File created: 28-Jun-99 17:18:50 **{DSK}<project>medley3.5>sources>ATTACHEDWINDOW.;3**

changes to: (FNS RESHAPEALLWINDOWS)

previous date: 28-Jun-99 15:59:05 {DSK}<project>medley3.5>sources>ATTACHEDWINDOW.;2

Read Table: INTERLISP

Package: INTERLISP

Format: XCCS

```
;;
```


```
(RPAQQ ATTACHEDWINDOWCOMS
       ((COMS (* User entries)
              (FNS ATTACHWINDOW ATTACHEDWINDOWS ALLATTACHEDWINDOWS DETACHWINDOW DETACHALLWINDOWS
                   FREEATTACHEDWINDOW MAINWINDOW REMOVEWINDOW REPOSITIONATTACHEDWINDOWS))
        (FNS ATTACHEDWINDOWREGION ATTACHEDWINDOWTOTOPFN CENTERINHEIGHT CENTERINWIDTH CENTRALWINDOW
             CLOSEATTACHEDWINDOWS DOATTACHEDWINDOWCOM DOATTACHEDWINDOWCOM2 DOMAINWINDOWCOMFN
             EXPANDATTACHEDWINDOWS MAKEMAINWINDOW MAXATTACHEDWINDOWEXTENT MAXIMUMMAINWINDOWSIZE
             MAXIMUMWINDOWSIZE MINATTACHEDWINDOWEXTENT MINIMUMMAINWINDOWSIZE MOVEATTACHEDWINDOWS
             MOVEATTACHEDWINDOWTOPLACE OPENATTACHEDWINDOWS RESHAPEALLWINDOWS \TOTALPROPOSEDSIZE
             SHRINKATTACHEDWINDOWS TOPATTACHEDWINDOWS UNMAKEMAINWINDOW UPIQUOTIENT WINDOWPOSITION WINDOWSIZE
             \ALLOCMINIMUMSIZES \ALLOCSPACETOGROUPEDWINDOWS \TOTALFIXEDHEIGHT \TOTALFIXEDWIDTH
             \ALLOCHEIGHTTOGROUPEDWINDOW \ALLOCWIDTHTOGROUPEDWINDOW \ATWGROUPSIZE \BREAKAPARTATWSTRUCTURE
             \BUILDATWSTRUCTURE \LIMITBYMAX \LIMITBYMIN \MAXHEIGHTOFGROUP \MAXWIDTHOFGROUP
             \RESHAPEATTACHEDWINDOWSAROUNDMAINW \SETGROUPMIN \SETWINFOXSIZE \SETWINFOYSIZE \SHAREOFXTRAX
             \SHAREOFXTRAY)
        (FNS ATTACHMENU CREATEMENUEDWINDOW MENUWINDOW MENUWMINSIZEFN MENUWRESHAPEFN)
        (FNS GETPROMPTWINDOW \PROMPTWINDOW.EXPAND \PROMPTWINDOW.SET.HEIGHT \PROMPTWINDOW.OPENFN
             \PROMPTWINDOW.PAGEFULLFN REATTACHPROMPTWINDOW REMOVEPROMPTWINDOW)
        (DECLARE%: DONTCOPY DOEVAL@COMPILE (RECORDS RESHAPINGWINDOWDATA)
               (GLOBALVARS WindowMenu WindowTitleDisplayStream WBorder WindowMenuCommands))
        (VARIABLES *ATTACHED-WINDOW-COMMAND-SYNONYMS*)))
```

(* * User entries)

```
(DEFINEQ
```

(ATTACHWINDOW

```
  [LAMBDA (WINDOWTOATTACH MAINWINDOW EDGE POSITIONONEDGE WINDOWCOMACTION)
                                                             ; Edited 12-Jan-87 18:12 by woz
```

;; attaches a window to another window. EDGE is one of LEFT, RIGHT, TOP or BOTTOM. POSITIONONEDGE is one of NIL {means reshape ;; window to fit new main window size}, {left or bottom}, {center} or {right or top}. The attached window is opened if the main window is open, and ;; not if not.

```
    (PROG (MAINW ATTACHW)
          (SETQ MAINW (INSURE.WINDOW MAINWINDOW))
          (SETQ ATTACHW (INSURE.WINDOW WINDOWTOATTACH))
          (COND
             ((OR (EQ WINDOWTOATTACH MAINWINDOW)
                  (MEMB MAINW (ALLATTACHEDWINDOWS ATTACHW)))
              (ERROR "Attempt to create a loop in window attachment" ATTACHW)
              (RETURN)))
          (SELECTQ EDGE
               ((LEFT RIGHT TOP BOTTOM))
               (NIL (SETQ EDGE 'TOP))
               (\ILLEGAL.ARG EDGE))
          (SELECTQ POSITIONONEDGE
               ((JUSTIFY CENTER LEFT RIGHT TOP BOTTOM))
               (NIL (SETQ POSITIONONEDGE 'JUSTIFY))
               (\ILLEGAL.ARG POSITIONONEDGE))
          (MAKEMAINWINDOW MAINW)
          (WINDOWADDPROP MAINW 'ATTACHEDWINDOWS ATTACHW)
          (WINDOWPROP ATTACHW 'WHEREATTACHED (CONS EDGE POSITIONONEDGE))
          (WINDOWPROP ATTACHW 'MAINWINDOW MAINW)
          (WINDOWPROP ATTACHW 'TOTOPFN (FUNCTION ATTACHEDWINDOWTOTOPFN))
```

;; put a property on the window that will be noticed by DOWINDOWCOM to decide what to do with window command requests.

```
          (WINDOWPROP ATTACHW 'DOWINDOWCOMFN (FUNCTION DOATTACHEDWINDOWCOM))
          [SELECTQ WINDOWCOMACTION
               (MAIN (WINDOWPROP ATTACHW 'PASSTOMAINCOMS T))
               (HERE                                         ; leave it alone
                     (WINDOWPROP ATTACHW 'PASSTOMAINCOMS NIL))
               (LOCALCLOSE                                   ; set up so that closing is handled locally and detaches the
                                                             ; window.
                           (WINDOWADDPROP ATTACHW 'CLOSEFN (FUNCTION DETACHWINDOW))
                           (WINDOWPROP ATTACHW 'PASSTOMAINCOMS '(MOVEW SHAPEW SHRINKW BURYW)))
               (WINDOWPROP ATTACHW 'PASSTOMAINCOMS '(CLOSEW MOVEW SHAPEW SHRINKW BURYW]
          (MOVEATTACHEDWINDOWTOPLACE ATTACHW MAINW EDGE POSITIONONEDGE)
          (AND (OPENWP MAINW)
               (OPENW ATTACHW))
          (RETURN MAINW])
```

(ATTACHEDWINDOWS

```
[LAMBDA (WINDOW COM)                                    ; Edited  5-Jul-88 19:01 by drc:
```

;; Returns the list of windows attached to this window. COM can be a window command, only the attached windows who allow this COM to be ;; applied to them from above will be returned. An attached window can have a ALLOWMAINCOMS prop, which is a list of allowable commands. If ;; ALLOWMAINCOMS is NIL, all commands are allowed. If COM is not given, all attached windows are returned.

```
(DECLARE (GLOBALVARS *ATTACHED-WINDOW-COMMAND-SYNONYMS*))
(LET [(AWS (WINDOWPROP WINDOW 'ATTACHEDWINDOWS]
     (COND
        ((NULL COM)
         AWS)
        (T (LET ((REALCOM (OR (CDR (ASSOC COM *ATTACHED-WINDOW-COMMAND-SYNONYMS*))
                              COM)))
                (COND
                   ([for ATTW in AWS thereis (FMEMB COM (WINDOWPROP ATTW 'REJECTMAINCOMS]
                    ;; don't cons new list of windows unless we must
                    (for ATTW in AWS unless (FMEMB COM (WINDOWPROP ATTW 'REJECTMAINCOMS)) collect ATTW))
                   (T AWS])
```

(ALLATTACHEDWINDOWS

```
[LAMBDA (MAINW)                                         (* rrb "30-NOV-83 16:29")
```

(* returns a list of all of the windows attached to MAINW or any of its attached windows.)

```
(PROG ((ATWS (ATTACHEDWINDOWS MAINW)))
      (RETURN (COND
                 (ATWS (APPEND ATWS (for ATW in ATWS join (ALLATTACHEDWINDOWS ATW])
```

(DETACHWINDOW

```
[LAMBDA (WINDOWTODETACH MAINWINDOW)                     ; Edited  5-Jul-88 19:43 by drc:
```

;;; detaches a window from its main window.

```
(PROG ((WHEREAT (WINDOWPROP WINDOWTODETACH 'WHEREATTACHED NIL))
       (MAINW (OR MAINWINDOW (WINDOWPROP WINDOWTODETACH 'MAINWINDOW NIL)))
       ATWINS PWINDOW OLDFN)
      (OR MAINW (RETURN NIL))
      (WINDOWDELPROP MAINW 'ATTACHEDWINDOWS WINDOWTODETACH)
      (COND
         ((NOT (ATTACHEDWINDOWS MAINW))
          (UNMAKEMAINWINDOW MAINW)))
      (SELECTQ (SETQ OLDFN (WINDOWPROP WINDOWTODETACH 'DOWINDOWCOMFN NIL))
           ((DOMAINWINDOWCOMFN DOATTACHEDWINDOWCOM))
           (WINDOWPROP WINDOWTODETACH 'DOWINDOWCOMFN OLDFN)) ; Remove window's TOTOPFN and DOWINDOWCOMFN if they
                                                             ; were the ones that ATTACHWINDOW put there
      (SELECTQ (SETQ OLDFN (WINDOWPROP WINDOWTODETACH 'TOTOPFN NIL))
           (ATTACHEDWINDOWTOTOPFN)
           (WINDOWPROP WINDOWTODETACH 'TOTOPFN OLDFN))
      (RETURN WHEREAT])
```

(DETACHALLWINDOWS

```
[LAMBDA (MAINWINDOW)                                    ; Edited  5-Jul-88 19:45 by drc:
  (REMOVEPROMPTWINDOW MAINWINDOW)
```

;; Do this separately so that prompt window is 'permanently' removed, not just locally closed

```
  (for W in (WINDOWPROP MAINWINDOW 'ATTACHEDWINDOWS) do (DETACHWINDOW W MAINWINDOW)
                                                        (CLOSEW W])
```

(FREEATTACHEDWINDOW

```
[LAMBDA (WINDOW)                                        (* jow "16-Aug-85 14:35")
```

(* frees an attached window and snuggles any other attached windows closer to the main window.
Only the windows that allowed MOVEW will be snuggled.)

```
(LET* [(MAINWINDOW (MAINWINDOW WINDOW))
       [ATWINS (COPY (ATTACHEDWINDOWS MAINWINDOW 'MOVEW]
       (REGION (WINDOWPROP WINDOW 'REGION))
       (BOTTOM (fetch (REGION BOTTOM) of REGION))
       (HEIGHT (fetch (REGION HEIGHT) of REGION))
       (EDGE (CAR (WINDOWPROP WINDOW 'WHEREATTACHED]
      (DETACHWINDOW WINDOW)
      (SELECTQ EDGE
           (TOP [for ATWIN in ATWINS when (IGREATERP (fetch (REGION BOTTOM) of (WINDOWPROP ATWIN 'REGION))
                                                     BOTTOM)
                   do (RELMOVEW ATWIN (create POSITION
                                              XCOORD _ 0
                                              YCOORD _ (IMINUS HEIGHT])
           (BOTTOM (for ATWIN in ATWINS when (ILESSP (fetch (REGION BOTTOM) of (WINDOWPROP ATWIN 'REGION))
                                                     BOTTOM)
                      do (RELMOVEW ATWIN (create POSITION
```

```
                                                    XCOORD _ 0
                                                    YCOORD _ HEIGHT))))
            NIL])
```

(MAINWINDOW

```
  [LAMBDA (WINDOW RECURSEFLG)                                   (* rrb "20-Aug-84 09:45")
```

(* * returns the main window of a window. If recurseflg is T, continues until it finds a window not attached to any other.)

```
    (PROG ((WIN (\INSUREWINDOW WINDOW))
           MAINW)
          (COND
             ([NULL (SETQ MAINW (WINDOWPROP WIN 'MAINWINDOW]
              (RETURN WIN))
             ((NULL RECURSEFLG)
              (RETURN MAINW)))
      LP  (COND
             ([NULL (SETQ WIN (WINDOWPROP MAINW 'MAINWINDOW]
              (RETURN MAINW))
             (T (SETQ MAINW WIN)
                (GO LP])
```

