(LET (DSTRM SCREEN)
     [COND
        [(NULL DESTINATION)
         (SETQ DESTINATION ScreenBitMap)
         (SETQ SCREEN (for SC in \SCREENS suchthat (EQ DESTINATION (fetch (SCREEN SCDESTINATION) of SC]
        ((type? SCREEN DESTINATION)
         (SETQ SCREEN DESTINATION))
        (T ;; This is overlaid by BIGBITMAPS
           (\GENERIC.DSPCREATE.DESTINATION.BITMAP? DESTINATION)
           (SETQ SCREEN (for SC in \SCREENS suchthat (EQ DESTINATION (fetch (SCREEN SCDESTINATION) of SC]
     [COND
        (SCREEN (SETQ DSTRM (WINDOWOP 'DSPCREATEFN SCREEN DESTINATION OLDDSP)))
        (T ;; NO SCREEN SPECIFIED, SO THIS IS TO A BITMAP. FILL IT IN:
           (SETQ DSTRM (\COMMON.DSPCREATE DESTINATION]
     DSTRM])
```

(\GENERIC.DSPCREATE.DESTINATION.BITMAP?

```
[LAMBDA (DESTINATION)                                     ; Edited 9-Jul-2022 09:24 by rmk
  (\DTEST DESTINATION 'BITMAP])
```

(\MEDW.GETWINDOWPROP

```
[LAMBDA (SCREEN WINDOW PROP)                              ; Edited 27-Dec-93 11:41 by sybalsky:mv:envos
```

;; gets values from a window. Called by the macro for WINDOWPROP.

```
[OR (type? WINDOW WINDOW)
    (COND
```

```
          ((DISPLAYSTREAMP (\OUTSTREAMARG WINDOW T))
           (SETQ WINDOW (WFROMDS WINDOW)))
          (T (\ILLEGAL.ARG WINDOW]
    (SELECTQ PROP
        (HEIGHT (\GETWINDOWHEIGHT WINDOW))
        (WIDTH                                          ; calculate the width from the REGION in case the user has
                                                        ; changed the clipping region.
               (\GETWINDOWWIDTH WINDOW))
        (RIGHTBUTTONFN
             (fetch (WINDOW RIGHTBUTTONFN) of WINDOW))
        (BUTTONEVENTFN
             (fetch (WINDOW BUTTONEVENTFN) of WINDOW))
        (CURSORINFN (fetch (WINDOW CURSORINFN) of WINDOW))
        (CURSOROUTFN (fetch (WINDOW CURSOROUTFN) of WINDOW))
        (CURSORMOVEDFN
             (fetch (WINDOW CURSORMOVEDFN) of WINDOW))
        (DSP (fetch (WINDOW DSP) of WINDOW))
        (SCREEN (fetch (WINDOW SCREEN) of WINDOW))
        (SCROLLFN (fetch (WINDOW SCROLLFN) of WINDOW))
        (RESHAPEFN (fetch (WINDOW RESHAPEFN) of WINDOW))
        (EXTENT (fetch (WINDOW EXTENT) of WINDOW))
        (REPAINTFN (fetch (WINDOW REPAINTFN) of WINDOW))
        (MOVEFN (fetch (WINDOW MOVEFN) of WINDOW))
        (CLOSEFN (fetch (WINDOW CLOSEFN) of WINDOW))
        (WINDOWENTRYFN
             (fetch (WINDOW WINDOWENTRYFN) of WINDOW))
        (PROCESS (fetch (WINDOW PROCESS) of WINDOW))
        (REGION                                         ; make a copy so we don't have to worry about {or document} the
                                                        ; user clobbering it.
                (fetch (WINDOW REG) of WINDOW))
        (NEWREGIONFN (fetch (WINDOW NEWREGIONFN) of WINDOW))
        (TITLE (fetch (WINDOW WTITLE) of WINDOW))
        (BORDER (fetch (WINDOW WBORDER) of WINDOW))
        (IMAGECOVERED (fetch (WINDOW SAVE) of WINDOW))
        (GETWINDOWUSERPROP WINDOW PROP])
```

(**MEDW.PUTWINDOWPROP**