(REMOVEWINDOW

```
  [LAMBDA (WINDOW)                                              (* jow "16-Aug-85 14:37")
                                                                (* Closes an attached window and then calls
                                                                FREEATTACHEDWINDOW to snuggle up other windows)
    (CLOSEW WINDOW)
    (FREEATTACHEDWINDOW WINDOW])
```

(REPOSITIONATTACHEDWINDOWS

```
  [LAMBDA (WINDOW)                                              ; Edited  6-Jan-87 14:38 by woz
```

(* can be a main window's RESHAPEFN. used when some attached windows don't want to be reshaped, but do want to be repositioned after a reshape.)

```
    (for ATTW in (ATTACHEDWINDOWS WINDOW 'MOVEW) do (MOVEATTACHEDWINDOWTOPLACE ATTW WINDOW)
                                                    (OR (OPENWP ATTW)
                                                        (\OPENW1 ATTW])

)

(DEFINEQ
```

(ATTACHEDWINDOWREGION

```
  [LAMBDA (MAINW COM)                                           (* jow "15-Aug-85 13:08")
```

(* returns the region of the area taken up by a window and all of its attached windows.
COM can be the command that this region is being calculated for, and is passed to ATTACHEDWINDOWS so windows can except themselves.)

```
    (PROG [(REG (WINDOWPROP MAINW 'REGION]
          [for ATWIN in (ATTACHEDWINDOWS MAINW COM) do (SETQ REG (UNIONREGIONS REG (WINDOWREGION ATWIN]
          (RETURN REG])
```

(ATTACHEDWINDOWTOTOPFN

```
  [LAMBDA (WINDOW)                                              ; Edited 17-Aug-88 19:46 by jds
```

;; This function causes both the main window and its attached windows to be visible when either is selected

```
    (LET ((ROOT (MAINWINDOW WINDOW T)))
```

;; start at the root & let it propagate down

```
         (COND
            ((AND (WINDOWP ROOT)
                  (NEQ ROOT WINDOW))
             (TOTOPW ROOT])
```

(CENTERINHEIGHT

```
  [LAMBDA (HEIGHTTOCENTER RELATIVETOREGION)                     ; Edited 13-Jan-87 13:52 by woz
```

(* returns the bottom coordinate that a height needs to be centered relative to a region.)

```
    (PLUS (fetch (REGION BOTTOM) of RELATIVETOREGION)
          (IQUOTIENT (DIFFERENCE (fetch (REGION HEIGHT) of RELATIVETOREGION)
                                 HEIGHTTOCENTER)
                2])
```

(CENTERINWIDTH

```
  [LAMBDA (WIDTHTOCENTER RELATIVETOREGION)                      (* rrb "15-NOV-83 13:21")
```

(* returns the left coordinate that a width needs to be centered relative to a region.)

```
    (PLUS (fetch (REGION LEFT) of RELATIVETOREGION)
          (IQUOTIENT (DIFFERENCE (fetch (REGION WIDTH) of RELATIVETOREGION)
                                 WIDTHTOCENTER)
                2]))
```

(CENTRALWINDOW

```
  [LAMBDA (WINDOW)                                                    (* rrb "30-Dec-83 13:59")

          (* returns the window that is a main window to this one and is not itself attached to any other.)

    (PROG (MAINW)
      LP  (COND
             ((SETQ MAINW (WINDOWPROP WINDOW 'MAINWINDOW))
              (SETQ WINDOW MAINW)
              (GO LP)))
          (RETURN WINDOW])
```

(CLOSEATTACHEDWINDOWS

```
  [LAMBDA (WINDOW)                                                    (* jow "15-Aug-85 13:02")
                                                                      (* propagates closing to attached windows.)
    (for ATTACHEDWINDOW in (ATTACHEDWINDOWS WINDOW 'CLOSEW) do (CLOSEW ATTACHEDWINDOW)
                                                               (WINDOWPROP ATTACHEDWINDOW 'MAINWINDOW NIL])
```

(DOATTACHEDWINDOWCOM

```
  [LAMBDA (ATTACHEDW)                                                 ; Edited 16-Jul-92 11:22 by cat
                                                                      ; Edited 22-Jan-88 13:35 by woz

    ;; a right button function for attached windows that brings up the window command menu and then, depending upon the command selected, either
    ;; passes the command to the main window or performs it on the attached window.  The commands in the windowprop PASSTOMAINCOMS are
    ;; passed to the central window.  Others are applied to ATTACHEDW.

    (COND
       ((WINDOWP ATTACHEDW)
        (TOTOPW ATTACHEDW)
        (LET [(COM (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]
             (CL:WHEN COM
                 (COND
                    ([OR (EQ (WINDOWPROP ATTACHEDW 'PASSTOMAINCOMS)
                             T)
                         (MEMB (OR (CDR (ASSOC COM *ATTACHED-WINDOW-COMMAND-SYNONYMS*))
                                   COM)
                               (WINDOWPROP ATTACHEDW 'PASSTOMAINCOMS]
                     (APPLY* COM (CENTRALWINDOW ATTACHEDW)))
                    (T (APPLY* COM ATTACHEDW)))
                 T)))
       ((NULL ATTACHEDW)
        (DOBACKGROUNDCOM])
```

(DOATTACHEDWINDOWCOM2

```
  [LAMBDA (ATTACHEDW)                                                 (* rrb "28-Mar-84 11:25")
                                                                      (* a right button function for attached windows that want to
                                                                      handle CLOSE locally.)
    (DOATTACHEDWINDOWCOM ATTACHEDW T])
```

(DOMAINWINDOWCOMFN

```
  [LAMBDA (ATTACHEDW)                                                 (* rrb "10-Dec-83 14:57")
                                                                      (* applies the right button function of the main window.)

    (PROG (MAINW)
          (RETURN (APPLY* (OR (WINDOWPROP (SETQ MAINW (WINDOWPROP ATTACHEDW 'MAINWINDOW))
                                          'RIGHTBUTTONFN)
                              (FUNCTION DOWINDOWCOM))
                          MAINW])
```

(EXPANDATTACHEDWINDOWS

```
  [LAMBDA (WINDOW)                                                    ; Edited  5-Mar-87 11:03 by lal
                                                                      ; propagates expanding to attached windows.
                                                                      ; doesn't allow the attached window functions to stop the
                                                                      ; expanding.

    (if (WINDOWPROP WINDOW 'EXPANDREGIONFN)
        then (REPOSITIONATTACHEDWINDOWS WINDOW)
      else (for ATTACHEDWINDOW in (ATTACHEDWINDOWS WINDOW 'EXPANDW)
```

```
            do (OR (OPENWP ATTACHEDWINDOW)
                   (DOUSERFNS (WINDOWPROP ATTACHEDWINDOW 'EXPANDFN)
                          ATTACHEDWINDOW))                    ; the expandfn may have opened the window.
               (OR (OPENWP ATTACHEDWINDOW)
                   (\OPENW1 ATTACHEDWINDOW])
```

(MAKEMAINWINDOW

```
[LAMBDA (MAINWINDOW)                                          (* jow "15-Aug-85 13:23")

        (* puts the necessary functions on a window to propagate its activities to all of its attached windows.)
                                                              (* has functions for moving, reshaping, totoping)
  (WINDOWADDPROP MAINWINDOW 'TOTOPFN (FUNCTION TOPATTACHEDWINDOWS))
  (WINDOWADDPROP MAINWINDOW 'CLOSEFN (FUNCTION CLOSEATTACHEDWINDOWS))
  (WINDOWADDPROP MAINWINDOW 'OPENFN (FUNCTION OPENATTACHEDWINDOWS))
  (WINDOWADDPROP MAINWINDOW 'SHRINKFN (FUNCTION SHRINKATTACHEDWINDOWS))
  (WINDOWADDPROP MAINWINDOW 'EXPANDFN (FUNCTION EXPANDATTACHEDWINDOWS))
  (WINDOWPROP MAINWINDOW 'CALCULATEREGIONFN (FUNCTION ATTACHEDWINDOWREGION))
  [PROG [(OLDMINSIZE (WINDOWPROP MAINWINDOW 'MINSIZE))
         (OLDMAXSIZE (WINDOWPROP MAINWINDOW 'MAXSIZE]         (* move this windows minsize function and maxsize onto a
                                                              different place.)
        (COND
           ((AND OLDMINSIZE (NEQ OLDMINSIZE (FUNCTION MINATTACHEDWINDOWEXTENT)))
            (WINDOWPROP MAINWINDOW 'MAINWINDOWMINSIZE OLDMINSIZE)))
        (COND
           ((AND OLDMAXSIZE (NEQ OLDMAXSIZE (FUNCTION MAXATTACHEDWINDOWEXTENT)))
            (WINDOWPROP MAINWINDOW 'MAINWINDOWMAXSIZE OLDMAXSIZE]
  (WINDOWPROP MAINWINDOW 'MINSIZE (FUNCTION MINATTACHEDWINDOWEXTENT))
  (WINDOWPROP MAINWINDOW 'MAXSIZE (FUNCTION MAXATTACHEDWINDOWEXTENT))
  (WINDOWADDPROP MAINWINDOW 'MOVEFN (FUNCTION MOVEATTACHEDWINDOWS))
  (WINDOWPROP MAINWINDOW 'DOSHAPEFN (FUNCTION RESHAPEALLWINDOWS])
```

(MAXATTACHEDWINDOWEXTENT

```
[LAMBDA (MAINW)                                               (* bvm%: "29-Dec-83 15:57")

        (* returns the maximum extent of a window computing it from the attached windows if necessary.)

  (PROG ((ATWS (ATTACHEDWINDOWS MAINW))
         (EXTENT (MAXIMUMMAINWINDOWSIZE MAINW))
         TL TC TR RT RC RB BR BC BL LB LC LT)
        [COND
           ((NULL ATWS)
            (RETURN EXTENT))
           ((NULL EXTENT)                                     (* if the main window is willing to expand, start with a large
                                                              maximum)
            (RETURN (SETQ EXTENT (CONS 64000 64000]
        [SETQ TL (SETQ TC (SETQ TR (CDR EXTENT]
        [SETQ RT (SETQ RC (SETQ RB (CAR EXTENT]
        (SETQ BR (SETQ BC (SETQ BL 0)))
        (SETQ LB (SETQ LC (SETQ LT 0)))
        (bind ATWHERE WHERECODE ATWDTH ATWHGHT for ATW in ATWS
           do                                                 (* go through the attached windows keeping track of their effect
                                                              on the extent.)
              (SETQ EXTENT (MAXIMUMWINDOWSIZE ATW))
              (SETQ ATWDTH (OR (CAR EXTENT)
                               64000))
              (SETQ ATWHGHT (OR (CDR EXTENT)
                                64000))
              (SETQ WHERECODE (SELECTQ [CDR (SETQ ATWHERE (WINDOWPROP ATW 'WHEREATTACHED]
                                       (JUSTIFY 'JUSTIFY)
                                       (CENTER 0)
                                       ((LEFT BOTTOM)
                                           -1)
                                       1))
              (SELECTQ (CAR ATWHERE)
                   (TOP [COND
                           ((GREATERP ATWDTH (DIFFERENCE RT LT))

        (* check to see if min width pushes the width. This could push either way and is actually not right because a later window on
        the left or right top could use this extra.)

                               (SETQ RT (PLUS ATWDTH LT]
                            (SELECTQ WHERECODE
                                 (JUSTIFY [SETQ TL (SETQ TC (SETQ TR (PLUS (MAX TL TC TR)
                                                                           ATWHGHT])
                                 (-1 (SETQ TL (PLUS TL ATWHGHT)))
                                 (0 (SETQ TC (PLUS TC ATWHGHT)))
                                 (1 (SETQ TR (PLUS TR ATWHGHT)))
                                 (SHOULDNT)))
                   (RIGHT [COND
                              ((GREATERP ATWHGHT (DIFFERENCE TR BR))
                               (SETQ TR (PLUS ATWHGHT BR]
                          (SELECTQ WHERECODE
                               (JUSTIFY [SETQ RT (SETQ RC (SETQ RB (PLUS (MAX RT RC RB)
                                                                         ATWDTH])
                               (1 (SETQ RT (PLUS RT ATWDTH)))
```

```
                              (0 (SETQ RC (PLUS RC ATWDTH)))
                              (-1 (SETQ RB (PLUS RB ATWDTH)))
                              (SHOULDNT)))
                    (LEFT [COND
                              ((GREATERP ATWHGHT (DIFFERENCE TL BL))
                               (SETQ TL (PLUS ATWHGHT BL]
                          (SELECTQ WHERECODE
                              (JUSTIFY [SETQ LT (SETQ LC (SETQ LB (DIFFERENCE (MIN LT LC LB)
                                                                              ATWDTH])
                              (1 (SETQ LT (DIFFERENCE LT ATWDTH)))
                              (0 (SETQ LC (DIFFERENCE LC ATWDTH)))
                              (-1 (SETQ LB (DIFFERENCE LB ATWDTH)))
                              (SHOULDNT)))
                    (BOTTOM [COND
                               ((GREATERP ATWDTH (DIFFERENCE RB LB))
                                (SETQ RB (PLUS ATWDTH LB]
                            (SELECTQ WHERECODE
                                (JUSTIFY [SETQ BL (SETQ BC (SETQ BR (DIFFERENCE (MIN BL BC BR)
                                                                                ATWHGHT])
                                (-1 (SETQ BL (DIFFERENCE BL ATWHGHT)))
                                (0 (SETQ BC (DIFFERENCE BC ATWHGHT)))
                                (1 (SETQ BR (DIFFERENCE BR ATWHGHT)))
                                (SHOULDNT)))
                    (SHOULDNT)))
          (RETURN (CONS (DIFFERENCE (MAX RT RC RB)
                                    (MIN LT LC LB))
                        (DIFFERENCE (MAX TL TC TR)
                                    (MIN BL BC BR])
```

(MAXIMUMMAINWINDOWSIZE

```
  [LAMBDA (WINDOW)                                        (* bvm%: "29-Dec-83 15:46")
                                                          (* returns the maximum extent of a main window)
    (PROG [(EXT (WINDOWPROP WINDOW 'MAINWINDOWMAXSIZE]
          [COND
             ((NULL EXT)
              (RETURN NIL))
             ((LITATOM EXT)
              (SETQ EXT (APPLY* EXT WINDOW]
          [COND
             [(AND (NUMBERP (CAR EXT))
                   (NUMBERP (CDR EXT]
             (T (SETQ EXT (ERROR "Illegal maximum size property" EXT]
          (RETURN EXT])
```

(MAXIMUMWINDOWSIZE

```
  [LAMBDA (WINDOW)                                        (* rrb "19-Mar-84 14:23")
                                                          (* returns the maximum extent of a window)
    (PROG [(EXT (WINDOWPROP WINDOW 'MAXSIZE]
          [COND
             ((NULL EXT)
              (RETURN NIL))
             ((LITATOM EXT)
              (SETQ EXT (APPLY* EXT WINDOW]
          [COND
             [(AND (OR (NULL (CAR EXT))
                       (NUMBERP (CAR EXT)))
                   (OR (NULL (CDR EXT))
                       (NUMBERP (CDR EXT]
             (EXT (SETQ EXT (ERROR "Illegal extent property" EXT]
          (RETURN EXT])
```

(MINATTACHEDWINDOWEXTENT

```
  [LAMBDA (MAINW)                                         (* rrb "15-Dec-83 10:16")
                                                          (* returns the extent of a window computing it from the attached
    windows if necessary.)
    (PROG ((ATWS (ATTACHEDWINDOWS MAINW))
           (EXTENT (MINIMUMMAINWINDOWSIZE MAINW))
           TL TC TR RT RC RB BR BC BL LB LC LT)
          (COND
             ((NULL ATWS)
              (RETURN EXTENT)))
          [SETQ TL (SETQ TC (SETQ TR (CDR EXTENT]
          [SETQ RT (SETQ RC (SETQ RB (CAR EXTENT]
          (SETQ BR (SETQ BC (SETQ BL 0)))
          (SETQ LB (SETQ LC (SETQ LT 0)))
          (bind ATWHERE WHERECODE ATWDTH ATWHGHT for ATW in ATWS
             do                                           (* go through the attached windows keeping track of their effect
                on the extent.)
                (SETQ EXTENT (MINIMUMWINDOWSIZE ATW))
                (SETQ ATWDTH (CAR EXTENT))
                (SETQ ATWHGHT (CDR EXTENT))
                (SETQ WHERECODE (SELECTQ [CDR (SETQ ATWHERE (WINDOWPROP ATW 'WHEREATTACHED]
                                      (JUSTIFY 'JUSTIFY)
                                      (CENTER 0)
```

```
                                ((LEFT BOTTOM)
                                   -1)
                                1))
                (SELECTQ (CAR ATWHERE)
                     (TOP [COND
                        ((GREATERP ATWDTH (DIFFERENCE RT LT))
```

(* check to see if min width pushes the width. This could push either way and is actually not right because a later window on the left or right top could use this extra.)

```
                            (SETQ RT (PLUS ATWDTH LT]
                       (SELECTQ WHERECODE
                            (JUSTIFY [SETQ TL (SETQ TC (SETQ TR (PLUS (MAX TL TC TR)
                                                                      ATWHGHT])
                            (-1 (SETQ TL (PLUS TL ATWHGHT)))
                            (0 (SETQ TC (PLUS TC ATWHGHT)))
                            (1 (SETQ TR (PLUS TR ATWHGHT)))
                            (SHOULDNT)))
                     (RIGHT [COND
                              ((GREATERP ATWHGHT (DIFFERENCE TR BR))
                               (SETQ TR (PLUS ATWHGHT BR]
                            (SELECTQ WHERECODE
                                 (JUSTIFY [SETQ RT (SETQ RC (SETQ RB (PLUS (MAX RT RC RB)
                                                                           ATWDTH])
                                 (1 (SETQ RT (PLUS RT ATWDTH)))
                                 (0 (SETQ RC (PLUS RC ATWDTH)))
                                 (-1 (SETQ RB (PLUS RB ATWDTH)))
                                 (SHOULDNT)))
                     (LEFT [COND
                              ((GREATERP ATWHGHT (DIFFERENCE TL BL))
                               (SETQ TL (PLUS ATWHGHT BL]
                            (SELECTQ WHERECODE
                                 (JUSTIFY [SETQ LT (SETQ LC (SETQ LB (DIFFERENCE (MIN LT LC LB)
                                                                           ATWDTH])
                                 (1 (SETQ LT (DIFFERENCE LT ATWDTH)))
                                 (0 (SETQ LC (DIFFERENCE LC ATWDTH)))
                                 (-1 (SETQ LB (DIFFERENCE LB ATWDTH)))
                                 (SHOULDNT)))
                     (BOTTOM [COND
                                 ((GREATERP ATWDTH (DIFFERENCE RB LB))
                                  (SETQ RB (PLUS ATWDTH LB]
                               (SELECTQ WHERECODE
                                    (JUSTIFY [SETQ BL (SETQ BC (SETQ BR (DIFFERENCE (MIN BL BC BR)
                                                                              ATWHGHT])
                                    (-1 (SETQ BL (DIFFERENCE BL ATWHGHT)))
                                    (0 (SETQ BC (DIFFERENCE BC ATWHGHT)))
                                    (1 (SETQ BR (DIFFERENCE BR ATWHGHT)))
                                    (SHOULDNT)))
                     (SHOULDNT)))
            (RETURN (CONS (DIFFERENCE (MAX RT RC RB)
                              (MIN LT LC LB))
                          (DIFFERENCE (MAX TL TC TR)
                              (MIN BL BC BR])
```

(**MINIMUMMAINWINDOWSIZE**

```
[LAMBDA (WINDOW)                                        (* rrb "24-Sep-86 14:03")
                                                        (* returns the minimum extent of a window)
  (PROG [(EXT (WINDOWPROP WINDOW 'MAINWINDOWMINSIZE]
        [COND
           [(NULL EXT)
            (SETQ EXT (CONS 26 (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])
```

(**MOVEATTACHEDWINDOWS**

```
[LAMBDA (WINDOW NEWPOS)                                 ; Edited  8-Jul-88 11:00 by drc:
                                                        ; propagates moving to attached windows.
  (PROG [(DELTA (PTDIFFERENCE NEWPOS (WINDOWPOSITION WINDOW]
        (for ATTACHEDWINDOW in (ATTACHEDWINDOWS WINDOW 'MOVEW) do
                                                        ;; bring each to top by hand so we don't bring whole tree
                                                        ;; to top for each one we move
            (AND (OPENWP ATTACHEDWINDOW)
                 (TOTOPW ATTACHEDWINDOW T))
            (MOVEW ATTACHEDWINDOW (PTPLUS (
                                                                  WINDOWPOSITION
                                                                    ATTACHEDWINDOW
                                                                      )
                                                                    DELTA))
```

```
                                        ;; main window (non-terminal) about to be moved.  bring it to top by hand so that
                                        ;; whole tree doesn't get brought to top.

                                                (AND (OPENWP WINDOW)
                                                     (TOTOPW WINDOW T])
```

(**MOVEATTACHEDWINDOWTOPLACE**

```
  [LAMBDA (ATWIN MAINW EDGE POSONEDGE)                  ; Edited 12-Jan-87 17:01 by woz
                                                        (* DECLARATIONS%: (RECORD ATTACHEDWINDATA
                                                        ((EDGE . WHEREONEDGE) WID . HGHT)))
```

;;; moves a window to the place it should be relative to MAINW and reshapes it if it is JUSTIFY. The window will be opened if it is justified, and ;;; otherwise will not. This function should not open the window; it is a nasty side effect of reshaping the window. So if the main window is not open, ;;; punt, and let the openfn take care of calling me again, because the attached window shouldn't be opened. If the main window is open, the attached ;;; window will be moved into position, and it is the responsibility of the caller to ensure that the window gets opened.

```
    (AND (OPENWP MAINW)
         (PROG (MAINWEXTENT EXTENT ATMINWIDTH ATMINHEIGHT ATWHGHT ATWDTH TL TC TR RT RC RB BR BC BL LB LC LT)
               [COND
                  ((NULL EDGE)
                   (SETQ EDGE (WINDOWPROP ATWIN 'WHEREATTACHED))
                   (SETQ POSONEDGE (CDR EDGE))
                   (SETQ EDGE (CAR EDGE]                        ; calculate the minimum so that this window won't be reshaped
                                                                ; smaller than its minimum.
               [SETQ ATMINHEIGHT (CDR (SETQ ATMINWIDTH (MINIMUMWINDOWSIZE ATWIN]
               (SETQ ATMINWIDTH (CAR ATMINWIDTH))
               (SETQ POSONEDGE (SELECTQ POSONEDGE
                                    (JUSTIFY 'JUSTIFY)
                                    (CENTER 0)
                                    ((LEFT BOTTOM)
                                        -1)
                                    1))
               (SETQ MAINWEXTENT (WINDOWPROP MAINW 'REGION))
```

;; the extent of a group of windows is thought of as its maximum extent along each edge and each position on that edge eg. top-left, ;; top-center, top-right. A justify takes the maximum of the three positions along that edge.

```
               [SETQ TL (SETQ TC (SETQ TR (fetch (REGION TOP) of MAINWEXTENT]
               [SETQ RT (SETQ RC (SETQ RB (fetch (REGION RIGHT) of MAINWEXTENT]
               [SETQ BR (SETQ BC (SETQ BL (fetch (REGION BOTTOM) of MAINWEXTENT]
               [SETQ LB (SETQ LC (SETQ LT (fetch (REGION LEFT) of MAINWEXTENT]
               (bind ATWHERE ATPOSONEDGE ATWREG for ATW in (ATTACHEDWINDOWS MAINW)
                  until (EQ ATW ATWIN)
                  do
```