```
  [LAMBDA (SCREEN WINDOW PROP VALUE)                    ; Edited 27-Dec-93 11:39 by sybalsky:mv:envos
    [OR (type? WINDOW WINDOW)
        (COND
           ((DISPLAYSTREAMP (\OUTSTREAMARG WINDOW))
            (SETQ WINDOW (WFROMDS WINDOW)))
           (T (\ILLEGAL.ARG WINDOW]
    (SELECTQ PROP
        (RIGHTBUTTONFN
             (PROG1 (fetch (WINDOW RIGHTBUTTONFN) of WINDOW)
                    (replace (WINDOW RIGHTBUTTONFN) of WINDOW with VALUE)))
        (BUTTONEVENTFN
             (PROG1 (fetch (WINDOW BUTTONEVENTFN) of WINDOW)
                    (replace (WINDOW BUTTONEVENTFN) of WINDOW with VALUE)))
        (CLOSEFN (PROG1 (fetch (WINDOW CLOSEFN) of WINDOW)
                        (replace (WINDOW CLOSEFN) of WINDOW with VALUE)))
        (MOVEFN (PROG1 (fetch (WINDOW MOVEFN) of WINDOW)
                       (replace (WINDOW MOVEFN) of WINDOW with VALUE)))
        (CURSORINFN (PROG1 (fetch (WINDOW CURSORINFN) of WINDOW)
                           (replace (WINDOW CURSORINFN) of WINDOW with VALUE)))
        (CURSOROUTFN (PROG1 (fetch (WINDOW CURSOROUTFN) of WINDOW)
                            (replace (WINDOW CURSOROUTFN) of WINDOW with VALUE)))
        (CURSORMOVEDFN
             (PROG1 (fetch (WINDOW CURSORMOVEDFN) of WINDOW)
                    (replace (WINDOW CURSORMOVEDFN) of WINDOW with VALUE)))
        (DSP (ERROR "Can't change DSP of a window" WINDOW))
        (SCREEN (ERROR "Can't change SCREEN  of a window" WINDOW))
        (RESHAPEFN (PROG1 (fetch (WINDOW RESHAPEFN) of WINDOW)
                          (replace (WINDOW RESHAPEFN) of WINDOW with VALUE)))
        (REPAINTFN (PROG1 (fetch (WINDOW REPAINTFN) of WINDOW)
                          (replace (WINDOW REPAINTFN) of WINDOW with VALUE)))
        (EXTENT (PROG1 (fetch (WINDOW EXTENT) of WINDOW)
                       (OR (NULL VALUE)
                           (REGIONP VALUE)
                           (\ILLEGAL.ARG VALUE))
                       (replace (WINDOW EXTENT) of WINDOW with VALUE)))
        (SCROLLFN (PROG1 (fetch (WINDOW SCROLLFN) of WINDOW)
                         (replace (WINDOW SCROLLFN) of WINDOW with VALUE)
                         (UPDATE/SCROLL/REG WINDOW)))
        (IMAGECOVERED (ERROR "Not implemented to change IMAGECOVERED property." WINDOW))
        (HEIGHT (ERROR "Not implemented to change HEIGHT as property." WINDOW))
        (WIDTH (ERROR "Not implemented to change WIDTH as property." WINDOW))
        (REGION [PROG (CURREGION)
                      (SETQ CURREGION (WINDOWPROP WINDOW 'REGION))
                      (COND
                         ((NOT (REGIONP VALUE))
                          (\ILLEGAL.ARG VALUE)))
                ;; there is no check for where the new region is nor how big it is;  this is left to MOVEW and RESHAPEW.
                      (COND
```

```
                        ((AND (EQ (fetch (REGION WIDTH) of CURREGION)
                                  (fetch (REGION WIDTH) of VALUE))
                              (EQ (fetch (REGION HEIGHT) of CURREGION)
                                  (fetch (REGION HEIGHT) of VALUE)))
                                                          ; width and height are the same, move the window
                         (MOVEW WINDOW (fetch (REGION LEFT) of VALUE)
                                (fetch (REGION BOTTOM) of VALUE)))
                        (T                                ; dimensions changed, reshape it.
                           (SHAPEW WINDOW VALUE])
      (NEWREGIONFN (PROG1 (fetch (WINDOW NEWREGIONFN) of WINDOW)
                          (replace (WINDOW NEWREGIONFN) of WINDOW with VALUE)))
      (TITLE (PROG1 (fetch (WINDOW WTITLE) of WINDOW)
                    (RESHOWTITLE VALUE WINDOW)))
      (BORDER (PROG1 (fetch (WINDOW WBORDER) of WINDOW)
                     (COND
                        ((NUMBERP VALUE)
                         (RESHOWBORDER VALUE WINDOW))
                        (T (\ILLEGAL.ARG VALUE)))))
      (PROCESS (PROG1 (fetch (WINDOW PROCESS) of WINDOW)
                      (replace (WINDOW PROCESS) of WINDOW with VALUE)))
      (WINDOWENTRYFN
           (PROG1 (fetch (WINDOW WINDOWENTRYFN) of WINDOW)
                  (replace (WINDOW WINDOWENTRYFN) of WINDOW with VALUE)))
      (PROG (OLDDATA OLDVALUE)
            (SETQ OLDDATA (fetch (WINDOW USERDATA) of WINDOW))
            (RETURN (PROG1 (COND
                              (OLDDATA (SETQ OLDVALUE (LISTGET OLDDATA PROP))
                                       [COND
                                          (VALUE (LISTPUT OLDDATA PROP VALUE))
                                          (OLDVALUE             (* Remove the property)
                                                 (COND
                                                    ((EQ (CAR OLDDATA)
                                                         PROP)
                                                     (replace (WINDOW USERDATA) of WINDOW with (CDDR OLDDATA)))
                                                    (T (for TAIL on (CDR OLDDATA) by (CDDR TAIL)
                                                          when (EQ (CADR TAIL)
                                                                   PROP)
                                                          do (FRPLACD TAIL (CDDDR TAIL))
                                                             (RETURN]
                                       OLDVALUE)
                              (VALUE (replace (WINDOW USERDATA) of WINDOW with (LIST PROP VALUE))
                                                          (* know old value is NIL)
                                       NIL))
                           (COND
                              ((AND (fetch (WINDOW WTITLE) of WINDOW)
                                    (EQ PROP 'WINDOWTITLESHADE)) (* change windowtitleshade.)
                               (RESHOWTITLE (fetch (WINDOW WTITLE) of WINDOW)
                                      WINDOW T))))])
```

(**MEDW.CURSOR**

```
  [LAMBDA (SCREEN NEWCURSOR INVERTFLG)                    ; Edited 23-Feb-94 12:16 by sybalsky
```

;; Installs NEWCURSOR as the cursor and returns the old cursor state. If INVERTFLG is non-NIL, the cursor image is inverted during installation.
;; If NEWCURSOR is NIL, just returns the current cursor state.

```
    (DECLARE (GLOBALVARS DEFAULTCURSOR \SOFTCURSORP))
    (PROG (OLDCURSOR)
          (SETQ OLDCURSOR \CURRENTCURSOR)
          (COND
             ((EQ NEWCURSOR T)                            ; If NEWCURSOR is T, use the system default cursor.
              (SETQ NEWCURSOR DEFAULTCURSOR)))
          (COND
             [(\CURSOR-VALID-P NEWCURSOR \SOFTCURSORP)    ; Only install the cursor if it's a real, valid one.
              (\CURSORDOWN)
              (\CURSORUP NEWCURSOR INVERTFLG)              ; set after adjustment to avoid confusion about hotspot during
                                                           ; adjustment.
              (SETQ \CURSORHOTSPOTX (fetch (CURSOR CUHOTSPOTX) of NEWCURSOR))
              (SETQ \CURSORHOTSPOTY (IDIFFERENCE (SUB1 (fetch (BITMAP BITMAPHEIGHT) of (fetch (CURSOR CUIMAGE)
                                                                                          of NEWCURSOR)))
                                                 (fetch (CURSOR CUHOTSPOTY) of NEWCURSOR]
             (NEWCURSOR                                    ; NEWCURSOR = NIL means just return the old one, so only
                                                           ; error if one got specified that wasn't valid.
                      (\ILLEGAL.ARG NEWCURSOR)))
          (RETURN OLDCURSOR])