;; go through the attached windows keeping track of their effect on the position. Only consider windows attached to MAINW ;; before ATWIN.

```
                     (SETQ ATWREG (WINDOWREGION ATW))
                     (SETQ ATWHGHT (fetch (REGION HEIGHT) of ATWREG))
                     (SETQ ATWDTH (fetch (REGION WIDTH) of ATWREG))
                     (SETQ ATPOSONEDGE (SELECTQ [CDR (SETQ ATWHERE (WINDOWPROP ATW 'WHEREATTACHED]
                                                (JUSTIFY 'JUSTIFY)
                                                (CENTER 0)
                                                ((LEFT BOTTOM)
                                                    -1)
                                                1))
                     (SELECTQ (CAR ATWHERE)
                          (TOP (SELECTQ ATPOSONEDGE
                                    (JUSTIFY [SETQ TL (SETQ TC (SETQ TR (PLUS (MAX TL TC TR)
                                                                              ATWHGHT])
                                    (-1 (SETQ TL (PLUS TL ATWHGHT)))
                                    (0 (SETQ TC (PLUS TC ATWHGHT)))
                                    (1 (SETQ TR (PLUS TR ATWHGHT)))
                                    (SHOULDNT)))
                          (RIGHT (SELECTQ ATPOSONEDGE
                                      (JUSTIFY [SETQ RT (SETQ RC (SETQ RB (PLUS (MAX RT RC RB)
                                                                                ATWDTH])
                                      (1 (SETQ RT (PLUS RT ATWDTH)))
                                      (0 (SETQ RC (PLUS RC ATWDTH)))
                                      (-1 (SETQ RB (PLUS RB ATWDTH)))
                                      (SHOULDNT)))
                          (LEFT (SELECTQ ATPOSONEDGE
                                    (JUSTIFY [SETQ LT (SETQ LC (SETQ LB (DIFFERENCE (MIN LT LC LB)
                                                                                    ATWDTH])
                                    (1 (SETQ LT (DIFFERENCE LT ATWDTH)))
                                    (0 (SETQ LC (DIFFERENCE LC ATWDTH)))
                                    (-1 (SETQ LB (DIFFERENCE LB ATWDTH)))
                                    (SHOULDNT)))
                          (BOTTOM (SELECTQ ATPOSONEDGE
                                      (JUSTIFY [SETQ BL (SETQ BC (SETQ BR (DIFFERENCE (MAX BL BC BR)
                                                                                      ATWHGHT])
                                      (-1 (SETQ BL (DIFFERENCE BL ATWHGHT)))
                                      (0 (SETQ BC (DIFFERENCE BC ATWHGHT)))
                                      (1 (SETQ BR (DIFFERENCE BR ATWHGHT)))
                                      (SHOULDNT)))
                          (SHOULDNT)))                          ; now position the window
               (SETQ EXTENT (WINDOWREGION ATWIN))
               (SETQ ATWHGHT (fetch (REGION HEIGHT) of EXTENT))
```

```
            (SETQ ATWDTH (fetch (REGION WIDTH) of EXTENT))
            (COND
               ((EQ POSONEDGE 'JUSTIFY)
                (SHAPEW ATWIN (SELECTQ EDGE
                                   (TOP (CREATEREGION LT (ADD1 (MAX TL TC TR))
                                                      (IMAX (ADD1 (DIFFERENCE RT LT))
                                                            ATMINWIDTH)
                                                      ATWHGHT))
                                   (RIGHT (CREATEREGION (ADD1 (MAX RT RC RB))
                                                        BR ATWDTH (IMAX (ADD1 (DIFFERENCE TR BR))
                                                                        ATMINHEIGHT)))
                                   (LEFT (CREATEREGION (DIFFERENCE (MIN LT LC LB)
                                                                   ATWDTH)
                                                       BL ATWDTH (IMAX (ADD1 (DIFFERENCE TL BL))
                                                                       ATMINHEIGHT)))
                                   (BOTTOM (CREATEREGION LB (DIFFERENCE (MIN BL BC BR)
                                                                        ATWHGHT)
                                                         (IMAX (ADD1 (DIFFERENCE RB LB))
                                                               ATMINWIDTH)
                                                         ATWHGHT))
                                   NIL)))
               (T (SELECTQ EDGE
                       (TOP (SELECTQ POSONEDGE
                                 (1 (MOVEW ATWIN (ADD1 (DIFFERENCE RT ATWDTH))
                                           (ADD1 TR)))
                                 (0 (MOVEW ATWIN (CENTERINWIDTH ATWDTH MAINWEXTENT)
                                           (ADD1 TC)))
                                 (MOVEW ATWIN LT (ADD1 TL))))
                       (RIGHT (SELECTQ POSONEDGE
                                   (1 (MOVEW ATWIN (ADD1 RT)
                                             (ADD1 (DIFFERENCE TR ATWHGHT))))
                                   (0 (MOVEW ATWIN (ADD1 RC)
                                             (CENTERINHEIGHT ATWHGHT MAINWEXTENT)))
                                   (MOVEW ATWIN (ADD1 RB)
                                          BR)))
                       (LEFT (SELECTQ POSONEDGE
                                  (1 (MOVEW ATWIN (DIFFERENCE LT ATWDTH)
                                            (ADD1 (DIFFERENCE TL ATWHGHT))))
                                  (0 (MOVEW ATWIN (DIFFERENCE LC ATWDTH)
                                            (CENTERINHEIGHT ATWHGHT MAINWEXTENT)))
                                  (MOVEW ATWIN (DIFFERENCE LB ATWDTH)
                                         BL)))
                       (BOTTOM (SELECTQ POSONEDGE
                                   (1 (MOVEW ATWIN (ADD1 (DIFFERENCE RB ATWDTH))
                                             (DIFFERENCE BR ATWHGHT)))
                                   (0 (MOVEW ATWIN (CENTERINWIDTH ATWDTH MAINWEXTENT)
                                             (DIFFERENCE BC ATWHGHT)))
                                   (MOVEW ATWIN LB (DIFFERENCE BL ATWHGHT))))
                       NIL])
```

(OPENATTACHEDWINDOWS

```
  [LAMBDA (WINDOW)                                          ; Edited 12-Jan-87 11:11 by woz
```

;;; propagates opening to attached windows. since MOVEATTACHEDWINDOWTOPLACE punts when the main window is closed, must call it here to
;;; ensure the attached window is positioned.

```
    (for ATTACHEDWINDOW in (ATTACHEDWINDOWS WINDOW 'OPENW) do ;; reestablish the link from the attached window and the main
                                                              ;; window.
                                                              (WINDOWPROP ATTACHEDWINDOW 'MAINWINDOW WINDOW)
                                                              (MOVEATTACHEDWINDOWTOPLACE ATTACHEDWINDOW
                                                                     WINDOW)
                                                              (OPENW ATTACHEDWINDOW])
```

(RESHAPEALLWINDOWS

```
  [LAMBDA (MAINW NEWREGION MAINONLYFLG)                     ; Edited 24-Jan-97 11:27 by rmk:
                                                            (* DAHJr "11-Oct-86 18:57")
                                                            (* reshapes all of the windows in a group.)
                                                            (* calculate all of the attached window sizes)
    (PROG ((ATWINS (ATTACHEDWINDOWS MAINW 'SHAPEW))
           (MWXOFF 0)
           (MWYOFF 0)
           (NEWWIDTH (fetch (REGION WIDTH) of NEWREGION))
           (NEWHEIGHT (fetch (REGION HEIGHT) of NEWREGION))
           FIXEDVAR TOTALNOWSIZE EXPANSIONWIDTH EXPANSIONHEIGHT NEWEXPANDABLEWIDTH NEWEXPANDABLEHEIGHT
           ATWINSINFO EXCESS NOW)
          [COND
             ((NULL ATWINS)
              (RETURN (SHAPEW1 MAINW NEWREGION)))
             (MAINONLYFLG (SHAPEW1 MAINW NEWREGION)
                    (RETURN (\RESHAPEATTACHEDWINDOWSAROUNDMAINW MAINW (\BREAKAPARTATWSTRUCTURE
                                                                       (CDR (\BUILDATWSTRUCTURE MAINW
                                                                                  ATWINS]
          (SETQ TOTALNOWSIZE (WINDOWSIZE MAINW))
```

```
(* calculate the amount of the total size that is available to change.
This ignores the case where a window can only expand 5 but its share would be 10 but it is easy and better than nothing.)

(SETQ ATWINSINFO (\BUILDATWSTRUCTURE MAINW ATWINS))
(\ALLOCMINIMUMSIZES ATWINSINFO 0 0)
[SETQ EXPANSIONWIDTH (IDIFFERENCE (CAR TOTALNOWSIZE)
                                  (SETQ FIXEDVAR (\TOTALFIXEDWIDTH ATWINSINFO]
(SETQ NEWEXPANDABLEWIDTH (IMAX (DIFFERENCE NEWWIDTH FIXEDVAR)
                               0))
[SETQ EXPANSIONHEIGHT (IDIFFERENCE (CDR TOTALNOWSIZE)
                                   (SETQ FIXEDVAR (\TOTALFIXEDHEIGHT ATWINSINFO]
(SETQ NEWEXPANDABLEHEIGHT (IMAX (DIFFERENCE NEWHEIGHT FIXEDVAR)
                                0))

(* make a pass through allocating each window a portion of the space that is in excess of the minimum.
In this pass, the grouped windows are treated as a whole. (If there is no space in excess of minimum, allocate on the basis
of the actual size of the windows -- Austin Henderson 10-11-86))

[for ATWINFO in ATWINSINFO do [COND
                                  [(EQP EXPANSIONWIDTH 0)
                                   (\SETWINFOXSIZE ATWINFO (\SHAREOFXTRAX ATWINFO (fetch (
                                                                                          RESHAPINGWINDOWDATA
                                                                                              ATNOWX)
                                                                                           of ATWINFO)
                                                                                   (CAR TOTALNOWSIZE]
                                  (T (\SETWINFOXSIZE ATWINFO (\SHAREOFXTRAX ATWINFO NEWEXPANDABLEWIDTH
                                                                             EXPANSIONWIDTH]
                               (COND
                                  [(EQP EXPANSIONHEIGHT 0)
                                   (\SETWINFOYSIZE ATWINFO (\SHAREOFXTRAY ATWINFO (fetch (
                                                                                          RESHAPINGWINDOWDATA
                                                                                              ATNOWY)
                                                                                           of ATWINFO)
                                                                                   (CDR TOTALNOWSIZE]
                                  (T (\SETWINFOYSIZE ATWINFO (\SHAREOFXTRAY ATWINFO NEWEXPANDABLEHEIGHT
                                                                             EXPANSIONHEIGHT]
                                                                        (* now go through allocate the space within the groups of
                                                                        windows.)
(for ATWINFO in ATWINSINFO when (LISTP (fetch (RESHAPINGWINDOWDATA ATTACHEDW) of ATWINFO))
   do (\ALLOCSPACETOGROUPEDWINDOWS ATWINFO))

(* calculate how much of the available space was actually allocated.
This is necessary because some of the windows may have reached their maximum and hence left some space not used.
The extra is given to the main window. The main window is shaped first so that user reshape functions can determine its
size and shape as they do their thing.)

(SETQ TOTALNOWSIZE (\TOTALPROPOSEDSIZE ATWINSINFO))
[COND
   ((NEQ (SETQ EXCESS (IDIFFERENCE NEWWIDTH (CAR TOTALNOWSIZE)))
         0)

(* Feed the excess width to any windows that will take it, starting with the main window)

    (for ATWINFO in ATWINSINFO do (SETQ EXCESS (IDIFFERENCE EXCESS
                                                            (IDIFFERENCE (\SETWINFOXSIZE
                                                                          ATWINFO
                                                                          (IPLUS (SETQ NOW
                                                                                       (fetch (RESHAPINGWINDOWDATA
                                                                                                  ATXSIZE)
                                                                                              of ATWINFO))
                                                                                 EXCESS))
                                                                         NOW)))
       repeatuntil (EQ EXCESS 0]
[COND
   ((NEQ (SETQ EXCESS (IDIFFERENCE NEWHEIGHT (CDR TOTALNOWSIZE)))
         0)

(* Feed the excess width to any windows that will take it, starting with the main window)

    (for ATWINFO in ATWINSINFO do (SETQ EXCESS (IDIFFERENCE EXCESS
                                                            (IDIFFERENCE (\SETWINFOYSIZE
                                                                          ATWINFO
                                                                          (IPLUS (SETQ NOW
                                                                                       (fetch (RESHAPINGWINDOWDATA
                                                                                                  ATYSIZE)
                                                                                              of ATWINFO))
                                                                                 EXCESS))
                                                                         NOW)))
       repeatuntil (EQ EXCESS 0]
(for ATWINFO in ATWINSINFO do                                   (* Calculate new position of main window inside the total region)
                              (SELECTQ (fetch (RESHAPINGWINDOWDATA ATEDGE) of ATWINFO)
                                   (BOTTOM (add MWYOFF (fetch (RESHAPINGWINDOWDATA ATYSIZE) of ATWINFO)))
                                   (LEFT (add MWXOFF (fetch (RESHAPINGWINDOWDATA ATXSIZE) of ATWINFO)))
                                   NIL))
[SHAPEW1 MAINW (CREATEREGION (IPLUS MWXOFF (fetch (REGION LEFT) of NEWREGION))
                             (IPLUS MWYOFF (fetch (REGION BOTTOM) of NEWREGION))
                             (fetch (RESHAPINGWINDOWDATA ATXSIZE) of (CAR ATWINSINFO))
```

```
                              (fetch (RESHAPINGWINDOWDATA ATYSIZE) of (CAR ATWINSINFO]
                                                                    (* reshape all of the attached windows according to the
                                                                    calculated new sizes.)
          (\RESHAPEATTACHEDWINDOWSAROUNDMAINW MAINW (\BREAKAPARTATWSTRUCTURE (CDR ATWINSINFO])
```

(\TOTALPROPOSEDSIZE

```
  [LAMBDA (ATWSINFO PWIDTH PHEIGHT)                                 (* rrb " 9-Dec-83 16:12")
                                                                    (* determines the width of the windows that do not change their
    size.)
    (COND
       [ATWSINFO (PROG (THISWID THISHEIGHT THISMINWIDTH THISMINHEIGHT (ATW (CAR ATWSINFO))
                             (RESTATWS (CDR ATWSINFO)))
                       (SETQ THISMINWIDTH (fetch (RESHAPINGWINDOWDATA ATMINX) of ATW))
                       (SETQ THISMINHEIGHT (fetch (RESHAPINGWINDOWDATA ATMINY) of ATW))
                       (SETQ THISWID (fetch (RESHAPINGWINDOWDATA ATXSIZE) of ATW))
                       (SETQ THISHEIGHT (fetch (RESHAPINGWINDOWDATA ATYSIZE) of ATW))
                       (RETURN (SELECTQ (fetch (RESHAPINGWINDOWDATA ATEDGE) of ATW)
                                   ((LEFT RIGHT)
                                       (\TOTALPROPOSEDSIZE RESTATWS (IPLUS PWIDTH THISWID)
                                             (IMAX PHEIGHT THISMINHEIGHT)))
                                   ((TOP BOTTOM)
                                       (\TOTALPROPOSEDSIZE RESTATWS (IMAX PWIDTH THISMINWIDTH)
                                             (IPLUS PHEIGHT THISHEIGHT)))
                                   (PROGN                           (* this is the main window.)
                                          (\TOTALPROPOSEDSIZE RESTATWS THISWID THISHEIGHT]
       (T (CONS PWIDTH PHEIGHT])
```

(SHRINKATTACHEDWINDOWS

```
  [LAMBDA (WINDOW)                                                  ; Edited  5-Mar-87 11:06 by lal
                                                                    ; propagates shrinking to attached windows.
                                                                    ; doesn't actually shrink, just closes and evaluates the shrink
                                                                    ; functions.
    (for ATTACHEDWINDOW in (ATTACHEDWINDOWS WINDOW 'SHRINKW)
       do                                                           ; Don't shrink the attached windows if they say not to
          (if (EQ (DOUSERFNS (WINDOWPROP ATTACHEDWINDOW 'SHRINKFN)
                             ATTACHEDWINDOW T)
                  'DON'T)
            then NIL
           else (\CLOSEW1 ATTACHEDWINDOW])
```

(TOPATTACHEDWINDOWS

```
  [LAMBDA (WINDOW RECURSIVE)                                        ; Edited 17-Aug-88 19:46 by jds
    ;; if WINDOW is root, propagate totoping down tree
    (COND
       ([OR RECURSIVE (NULL (WINDOWPROP WINDOW 'MAINWINDOW]
        (for ATTACHEDWINDOW in (ATTACHEDWINDOWS WINDOW 'TOTOPW) do  ;; walk tree, totoping
                                                                    (TOTOPW ATTACHEDWINDOW T)
                                                                    (TOPATTACHEDWINDOWS ATTACHEDWINDOW T])
```

(UNMAKEMAINWINDOW

```
  [LAMBDA (MAINWINDOW)                                              (* rrb "21-NOV-83 14:37")
                                                                    (* the last attached window has been detached, clear any
                                                                    relevant window properties.)
    (WINDOWDELPROP MAINWINDOW 'TOTOPFN (FUNCTION TOPATTACHEDWINDOWS))
    (WINDOWDELPROP MAINWINDOW 'CLOSEFN (FUNCTION CLOSEATTACHEDWINDOWS))
    (WINDOWDELPROP MAINWINDOW 'OPENFN (FUNCTION OPENATTACHEDWINDOWS))
    (WINDOWDELPROP MAINWINDOW 'SHRINKFN (FUNCTION SHRINKATTACHEDWINDOWS))
    (WINDOWDELPROP MAINWINDOW 'EXPANDFN (FUNCTION EXPANDATTACHEDWINDOWS))
    (WINDOWPROP MAINWINDOW 'CALCULATEREGIONFN NIL)
    (WINDOWDELPROP MAINWINDOW 'MOVEFN (FUNCTION MOVEATTACHEDWINDOWS))
    (WINDOWPROP MAINWINDOW 'DOSHAPEFN NIL])
```

(UPIQUOTIENT

```
  [LAMBDA (N DIVISOR)                                               (* rrb "20-NOV-83 13:41")

          (* returns the smallest integer such that DIVISOR * that number is greater than or equal to N.)

    (IQUOTIENT (IPLUS N (SUB1 DIVISOR))
           DIVISOR])
```

(WINDOWPOSITION

```
  [LAMBDA (WINDOW)                                                  (* rrb "27-OCT-83 15:41")
    (PROG [(REG (WINDOWPROP WINDOW 'REGION]
          (RETURN (create POSITION
                          XCOORD _ (fetch (REGION LEFT) of REG)
                          YCOORD _ (fetch (REGION BOTTOM) of REG])
```

(WINDOWSIZE

```
  [LAMBDA (WINDOW)                                        (* rrb " 6-Dec-83 17:45")
                                                          (* returns the size (WIDTH . HEIGHT) of a window and its
                                                          attached windows if any.)
    (PROG ((EXT (WINDOWREGION WINDOW)))
```

(* this will give the wrong answer if the attached windows have been moved and have gaps between them.)

```
          (RETURN (CONS (fetch (REGION WIDTH) of EXT)
                        (fetch (REGION HEIGHT) of EXT])
```

(\ALLOCMINIMUMSIZES

```
  [LAMBDA (ATWSINFO INTMINWIDTH INTMINHEIGHT NOWWIDTH NOWHEIGHT) (* rrb " 7-Jan-86 14:37")
```

(* allocates to each window in the list of window structures ATWSINFO the minimum space it should get based on the minimums of all of the other windows in ATWSINFO)

```
                                                          (* returns the minimum size dictated by the first window on
                                                          ATWSINFO)
    (COND
       [ATWSINFO (PROG ((ATW (CAR ATWSINFO))
                        (THISMINWIDTH INTMINWIDTH)
                        (THISMINHEIGHT INTMINHEIGHT)
                        EXTSIZE EDGE WINDOWPILE RESTATWS FIXEDVAR EXPANSIONWIDTH NEWEXPANDABLEWIDTH NEWWIDTH
                        EXPANSIONHEIGHT NEWEXPANDABLEHEIGHT NEWHEIGHT)
                       (SETQ RESTATWS ATWSINFO)
                       (SELECTQ (fetch (RESHAPINGWINDOWDATA ATEDGE) of ATW)
                           ((LEFT RIGHT)
```

(* collect a list of windows that fit on the sides. This is so that any excess size imposed by windows further out can be allocated among all of the windows piled together.)

```
                                  (for WININFO in RESTATWS until [NOT (FMEMB (fetch (RESHAPINGWINDOWDATA ATEDGE)
                                                                                of WININFO)
                                                                            '(LEFT RIGHT]
                                     do (SETQ THISMINHEIGHT (IMAX THISMINHEIGHT (fetch (RESHAPINGWINDOWDATA ATMINY)
                                                                                       of WININFO)))
                                                                  (* calculate the current size of this pile of windows.)
                                        (SETQ NOWHEIGHT (IMAX NOWHEIGHT (fetch (RESHAPINGWINDOWDATA ATNOWY)
                                                                               of WININFO)))
                                        (SETQ NOWWIDTH (IPLUS NOWWIDTH (fetch (RESHAPINGWINDOWDATA ATNOWX)
                                                                              of WININFO)))
                                        (SETQ THISMINWIDTH (IPLUS THISMINWIDTH (fetch (RESHAPINGWINDOWDATA ATMINX)
                                                                                      of WININFO)))
                                        (SETQ WINDOWPILE (CONS WININFO WINDOWPILE))
                                        (SETQ RESTATWS (CDR RESTATWS)))
```

(* calculate the dimensions imposed by the minimum sizes of windows further out on the attached window list.)

```
                                  [SETQ NEWWIDTH (CAR (SETQ EXTSIZE (\ALLOCMINIMUMSIZES RESTATWS THISMINWIDTH
                                                                                        THISMINHEIGHT NOWWIDTH NOWHEIGHT]
                                  (SETQ NEWHEIGHT (CDR EXTSIZE))   (* compute how much of the current width can be expanded.)
                                  [SETQ EXPANSIONWIDTH (IDIFFERENCE NOWWIDTH (SETQ FIXEDVAR (\TOTALFIXEDWIDTH
                                                                                             WINDOWPILE]
                                  (SETQ NEWEXPANDABLEWIDTH (IMAX (DIFFERENCE NEWWIDTH FIXEDVAR)
                                                                 0))
                                                                  (* allocate to each window on this level of the pile its share.)
                                  (for WININFO in WINDOWPILE do (\SETWINFOXSIZE WININFO (\SHAREOFXTRAX WININFO
                                                                                                       NEWEXPANDABLEWIDTH
                                                                                                       EXPANSIONWIDTH))
                                                                (\SETWINFOYSIZE WININFO NEWHEIGHT))
                                  (RETURN (CONS (IDIFFERENCE NEWWIDTH (for WININFO in WINDOWPILE
                                                                         sum
```

(* determine how much was actually allocated to the windows in the pile and give the rest to the initial window.)

```
                                                                                (fetch (RESHAPINGWINDOWDATA ATXSIZE)
                                                                                   of WININFO)))
                                                NEWHEIGHT)))
                           ((TOP BOTTOM)
```

(* collect a list of windows that fit on the sides. This is so that any excess size imposed by windows further out can be allocated among all of the windows piled together.)

```
                                  (for WININFO in RESTATWS until [NOT (FMEMB (fetch (RESHAPINGWINDOWDATA ATEDGE)
                                                                                of WININFO)
                                                                            '(TOP BOTTOM]
                                     do (SETQ THISMINHEIGHT (IPLUS THISMINHEIGHT (fetch (RESHAPINGWINDOWDATA ATMINY
                                                                                                                )
                                                                                       of WININFO)))
                                        (SETQ NOWHEIGHT (IPLUS NOWHEIGHT (fetch (RESHAPINGWINDOWDATA ATNOWY)
                                                                                of WININFO)))
                                        (SETQ NOWWIDTH (IMAX NOWWIDTH (fetch (RESHAPINGWINDOWDATA ATNOWX)
                                                                             of WININFO)))
                                        (SETQ THISMINWIDTH (IMAX THISMINWIDTH (fetch (RESHAPINGWINDOWDATA ATMINX)
                                                                                     of WININFO)))
                                        (SETQ WINDOWPILE (CONS WININFO WINDOWPILE))
                                        (SETQ RESTATWS (CDR RESTATWS)))
```

```
        (* calculate the dimensions imposed by the minimum sizes of windows further out on the attached window list.)

                                [SETQ NEWWIDTH (CAR (SETQ EXTSIZE (\ALLOCMINIMUMSIZES RESTATWS THISMINWIDTH
                                                                                       THISMINHEIGHT NOWWIDTH NOWHEIGHT]
                                (SETQ NEWHEIGHT (CDR EXTSIZE))   (* compute how much of the current height can be expanded.)
                                [SETQ EXPANSIONHEIGHT (IDIFFERENCE NOWHEIGHT (SETQ FIXEDVAR (\TOTALFIXEDHEIGHT
                                                                                              WINDOWPILE]
                                (SETQ NEWEXPANDABLEHEIGHT (IMAX (DIFFERENCE NEWHEIGHT FIXEDVAR)
                                                                0))
                                                                (* allocate to each window on this level of the pile its share.)
                                (for WININFO in WINDOWPILE do (\SETWINFOXSIZE WININFO NEWWIDTH)
                                                              (\SETWINFOYSIZE WININFO (\SHAREOFXTRAY WININFO
                                                                                                    NEWEXPANDABLEHEIGHT
                                                                                                    EXPANSIONHEIGHT)))
                                [RETURN (CONS NEWWIDTH (IDIFFERENCE NEWHEIGHT (for WININFO in WINDOWPILE
                                                                                  sum

        (* determine how much was actually allocated to the windows in the pile and give the rest to the initial window.)

                                                                                                (fetch (RESHAPINGWINDOWDATA
                                                                                                       ATYSIZE)
                                                                                                   of WININFO])
                                (PROGN                                  (* this is the main window.)
                                       (SETQ EXTSIZE (\ALLOCMINIMUMSIZES (CDR ATWSINFO)
                                                                         (fetch (RESHAPINGWINDOWDATA ATMINX) of ATW)
                                                                         (fetch (RESHAPINGWINDOWDATA ATMINY) of ATW)
                                                                         (fetch (RESHAPINGWINDOWDATA ATNOWX) of ATW)
                                                                         (fetch (RESHAPINGWINDOWDATA ATNOWY) of ATW)))
                                       (\SETWINFOXSIZE ATW (CAR EXTSIZE))
                                       (\SETWINFOYSIZE ATW (CDR EXTSIZE]
                (T (CONS INTMINWIDTH INTMINHEIGHT])
```