)

(DEFINEQ
```

(**GENERIC.CURSOR**

```
  [LAMBDA (NEWCURSOR INVERTFLG)                           ; Edited 25-Feb-94 15:07 by sybalsky
```

;; Installs NEWCURSOR as the cursor and returns the old cursor state. If INVERTFLG is non-NIL, the cursor image is inverted during installation.
;; If NEWCURSOR is NIL, just returns the current cursor state.

```
    (COND
       [NEWCURSOR (PROG1 \CURRENTCURSOR
                         (FOR SCREEN IN \SCREENS DO (WINDOWOP 'SETCURSORFN SCREEN NEWCURSOR INVERTFLG)))]
```

```
       (T \CURRENTCURSOR])

)

(DECLARE%: EVAL@COMPILE DONTCOPY
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

(PUTPROPS WINDOWOP DMACRO [ARGS (LET ((OPNAME (CAR ARGS))
                                      (METHOD-DEVICE (CADR ARGS))
                                      (TAIL (CDDR ARGS)))
                                     (COND
                                        [(AND (LISTP OPNAME)
                                              (EQ (CAR OPNAME)
                                                  'QUOTE))
                                         `(SPREADAPPLY* (fetch (SCREEN ,(CADR OPNAME)) of ,METHOD-DEVICE)
                                                 ,METHOD-DEVICE
                                                 ,@TAIL]
                                        (T (ERROR "OPNAME not quoted: " OPNAME])
)
)
```

;; END EXPORTED DEFINITIONS

```
(DECLARE%: DONTEVAL@COMPILE DONTEVAL@LOAD DOCOPY

(MOVD '\GENERIC.DSPCREATE 'DSPCREATE)

(CL:UNLESS (EQUAL (GETD 'CURSOR)
                  (GETD '\GENERIC.CURSOR))
    (MOVD '\GENERIC.CURSOR 'CURSOR))
)

(DECLARE%: EVAL@COMPILE DONTCOPY

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \LastCursorPosition \LastInWindow WindowMenu BackgroundMenu BackgroundMenuCommands
       \LastWindowButtons WWFNS WindowMenuCommands WindowTitleDisplayStream WINDOWTITLEPRINTLEVEL WBorder
       \TOPWDS WINDOWBACKGROUNDSHADE BACKGROUNDFNS)
)
```

;; FOLLOWING DEFINITIONS EXPORTED

```
(DECLARE%: EVAL@COMPILE

(RPAQQ MinWindowWidth 26)

(RPAQQ MinWindowHeight 16)

(CONSTANTS (MinWindowWidth 26)
       (MinWindowHeight 16))
)

(DECLARE%: EVAL@COMPILE

(DATATYPE WINDOW (DSP                                   ; The display stream you use to actually printto the window.
                     NEXTW                              ; Next window in the open-window list
                     SAVE                               ; Saved image from anything this window's on top of
                     REG                                ; Screen region this window occupies
                     BUTTONEVENTFN                      ; FN called when left/middle mouse button goes up/down
                     RIGHTBUTTONFN                      ; FN called when right mouse button goes up/down
                     CURSORINFN                         ; Fn called when mouse enters window
                     CURSOROUTFN                        ; Called when mouse leaves window
                     CURSORMOVEDFN                      ; Called when mouse moves in window
                     REPAINTFN                          ; Redisplay part of thie window
                     RESHAPEFN                          ; Called when window is reshaped
                     EXTENT                             ; Scrolling limits
                     USERDATA                           ; Proplist to hold other window properites
                     VERTSCROLLREG                      ; Region of vert scroll bar
                     HORIZSCROLLREG                     ; Tegion of horiz scroll bar
                     SCROLLFN                           ; Fn to scroll this window
                     VERTSCROLLWINDOW                   ; Vert scroll bar
                     HORIZSCROLLWINDOW                  ; Horiz scroll bar
                     CLOSEFN                            ; Called at close time
                     MOVEFN                             ; Called when window is moved
                     WTITLE                             ; Window's title string, if any
                     NEWREGIONFN                        ; Called to get new window shape
                     WBORDER                            ; Window border-width, in pixels
                     PROCESS                            ; Medley process associated with this window
                     WINDOWENTRYFN                      ; Fn to call when kbd focus is switched here
                     SCREEN                             ; Screen this window appears on
                     (NATIVE-HANDLE FIXP)               ; Uniterpreted place for native window to store a C pointer to its
                                                        ; private info
```

```
                        (NATIVE-INFO1 FIXP)                     ; Reserved in case the pointer must be 64 bits
                        (NATIVE-W1 WORD)                        ; Word for use by native handler
                        (NATIVE-W2 WORD)                        ; Word for use by native handler
                        (NATIVE-P1 POINTER)                     ; Lisp pointer for use by native handler

                        )
        BUTTONEVENTFN _ (FUNCTION TOTOPW)
        WBORDER _ WBorder WINDOWENTRYFN _ (FUNCTION GIVE.TTY.PROCESS)
        (SYSTEM))

(DATATYPE SCREEN (SCONOFF SCDESTINATION SCWIDTH SCHEIGHT SCTOPW SCTOPWDS SCTITLEDS SCFDEV SCDS SCDATA
                        ;; Space for native window manager interface to use.
                        (HANDLE FIXP)                           ; Handle for emulator to store info about display for C code use.
                        (HANDLE2 FIXP)                          ; Reserved in case HANDLE needs to be 64 bits on the C side.
                        (NATIVE-INFO POINTER)                   ; POINTER for the private use of the emulator window code
                        NATIVETYPE                              ; Symbol to tell what kind of native window system we're using.
                        ;; - - - Functional interface to screen management - - -
                        WINIMAGEOPS                             ; IMAGEOPS to be used in display streas on this kind of screen
                        WINFDEV                                 ; FDEV for display streams on this screen
                        CREATEWFN                               ; Create a window
                        OPENWFN                                 ; Open a window
                        CLOSEWFN                                ; Close a window
                        MOVEWFN                                 ; Move a window
                        RELMOVEWFN                              ; Move window, relative
                        SHRINKWFN                               ; Shrink window to icon
                        EXPANDWFN                               ; Expand icon to window
                        SHAPEWFN                                ; Reshape a window
                        REDISPLAYFN                             ; Redisplay (part of) a window
                        GETWINDOWPROPFN                         ; Get window property value
                        PUTWINDOWPROPFN                         ; Set window property value
                        BURYWFN                                 ; Move window behind all others
                        TOTOPWFN                                ; Move iwindow in front of all others
                        IMPORTWFN                               ; Take a native window and save its state internally
                        EXPORTWFN                               ; Take a saved window state and open it on this screen, filling in
                                                                ; screen and methods as needed.
                        DESTROYFN                               ; Destroy this window, for GC finaliszation
                        SETCURSORFN                             ; Set the cursor for this window.
                        PROMPTW                                 ; The prompt window for this screen
                        SHOWGCFN                                ; Show GC indication; called with ON/OFF arg, t=>show gcing
                                                                ; status, NIL=>turn off GC indicator.
                        DSPCREATEFN                             ; Create a displaystream on this screen.
                        BBTTOWIN                                ; BITBLT from a lisp bitmap to a window
                        BBTFROMWIN                              ; BITBLT from a window to a lisp bitmap
                        BBTWINWIN                               ; BITBLT from a window to another window.
                        SCCURSOR                                ; CURSOR that's in effect for this screen by default.
                        SCKEYBOARD                              ; Something about which keyboard we're receiving from.
                        SCDEPTH                                 ; # of bits per pixel on the screen.  THIS WILL REPLACE
                                                                ; SCBITSPERPIXEL ASAP.
                        SCCLOSEDOWN                             ; Close down this screen cleanly, saving window state.
                        SCCLOSESCREEN                           ; Close down thie screen cleanly, no state saving.
                        SCREOPEN                                ; Reopen this screen?
                        SCCARETFLASH                            ; Function to flash thecaret.
                        SCGETSCREENPOSITION                     ; GETSCREENPOSITION
                        SCGETBOXSCREENPOSITION                  ; GETBOXPOSITION
                        SCGETSCREENREGION                       ; GETREGION
                        SCMOVEPOINTER                           ; \CURSORPOSITION