(\ALLOCSPACETOGROUPEDWINDOWS

```
  [LAMBDA (WGROUPINFO)                                  (* rrb " 9-Dec-83 15:15")
                                                        (* allocates space to the windows on EXTBUCKETS.)
    (SELECTQ (fetch (RESHAPINGWINDOWDATA ATEDGE) of WGROUPINFO)
        ((LEFT RIGHT)                                   (* allocate in X)
             (\ALLOCWIDTHTOGROUPEDWINDOW WGROUPINFO))
        (\ALLOCHEIGHTTOGROUPEDWINDOW WGROUPINFO])
```

(\TOTALFIXEDHEIGHT

```
  [LAMBDA (ATWSINFO)                                    (* bvm%: "12-Apr-84 12:30")
                                                        (* determines the height of the windows that do not change their
    size.)
    (bind (MAXEDW _ 0)
          THISMINHEIGHT for ATW in ATWSINFO when (AND [NOT (AND (EQ (fetch (RESHAPINGWINDOWDATA ATWHEREONEDGE)
                                                                       of ATW)
                                                                  'JUSTIFY)
                                                            (FMEMB (fetch (RESHAPINGWINDOWDATA ATEDGE)
                                                                      of ATW)
                                                                   '(LEFT RIGHT]
                                                      (EQ (SETQ THISMINHEIGHT (fetch (RESHAPINGWINDOWDATA ATMINY)
                                                                                 of ATW))
                                                          (fetch (RESHAPINGWINDOWDATA ATMAXY) of ATW)))
       sum THISMINHEIGHT])
```

(\TOTALFIXEDWIDTH

```
  [LAMBDA (ATWSINFO)                                    (* bvm%: "12-Apr-84 12:30")

        (* determines the width of the windows that do not change their size.
        A window that is JUSTIFIED and is on the TOP or BOTTOM is not counted since it will be stretched as needed.)

    (bind (MAXEDW _ 0)
          THISMINWIDTH for ATW in ATWSINFO when (AND [NOT (AND (EQ (fetch (RESHAPINGWINDOWDATA ATWHEREONEDGE)
                                                                       of ATW)
                                                                  'JUSTIFY)
                                                            (FMEMB (fetch (RESHAPINGWINDOWDATA ATEDGE)
                                                                      of ATW)
                                                                   '(TOP BOTTOM]
                                                      (EQ (SETQ THISMINWIDTH (fetch (RESHAPINGWINDOWDATA ATMINX)
                                                                                of ATW))
                                                          (fetch (RESHAPINGWINDOWDATA ATMAXX) of ATW)))
       sum THISMINWIDTH])
```

(\ALLOCHEIGHTTOGROUPEDWINDOW

```
  [LAMBDA (WGROUPINFO)                                  (* rrb "15-Dec-83 10:19")

        (* allocates height to a collection of window all of which are attached to the top, or bottom edge but at different places on the
        edge. EXTBUCKET is a list of those window at the left center and right of the edge.
        Also sets the width field in the window information structure.)

    (PROG ((EXTBUCKET (for WHEREONEDGE in '(LEFT CENTER RIGHT)
```

```
                    collect (for ATW in (fetch (RESHAPINGWINDOWDATA ATTACHEDW) of WGROUPINFO)
                               when (EQ (fetch (RESHAPINGWINDOWDATA ATWHEREONEDGE) of ATW)
                                        WHEREONEDGE)
                               collect ATW)))
     (TOTALNOWSIZE (fetch (RESHAPINGWINDOWDATA ATNOWY) of WGROUPINFO))
     [TOTALXTRA (IDIFFERENCE (fetch (RESHAPINGWINDOWDATA ATYSIZE) of WGROUPINFO)
                             (\TOTALFIXEDHEIGHT (fetch (RESHAPINGWINDOWDATA ATTACHEDW) of WGROUPINFO]
     SHARE HEIGHTS MAXHEIGHT NOWSIZE MAXSIZE NEWSIZE XTRA)
    [SETQ HEIGHTS (for ATWS in EXTBUCKET collect (for ATW in ATWS sum
```

(* leave the width the same. Possibly this should expand but calculation depends on width of other windows attached next to this one.)

```
                                                                (\SETWINFOXSIZE ATW (fetch (
                                                                      RESHAPINGWINDOWDATA
                                                                           ATNOWX)
                                                                    of ATW))
                                                                (\SETWINFOYSIZE ATW
                                                                      (\SHAREOFXTRAY ATW TOTALXTRA
                                                                           TOTALNOWSIZE]
    (SETQ MAXHEIGHT (APPLY (FUNCTION MAX)
                           HEIGHTS))                        (* keep track of the width as part of the sizing.)
                                                            (* allocate extra to places which are not maximum yet.)
    (for ATWS in EXTBUCKET as HEIGHT in HEIGHTS unless (NULL ATWS)
       do (COND
             ((NEQ HEIGHT MAXHEIGHT)
              (SETQ XTRA (IDIFFERENCE MAXHEIGHT HEIGHT))
              (until (OR (NULL ATWS)
                         (EQ 0 XTRA))
                 do (SETQ SHARE (UPIQUOTIENT XTRA (LENGTH ATWS)))
```

(* UPIQUOTIENT is used to make sure that all of the space is allocated.
Having the shares be greater than the total means that before the share is given to the window, a check must be made to see that the space in fact exists. This is done by the IMAX in calculating NEWSIZE.)

(* THIS ALGORITHM HAS THE BAD PROPERTY THAT THE FIRST N-1 windows might get one point too much and cause the last window to be N-2 points smaller than it would be in the perfect case.)

```
                    (for ATW in ATWS do (COND
                                  ((EQ (SETQ NOWSIZE (fetch (RESHAPINGWINDOWDATA ATYSIZE)
                                                        of ATW))
                                       (SETQ MAXSIZE (fetch (RESHAPINGWINDOWDATA ATMAXY)
                                                        of ATW)))
                                                    (* window has reached max, remove it from getting more space.)
                                   (SETQ ATWS (REMOVE ATW ATWS)))
                                  ((PROGN           (* NEWSIZE needs to be calculated whether MAXSIZE exists or
                                                    not.)
                                          (SETQ NEWSIZE (PLUS (IMAX SHARE XTRA)
                                                              NOWSIZE))
                                          (AND MAXSIZE (LESSP MAXSIZE NEWSIZE)))
                                                    (* add only enough to reach maximum.)
                                   (SETQ XTRA (IDIFFERENCE XTRA (IDIFFERENCE MAXSIZE NOWSIZE)))
                                   (SETQ ATWS (REMOVE ATW ATWS))
                                   (replace (RESHAPINGWINDOWDATA ATYSIZE) of ATW with MAXSIZE))
                                  (T (SETQ XTRA (IDIFFERENCE XTRA (IDIFFERENCE NEWSIZE NOWSIZE)))
                                     (replace (RESHAPINGWINDOWDATA ATYSIZE) of ATW with NEWSIZE])
```

(**ALLOCWIDTHTOGROUPEDWINDOW**

```
  [LAMBDA (WGROUPINFO)                                  (* rrb "15-Dec-83 10:19")
```

(* allocates width to a collection of window all of which are attached to the left or right edge but at different places on the edge. EXTBUCKET is a list of those window at the top, center and bottom of the edge.)

```
    (PROG ((EXTBUCKET (for WHEREONEDGE in '(TOP CENTER BOTTOM)
                         collect (for ATW in (fetch (RESHAPINGWINDOWDATA ATTACHEDW) of WGROUPINFO)
                                    when (EQ (fetch (RESHAPINGWINDOWDATA ATWHEREONEDGE) of ATW)
                                             WHEREONEDGE)
                                    collect ATW)))
     (TOTALNOWSIZE (fetch (RESHAPINGWINDOWDATA ATNOWX) of WGROUPINFO))
     (TOTALXTRA (fetch (RESHAPINGWINDOWDATA ATXSIZE) of WGROUPINFO))
     SHARE WIDTHS MAXWIDTH NOWSIZE MAXSIZE NEWSIZE XTRA)
    [SETQ WIDTHS (for ATWS in EXTBUCKET collect (for ATW in ATWS sum
```

(* leave height the same as it was. Could expand if neighbors weren't too big but haven't bothered.)

```
                                                                (\SETWINFOYSIZE ATW (fetch (
                                                                      RESHAPINGWINDOWDATA
                                                                           ATNOWY)
                                                                    of ATW))
                                                                (\SETWINFOXSIZE ATW (\SHAREOFXTRAX ATW
                                                                           TOTALXTRA
                                                                           TOTALNOWSIZE]
    (SETQ MAXWIDTH (APPLY (FUNCTION MAX)
                          WIDTHS))                      (* keep track of the width as part of the sizing.)
                                                        (* allocate extra to places which are not maximum yet.)
    (for ATWS in EXTBUCKET as WIDTH in WIDTHS unless (NULL ATWS)
```

```
        do (COND
                ((NEQ WIDTH MAXWIDTH)
                 (SETQ XTRA (IDIFFERENCE MAXWIDTH WIDTH))
                 (until (OR (NULL ATWS)
                            (EQ 0 XTRA))
                    do (SETQ SHARE (UPIQUOTIENT XTRA (LENGTH ATWS)))
```

(* UPIQUOTIENT is used to make sure that all of the space is allocated.
Having the shares be greater than the total means that before the share is given to the window, a check must be made to see that the space in fact exists. This is done by the IMAX in calculating NEWSIZE.)

(* THIS ALGORITHM HAS THE BAD PROPERTY THAT THE FIRST N-1 windows might get one point too much and cause the last window to be N-2 points smaller than it would be in the perfect case.)

```
                        (for ATW in ATWS do (COND
                                               ((EQ (SETQ NOWSIZE (fetch (RESHAPINGWINDOWDATA ATXSIZE)
                                                                     of ATW))
                                                    (SETQ MAXSIZE (fetch (RESHAPINGWINDOWDATA ATMAXX)
                                                                     of ATW)))
                                                                  (* window has reached max, remove it from getting more space.)
                                                (SETQ ATWS (REMOVE ATW ATWS)))
                                               ((PROGN            (* NEWSIZE needs to be calculated whether MAXSIZE exists or
                                                                  not.)
                                                       (SETQ NEWSIZE (PLUS (IMAX SHARE XTRA)
                                                                           NOWSIZE))
                                                       (AND MAXSIZE (LESSP MAXSIZE NEWSIZE)))
                                                                  (* add only enough to reach maximum.)
                                                (SETQ XTRA (IDIFFERENCE XTRA (IDIFFERENCE MAXSIZE NOWSIZE)))
                                                (SETQ ATWS (REMOVE ATW ATWS))
                                                (replace (RESHAPINGWINDOWDATA ATXSIZE) of ATW with MAXSIZE))
                                               (T (SETQ XTRA (IDIFFERENCE XTRA (IDIFFERENCE NEWSIZE NOWSIZE)))
                                                  (replace (RESHAPINGWINDOWDATA ATXSIZE) of ATW with NEWSIZE])
```

(**ATWGROUPSIZE**

```
  [LAMBDA (ATWS)                                                    (* rrb " 8-Jan-86 11:48")
                                                                    (* returns the size of a group of attached window information
                                                                    structures.)
    (COND
       [ATWS (PROG [(EXTREGION (WINDOWREGION (fetch (RESHAPINGWINDOWDATA ATTACHEDW) of (CAR ATWS]
                   (for ATW in (CDR ATWS) do (SETQ EXTREGION (UNIONREGIONS (WINDOWREGION (fetch (
                                                                                  RESHAPINGWINDOWDATA
                                                                                   ATTACHEDW)
                                                                                of ATW))
                                                             EXTREGION)))
                   (RETURN (CONS (fetch (REGION WIDTH) of EXTREGION)
                                 (fetch (REGION HEIGHT) of EXTREGION]
       (T (CONS 0 0])
```

(**BREAKAPARTATWSTRUCTURE**

```
  [LAMBDA (ATWLST)                                                  (* rrb "14-Dec-83 12:45")
                                                                    (* breaks apart the window grouping that are in ATWLST)
    (for ATW in ATWLST join (COND
                               [(APPEND (LISTP (fetch (RESHAPINGWINDOWDATA ATTACHEDW) of ATW]
                               (T (CONS ATW])
```

(**BUILDATWSTRUCTURE**

```
  [LAMBDA (MAINW ATTACHEDWINDOWS)                                   (* bvm%: "29-Dec-83 15:58")
```

(* builds a structure which has place holders for each window or collection of windows on an edge.)

```
    (PROG ((EDGEBUCKETLIST (for SIDE in '(TOP RIGHT BOTTOM LEFT) collect (CONS SIDE NIL)))
           EDGEBUCKET WHEREAT ATWINFO WHEREONEDGE PLACEHOLDERATW)
          (RETURN (CONS (LIST MAINW NIL (MINIMUMMAINWINDOWSIZE MAINW)
                                (MAXIMUMMAINWINDOWSIZE MAINW)
                                (CONS 0 0)
                                (CONS [fetch (REGION WIDTH) of (SETQ WHEREAT (WINDOWPROP MAINW 'REGION]
                                      (fetch (REGION HEIGHT) of WHEREAT)))
                        (for ATWIN in ATTACHEDWINDOWS join
```

(* collect all of the information about an attached window and leave a place for its determined size.)

```
                                                        (SETQ WHEREAT (WINDOWPROP ATWIN 'WHEREATTACHED))
                                                        (SETQ EDGEBUCKET (FASSOC (CAR WHEREAT)
                                                                                 EDGEBUCKETLIST))
                                                        (SETQ ATWINFO (LIST ATWIN WHEREAT (MINIMUMWINDOWSIZE
                                                                                             ATWIN)
                                                                                  (MAXIMUMWINDOWSIZE ATWIN)
                                                                                  (CONS 0 0)
                                                                                  (WINDOWSIZE ATWIN)))
                                                        [COND
                                                           ((EQ (SETQ WHEREONEDGE (fetch ATWHEREONEDGE
                                                                                     of ATWINFO))
                                                                'JUSTIFY)
```

```
(* when a window that fits all the way across is encountered, set the fields in the group information structure and clear it.
Then return the structure for this window.)

                                                    (COND
                                                       ((CDR EDGEBUCKET)

(* compute the group mins from the windows that don't fit all the way across on this edge.)

                                                           (\SETGROUPMIN (CDR EDGEBUCKET))
                                                           (RPLACD EDGEBUCKET NIL)))
                                                     (CONS ATWINFO))
                                                    (T

(* if this window doesn't fit all the way across, put it in the group that is being formed for this edge.
If there isn't a group yet, form one and return it so that it will take this place in the structure.)

                                                  (COND
                                                     ((CDR EDGEBUCKET)
                                                      (* group already exists)
                                                       (NCONC1 (CADR EDGEBUCKET)
                                                               ATWINFO)
                                                       NIL)
                                                      (T

(* make a with dummy value in its fields and return it to save its place in the attached window list.)

                                                          (SETQ PLACEHOLDERATW
                                                           (LIST (LIST ATWINFO)
                                                                 WHEREAT
                                                                 (CONS 0 0)
                                                                 (CONS NIL NIL)
                                                                 (CONS 0 0)
                                                                 (CONS 0 0)))
                                                          (RPLACD EDGEBUCKET PLACEHOLDERATW)
                                                          (LIST PLACEHOLDERATW]
                                finally (for old EDGEBUCKET in EDGEBUCKETLIST do

(* allocate space for those groups that don't have any windows outside them that fit all the way across.)

                                                       (COND
                                                          ((CDR EDGEBUCKET)

(* compute the group mins from the windows that don't fit all the way across on this edge.)

                                                           (\SETGROUPMIN (CDR EDGEBUCKET))
                                                           (RPLACD EDGEBUCKET NIL])
```

(\LIMITBYMAX

```
[LAMBDA (N MAX)                                    (* limits the size of N to MAX)
  (COND
     (MAX (IMIN N MAX))
     (T N])
```

(\LIMITBYMIN

```
[LAMBDA (N MIN)                                    (* bvm%: "10-Nov-84 15:14")
                                                   (* limits the size of N to MIN)
  (COND
     (MIN (IMAX N MIN))
     (T N])
```

(\MAXHEIGHTOFGROUP

```
[LAMBDA (ATWINFOS)                                 (* rrb "14-Dec-83 12:22")

(* returns the largest minimum height of a group of windows all of which are on the same edge.
It must look at each position on the edge {left center right} and sum over the elements that are on that position.)

   (for WHEREONEDGE in '(LEFT CENTER RIGHT) largest (for ATW in ATWINFOS when (EQ (fetch (RESHAPINGWINDOWDATA
                                                                                          ATWHEREONEDGE)
                                                                                    of ATW)
                                                                                  WHEREONEDGE)
                                                                sum (fetch (RESHAPINGWINDOWDATA ATMINY) of ATW))
      finally (RETURN $$EXTREME])
```

(\MAXWIDTHOFGROUP

```
[LAMBDA (ATWINFOS)                                 (* rrb "15-Dec-83 10:21")

(* returns the largest minimum width of a group of windows all of which are on the same edge.
It must look at each position on the edge {top center bottom} and sum over the elements that are on that position.)

   (for WHEREONEDGE in '(TOP CENTER BOTTOM) largest (for ATW in ATWINFOS when (EQ (fetch (RESHAPINGWINDOWDATA
                                                                                          ATWHEREONEDGE)
                                                                                    of ATW)
                                                                                  WHEREONEDGE)
```

```
                                                    sum (fetch (RESHAPINGWINDOWDATA ATMINX) of ATW))
      finally (RETURN $$EXTREME])
```

(\RESHAPEATTACHEDWINDOWSAROUNDMAINW

```
 [LAMBDA (MAINW ATWINSINFO)                                    (* rrb " 3-Oct-84 16:33")
   (PROG ((MAINWEXTENT (WINDOWPROP MAINW 'REGION))
          EXTENT ATWHGHT ATWDTH TL TC TR RT RC RB BR BC BL LB LC LT)
         [SETQ TL (SETQ TC (SETQ TR (fetch (REGION TOP) of MAINWEXTENT]
         [SETQ RT (SETQ RC (SETQ RB (fetch (REGION RIGHT) of MAINWEXTENT]
         [SETQ BR (SETQ BC (SETQ BL (fetch (REGION BOTTOM) of MAINWEXTENT]
         [SETQ LB (SETQ LC (SETQ LT (fetch (REGION LEFT) of MAINWEXTENT]
         (bind ATWHERE ATW TEMP for ATWINFO in ATWINSINFO
            do
         (* go through the attached windows shaping them and keeping track of their effect on the position.)
               [SETQ EXTENT (WINDOWREGION (SETQ ATW (fetch (RESHAPINGWINDOWDATA ATTACHEDW) of ATWINFO]
               (SETQ ATWHGHT (fetch (REGION HEIGHT) of EXTENT))
               (SETQ ATWDTH (fetch (REGION WIDTH) of EXTENT))
               (SETQ ATWHERE (SELECTQ (fetch (RESHAPINGWINDOWDATA ATWHEREONEDGE) of ATWINFO)
                                      (JUSTIFY 'JUSTIFY)
                                      (CENTER 0)
                                      ((LEFT BOTTOM)
                                        -1)
                                      1))
               (SHAPEW ATW (SELECTQ (fetch (RESHAPINGWINDOWDATA ATEDGE) of ATWINFO)
                                    (TOP                              (* use new height)
                                         (SETQ ATWHGHT (fetch (RESHAPINGWINDOWDATA ATYSIZE) of ATWINFO))
                                         (SELECTQ ATWHERE
                                              (JUSTIFY (PROG1 (CREATEREGION LT (ADD1 (SETQ TEMP
                                                                                           (MAX TL TC TR)))
                                                                                     (ADD1 (DIFFERENCE RT LT))
                                                                                     ATWHGHT)
                                                              [SETQ TL (SETQ TC (SETQ TR (PLUS TEMP ATWHGHT]))
                                                       (-1 (PROG1 (CREATEREGION LT (ADD1 TL)
                                                                                ATWDTH ATWHGHT)
                                                                  (SETQ TL (PLUS TL ATWHGHT))))
                                                       (0 (PROG1 (CREATEREGION (CENTERINWIDTH ATWDTH MAINWEXTENT)
                                                                               (ADD1 TC)
                                                                               ATWDTH ATWHGHT)
                                                                 (SETQ TC (PLUS TC ATWHGHT))))
                                                       (1 (PROG1 (CREATEREGION (ADD1 (DIFFERENCE RT ATWDTH))
                                                                               (ADD1 TR)
                                                                               ATWDTH ATWHGHT)
                                                                 (SETQ TR (PLUS TR ATWHGHT))))
                                                       (SHOULDNT)))
                                    (RIGHT (SETQ ATWDTH (fetch (RESHAPINGWINDOWDATA ATXSIZE) of ATWINFO))
                                           (SELECTQ ATWHERE
                                                (JUSTIFY (PROG1 (CREATEREGION (ADD1 (SETQ TEMP (MAX RT RC RB)))
                                                                              BR ATWDTH (ADD1 (DIFFERENCE TR BR)))
                                                                [SETQ RT (SETQ RC (SETQ RB (PLUS TEMP ATWDTH]))
                                                         (1 (PROG1 (CREATEREGION (ADD1 RT)
                                                                                 (ADD1 (DIFFERENCE TR ATWHGHT))
                                                                                 ATWDTH ATWHGHT)
                                                                   (SETQ RT (PLUS RT ATWDTH))))
                                                         (0 (PROG1 (CREATEREGION (ADD1 RC)
                                                                                 (CENTERINHEIGHT ATWHGHT MAINWEXTENT)
                                                                                 ATWDTH ATWHGHT)
                                                                   (SETQ RC (PLUS RC ATWDTH))))
                                                         (-1 (PROG1 (CREATEREGION (ADD1 RB)
                                                                                  BR ATWDTH ATWHGHT)
                                                                    (SETQ RB (PLUS RB ATWDTH))))
                                                         (SHOULDNT)))
                                    (LEFT (SETQ ATWDTH (fetch (RESHAPINGWINDOWDATA ATXSIZE) of ATWINFO))
                                          (SELECTQ ATWHERE
                                               (JUSTIFY (CREATEREGION [SETQ LT (SETQ LC
                                                                                     (SETQ LB
                                                                                      (DIFFERENCE (MIN LT LC LB)
                                                                                                  ATWDTH]
                                                                                BL ATWDTH (ADD1 (DIFFERENCE TL BL))))
                                                        (1 (CREATEREGION (SETQ LT (DIFFERENCE LT ATWDTH))
                                                                         (ADD1 (DIFFERENCE TL ATWHGHT))
                                                                         ATWDTH ATWHGHT))
                                                        (0 (CREATEREGION (SETQ LC (DIFFERENCE LC ATWDTH))
                                                                         (CENTERINHEIGHT ATWHGHT MAINWEXTENT)
                                                                         ATWDTH ATWHGHT))
                                                        (-1 (CREATEREGION (SETQ LB (DIFFERENCE LB ATWDTH))
                                                                          BL ATWDTH ATWHGHT))
                                                        (SHOULDNT)))
                                    (BOTTOM (SETQ ATWHGHT (fetch (RESHAPINGWINDOWDATA ATYSIZE) of ATWINFO))
                                            (SELECTQ ATWHERE
                                                 (JUSTIFY (CREATEREGION LB [SETQ BL (SETQ BC
                                                                                      (SETQ BR
                                                                                       (DIFFERENCE (MIN BL BC BR)
                                                                                                   ATWHGHT]
                                                                                   (ADD1 (DIFFERENCE RB LB))
```

```
                                                    ATWHGHT))
                                          (-1 (CREATEREGION LB (SETQ BL (DIFFERENCE BL ATWHGHT))
                                                    ATWDTH ATWHGHT))
                                          (0 (CREATEREGION (CENTERINWIDTH ATWDTH MAINWEXTENT)
                                                    (SETQ BC (DIFFERENCE BC ATWHGHT))
                                                    ATWDTH ATWHGHT))
                                          (1 (CREATEREGION (ADD1 (DIFFERENCE RB ATWDTH))
                                                    (SETQ BR (DIFFERENCE BR ATWHGHT))
                                                    ATWDTH ATWHGHT))
                                          (SHOULDNT)))
                                (SHOULDNT])
```

(\SETGROUPMIN

```
[LAMBDA (GROUPATWINFO)                                  (* rrb "14-Dec-83 12:23")
                                                        (* sets the minimum of a group of attached windows.)
                                                        (* the CAR is the list of information structures of the members of
```

the group.)

(* set the size of the whole group so that the proportional calculation can go through.)