                        )
        SCONOFF _ 'OFF [ACCESSFNS ((SCBITSPERPIXEL (COND
                                                   ((fetch (SCREEN SCDESTINATION) of DATUM)
                                                    (fetch (BITMAP BITMAPBITSPERPIXEL)
                                                       of (fetch (SCREEN SCDESTINATION) of DATUM)))
                                                   (T 1)))
                                   (SCREGION (create REGION
                                                     LEFT _ 0
                                                     BOTTOM _ 0
                                                     WIDTH _ (fetch (SCREEN SCWIDTH) of DATUM)
                                                     HEIGHT _ (fetch (SCREEN SCHEIGHT) of DATUM]
        (SYSTEM))
)

(/DECLAREDATATYPE 'WINDOW
        '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
                POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
                POINTER FIXP FIXP WORD WORD POINTER)
        ;; ---field descriptor list elided by lister---
        '60)

(/DECLAREDATATYPE 'SCREEN
        '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER FIXP FIXP POINTER
                POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
                POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
                POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
                POINTER)
```

```
       ;; ---field descriptor list elided by lister---
       'l00)
)

;; END EXPORTED DEFINITIONS

(DECLARE%: EVAL@COMPILE

;; FOLLOWING DEFINITIONS EXPORTED

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS WINDOWUSERFORMS ENDOFWINDOWUSERFORMS PROMPTWINDOW)
)
)

;; END EXPORTED DEFINITIONS

(ADDTOVAR SYSTEMRECLST
          (DATATYPE WINDOW
                 (DSP NEXTW SAVE REG BUTTONEVENTFN RIGHTBUTTONFN CURSORINFN CURSOROUTFN CURSORMOVEDFN REPAINTFN
                      RESHAPEFN EXTENT USERDATA VERTSCROLLREG HORIZSCROLLREG SCROLLFN VERTSCROLLWINDOW
                      HORIZSCROLLWINDOW CLOSEFN MOVEFN WTITLE NEWREGIONFN WBORDER PROCESS WINDOWENTRYFN SCREEN
                      (NATIVE-HANDLE FIXP)
                      (NATIVE-INFO1 FIXP)
                      (NATIVE-W1 WORD)
                      (NATIVE-W2 WORD)
                      (NATIVE-P1 POINTER)))
          (DATATYPE SCREEN
                 (SCONOFF SCDESTINATION SCWIDTH SCHEIGHT SCTOPW SCTOPWDS SCTITLEDS SCFDEV SCDS SCDATA
                        (HANDLE FIXP)
                        (HANDLE2 FIXP)
                        (NATIVE-INFO POINTER)
                        NATIVETYPE WINIMAGEOPS WINFDEV CREATEWFN OPENWFN CLOSEWFN MOVEWFN RELMOVEWFN SHRINKWFN
                        EXPANDWFN SHAPEWFN REDISPLAYFN GETWINDOWPROPFN PUTWINDOWPROPFN BURYWFN TOTOPWFN
                        IMPORTWFN EXPORTWFN DESTROYFN SETCURSORFN PROMPTW SHOWGCFN DSPCREATEFN BBTTOWIN
                        BBTFROMWIN BBTWINWIN SCCURSOR SCKEYBOARD SCDEPTH SCCLOSEDOWN SCCLOSESCREEN SCREOPEN
                        SCCARETFLASH SCGETSCREENPOSITION SCGETBOXSCREENPOSITION SCGETSCREENREGION SCMOVEPOINTER)
                 ))

(/DECLAREDATATYPE 'WINDOW
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER FIXP FIXP WORD WORD POINTER)
       ;; ---field descriptor list elided by lister---
       '60)

(/DECLAREDATATYPE 'SCREEN
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER FIXP FIXP POINTER
              POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
              POINTER)
       ;; ---field descriptor list elided by lister---
       'l00)

(RPAQ? WindowMenu )

(RPAQ? BackgroundMenu )

(RPAQ? \LastCursorPosition (CREATEPOSITION))

(RPAQ? \LastInWindow )

(RPAQ? \LastWindowButtons 0)

(RPAQ? WINDOWBACKGROUNDSHADE 34850)

(RPAQ? WBorder 4)

(RPAQ? HIGHLIGHTSHADE 32800)

(RPAQ? WINDOWBACKGROUNDBORDER 34850)

(FILESLOAD PAINTW)

(ADDTOVAR WindowMenuCommands
          (Close '\INTERACTIVE.CLOSEW "Closes a window")
          (Snap 'SNAPW "Saves a snapshot of a region of the screen.")
          (Paint 'PAINTW "Starts a painting mode in which the mouse can be
                 used to draw pictures or make notes on windows.")
          (Clear 'CLEARW "Clears a window to its gray.")
          (Bury 'BURYW "Puts a window on the bottom.")
```

```
          (Redisplay 'REDISPLAYW "Redisplays a window using its REPAINTFN.")
          (Hardcopy 'HARDCOPYIMAGEW "Prints a window using its HARDCOPYFN." (SUBITEMS ("To a file"
                                                                                        '
                                                                                   HARDCOPYIMAGEW.TOFILE
                                                                                     "Puts image on a
                                                                                     file; prompts for
                                                                                     filename and
                                                                                     format")
                                                                           ("To a printer"
                                                                                  '
                                                                                HARDCOPYIMAGEW.TOPRINTER
                                                                                  "Sends image to a
                                                                                  printer of your
                                                                                  choosing")))
          (Move 'MOVEW "Moves a window by a corner.")
          (Shape 'SHAPEW "Gets a new region for a window.
                 Left button down marks fixed corner; sweep to other corner.
                 Middle button down moves closest corner.")
          (Shrink 'SHRINKW "Replaces this window with its icon (or title if it doesn't have an icon."))

(ADDTOVAR BackgroundMenuCommands
          (SaveVM '(SAVEVM)
                 "Updates the virtual memory.")
          (Snap '(SNAPW)
                "Saves a snapshot of a region of the screen.")
          (Hardcopy '(HARDCOPYW)
                 "Send hardcopy of screen region to printer."
                 (SUBITEMS ("To a file" '(HARDCOPYREGION.TOFILE)
                                  "Writes a region of screen to a file; prompts for filename and format")
                        ("To a printer" '(HARDCOPYREGION.TOPRINTER)
                               "Sends a region of screen to a printer of your choosing"))))

(ADDTOVAR WINDOWUSERFORMS )

(ADDTOVAR ENDOFWINDOWUSERFORMS )

(DECLARE%: DONTEVAL@LOAD DOCOPY

(COND
   ((NULL \MAINSCREEN)
    (SETQ \MAINSCREEN (CREATESCREEN (SCREENBITMAP)))
    (SETQ \CURSORSCREEN \MAINSCREEN)
    (SETQ LASTSCREEN \MAINSCREEN)
    (WINDOWWORLD 'ON \MAINSCREEN T)))

(MOVD? 'TRUE 'LISPWINDOWP)

(RPAQQ \WINDOWWORLD T)
)

;; Arrange for the proper compiler

(PUTPROPS WINDOW FILETYPE :FAKE-COMPILE-FILE)

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

(ADDTOVAR NLAMA )

(ADDTOVAR NLAML )

(ADDTOVAR LAMA PROMPTPRINT WINDOWPROP DOWINDOWCOM)
)

(PUTPROPS WINDOW COPYRIGHT ("Venue & Xerox Corporation" 1982 1983 1984 1985 1986 1987 1988 1990 1991 1992 1993
                                1994 1999 2000 2021))
```

FUNCTION INDEX

VARIABLE INDEX

MACRO INDEX

RECORD INDEX

PROPERTY INDEX

CONSTANT INDEX

OPTIMIZER INDEX