(* also sets the maximum in the dimension in which the group can expand if everyone in the group has a limit.
This information is used to determine allocation shares in the case where the group has its maximum size, no more space
will be given to it.)

```
(PROG [(GROUPSIZE (\ATWGROUPSIZE (CAR GROUPATWINFO]
      (replace (RESHAPINGWINDOWDATA ATNOWX) of GROUPATWINFO with (CAR GROUPSIZE))
      (replace (RESHAPINGWINDOWDATA ATNOWY) of GROUPATWINFO with (CDR GROUPSIZE)))
(SELECTQ (fetch (RESHAPINGWINDOWDATA ATEDGE) of GROUPATWINFO)
    ((LEFT RIGHT)
        (replace (RESHAPINGWINDOWDATA ATMINX) of GROUPATWINFO with (\MAXWIDTHOFGROUP (CAR GROUPATWINFO)))
        (replace (RESHAPINGWINDOWDATA ATMINY) of GROUPATWINFO with (for ATW in (CAR GROUPATWINFO)
                                                                      largest (fetch (RESHAPINGWINDOWDATA ATMINY
                                                                                                          )
                                                                                     of ATW)
                                                                      finally (RETURN $$EXTREME)))
        [replace (RESHAPINGWINDOWDATA ATMAXX) of GROUPATWINFO
           with (PROG ((TMAX 0)
                       (CMAX 0)
                       (BMAX 0)
                       THISMAX)
                      (RETURN (for ATW in (CAR GROUPATWINFO)
                                 do [COND
                                        ((NULL (SETQ THISMAX (fetch (RESHAPINGWINDOWDATA ATMAXX) of ATW)))
```

(* if any of the windows in the group doesn't have a max, the group doesn't either.)

```
                                         (RETURN NIL))
                                        (T (SELECTQ (fetch (RESHAPINGWINDOWDATA ATWHEREONEDGE) of ATW)
                                                (TOP (SETQ TMAX (IPLUS TMAX THISMAX)))
                                                (CENTER (SETQ CMAX (IPLUS CMAX THISMAX)))
                                                (SETQ BMAX (IPLUS BMAX THISMAX]
                                      finally (RETURN (IMAX TMAX CMAX BMAX])
    ((TOP BOTTOM)
        (replace (RESHAPINGWINDOWDATA ATMINX) of GROUPATWINFO with (for ATW in (CAR GROUPATWINFO)
                                                                      largest (fetch (RESHAPINGWINDOWDATA ATMINX
                                                                                                          )
                                                                                     of ATW)
                                                                      finally (RETURN $$EXTREME)))
        (replace (RESHAPINGWINDOWDATA ATMINY) of GROUPATWINFO with (\MAXHEIGHTOFGROUP (CAR GROUPATWINFO)))
        [replace (RESHAPINGWINDOWDATA ATMAXY) of GROUPATWINFO
           with (PROG ((LMAX 0)
                       (CMAX 0)
                       (RMAX 0)
                       THISMAX)
                      (RETURN (for ATW in (CAR GROUPATWINFO)
                                 do [COND
                                        ((NULL (SETQ THISMAX (fetch (RESHAPINGWINDOWDATA ATMAXY) of ATW)))
```

(* if any of the windows in the group doesn't have a max, the group doesn't either.)

```
                                         (RETURN NIL))
                                        (T (SELECTQ (fetch (RESHAPINGWINDOWDATA ATWHEREONEDGE) of ATW)
                                                (LEFT (SETQ LMAX (IPLUS LMAX THISMAX)))
                                                (CENTER (SETQ CMAX (IPLUS CMAX THISMAX)))
                                                (SETQ RMAX (IPLUS RMAX THISMAX]
                                      finally (RETURN (IMAX LMAX CMAX RMAX])
    (SHOULDNT])
```

(\SETWINFOXSIZE

```
[LAMBDA (WINFO PROPOSEDSIZE)                            (* bvm%: "10-Nov-84 15:14")
```

(* sets the X size of a window information structure, limiting by the maximum and returns the value put in.)

```
(replace (RESHAPINGWINDOWDATA ATXSIZE) of WINFO with (\LIMITBYMIN (\LIMITBYMAX PROPOSEDSIZE (fetch (
                                                                                    RESHAPINGWINDOWDATA
```

```
                                                                    ATMAXX)
                                                                of WINFO))
                                                    (fetch (RESHAPINGWINDOWDATA ATMINX) of WINFO])
```

(\SETWINFOYSIZE

```
  [LAMBDA (WINFO PROPOSEDSIZE)                                  (* bvm%: "10-Nov-84 15:17")
```

(* sets the Y size of a window information structure, limiting by the maximum and returns the value put in.)

(* bvm%: Used to say (IMAX this 0)%, but that is asymmetric with \SETWINFOXSIZE and the \LIMITBYMIN should catch it anyway)

```
    (replace (RESHAPINGWINDOWDATA ATYSIZE) of WINFO with (\LIMITBYMIN (\LIMITBYMAX PROPOSEDSIZE (fetch (
                                                                    RESHAPINGWINDOWDATA
                                                                    ATMAXY)
                                                                of WINFO))
                                                    (fetch (RESHAPINGWINDOWDATA ATMINY) of WINFO])
```

(\SHAREOFXTRAX

```
  [LAMBDA (WINFO TOTALNEWSIZE TOTALOLDSIZE)                     (* bvm%: "10-Nov-84 15:14")
```

(* returns the proportion of space in X that a window should get base on its size before the reshape.)

```
    (IMAX (IQUOTIENT (ITIMES (fetch (RESHAPINGWINDOWDATA ATNOWX) of WINFO)
                             TOTALNEWSIZE)
                     TOTALOLDSIZE)
          (fetch (RESHAPINGWINDOWDATA ATXSIZE) of WINFO])
```

(\SHAREOFXTRAY

```
  [LAMBDA (WINFO TOTALNEWSIZE TOTALOLDSIZE)                     (* rrb " 7-Jan-86 17:04")
```

(* returns the proportion of space in Y that a window should get based on its size before the reshape.)

```
    (COND
       ((EQ TOTALOLDSIZE 0)
        0)
       (T (IMAX (IQUOTIENT (ITIMES (fetch (RESHAPINGWINDOWDATA ATNOWY) of WINFO)
                                   TOTALNEWSIZE)
                           TOTALOLDSIZE)
                (fetch (RESHAPINGWINDOWDATA ATYSIZE) of WINFO])

)

(DEFINEQ
```

(ATTACHMENU

```
  [LAMBDA (MENU MAINWINDOW EDGE POSITIONONEDGE NOOPENFLG)       (* rrb "27-Jun-84 11:19")
                                                                (* this function associates a menu with a window.)
    (PROG (MENUWINDOW)
```

(* VERTFLG is non-NIL if the menu is to be layed out above or below the main window.)

```
          [SETQ MENUWINDOW (MENUWINDOW MENU (FMEMB EDGE '(LEFT RIGHT]
          (ATTACHWINDOW MENUWINDOW MAINWINDOW EDGE POSITIONONEDGE T)
          (OR NOOPENFLG (NOT (OPENWP MAINWINDOW))
              (OPENW MENUWINDOW))
          (RETURN MENUWINDOW])
```

(CREATEMENUEDWINDOW

```
  [LAMBDA (MENU WINDOWTITLE LOCATION WINDOWSPEC)                (* bvm%: "12-Apr-84 16:59")
```

(* This function is used to create a MAIN window MENU pair. MENU specifies the menu content and may be a menu, a list of items. WINDOWTITLE is a string specifying a title for the main window.
LOCATION specifies the placement of the window (TOP BOTTOM LEFT RIGHT);
WINDOWSPEC is a REGION. If it is NIL, a new window will be created.)

```
    (PROG ((VERTFLG (COND
                       ((NULL LOCATION)                         (* Default LOCATION is TOP)
                        (SETQ LOCATION 'TOP)
                        NIL)
                       ((FMEMB LOCATION '(LEFT RIGHT))
                        T)))
           WINDOW MENUW MENUWIDTH MENUHEIGHT WHOLEREGION MINTOTALHEIGHT MINTOTALWIDTH)
          (COND
             [(LISTP MENU)
              (SETQ MENU (create MENU
                                 ITEMS _ MENU
                                 CENTERFLG _ T
                                 TITLE _ (COND
                                            ((AND WINDOWTITLE VERTFLG)
```

(* If the menu is on the side continue the title bar even if the menu has no title)

```
                                        " "]
            ((type? MENU MENU))
            (T (\ILLEGAL.ARG MENU)))
         [COND
            ((NULL (fetch MENUROWS of MENU))
             (replace MENUROWS of MENU with (COND
                                              (VERTFLG (LENGTH (fetch (MENU ITEMS) of MENU)))
                                              (T 1]
         (SETQ MENUW (MENUWINDOW MENU VERTFLG))
         (SETQ MINTOTALWIDTH (SETQ MENUWIDTH (fetch (MENU IMAGEWIDTH) of MENU)))
         (SETQ MINTOTALHEIGHT (SETQ MENUHEIGHT (fetch (MENU IMAGEHEIGHT) of MENU)))
         (SELECTQ LOCATION
             ((TOP BOTTOM)
                 (add MINTOTALHEIGHT (FONTPROP (DEFAULTFONT 'DISPLAY)
                                               'HEIGHT)
                      (COND
                         (WINDOWTITLE (FONTPROP WindowTitleDisplayStream 'HEIGHT))
                         (T 0))))
             ((LEFT RIGHT)
                 (add MINTOTALWIDTH (TIMES 2 WBorder)))
             NIL)

         (* The window may be specified by the user. A region or an existing window may be supplied by the caller.
         In any case the size may have to be adjusted so that titles and and menu fit)

         [SETQ WHOLEREGION (COND
                              ((NULL WINDOWSPEC)
                               (PROMPTPRINT "Specify a region for " (OR WINDOWTITLE "the window"))
                               (PROG1 (GETREGION MINTOTALWIDTH MINTOTALHEIGHT)
                                      (CLRPROMPT)))
                              [(REGIONP WINDOWSPEC)
                               (create REGION using WINDOWSPEC WIDTH _ (IMAX MINTOTALWIDTH (fetch (REGION WIDTH)
                                                                                                   of WINDOWSPEC))
                                                                  HEIGHT _ (IMAX MINTOTALHEIGHT (fetch (REGION HEIGHT)
                                                                                                       of WINDOWSPEC]
                              (T (\ILLEGAL.ARG WINDOWSPEC]      (* Now set up the menu)
         (SELECTQ LOCATION
             ((TOP BOTTOM)                                     (* Shrink height of region by menu-occupied space to get main
                                                                  window region)
                 (COND
                    ((EQ LOCATION 'BOTTOM)
                     (add (fetch (REGION BOTTOM) of WHOLEREGION)
                          MENUHEIGHT)))
                 (replace (REGION HEIGHT) of WHOLEREGION with (IDIFFERENCE (fetch (REGION HEIGHT) of WHOLEREGION)
                                                                           MENUHEIGHT)))
             ((LEFT RIGHT)
                 (COND
                    ((EQ LOCATION 'LEFT)
                     (add (fetch (REGION LEFT) of WHOLEREGION)
                          MENUWIDTH)))
                 (replace (REGION WIDTH) of WHOLEREGION with (IDIFFERENCE (fetch (REGION WIDTH) of WHOLEREGION)
                                                                          MENUWIDTH)))
             NIL)
         [ATTACHWINDOW MENUW (SETQ WINDOW (CREATEW WHOLEREGION WINDOWTITLE))
                LOCATION
                (COND
                   (VERTFLG 'TOP)
                   (T 'JUSTIFY]
         (OPENW WINDOW)
         (OPENW MENUW)
         (RETURN WINDOW])
```

(MENUWINDOW

```
  [LAMBDA (MENU VERTFLG)                                       (* rrb "27-Jun-84 10:37")

         (* this function creates a window that has menu in it. The window has appropriate reshape, minsize and maxsize functions.)

    (PROG (WINDOW)
         [COND
            ((LISTP MENU)                                      (* assume its an item list)
             (SETQ MENU (create MENU
                                ITEMS _ MENU
                                CENTERFLG _ T]
         (COND
            [(type? MENU MENU)                                 (* check to make sure the number of rows and columns are set
                                                                  up.)
             (COND
                ((fetch MENUROWS of MENU))
                ((fetch MENUCOLUMNS of MENU))
                (VERTFLG (replace (MENU MENUCOLUMNS) of MENU with 1))
                (T (replace (MENU MENUROWS) of MENU with 1]
            (T (ERROR "arg not MENU" MENU)))                   (* update the menu image in case any of its fields were changed
     above.)
         (COND
            ((NOT (NUMBERP (fetch (MENU MENUOUTLINESIZE) of MENU)))
             (replace (MENU MENUOUTLINESIZE) of MENU with 0)))
```

```
        (UPDATE/MENU/IMAGE MENU)                                    (* Now build the menu window)
        (SETQ WINDOW (ADDMENU MENU (CREATEW (CREATEREGION 0 0 (WIDTHIFWINDOW (fetch (MENU IMAGEWIDTH)
                                                                                  of MENU)
                                                                              1)
                                                                    (HEIGHTIFWINDOW (fetch (MENU IMAGEHEIGHT) of MENU)
                                                                          NIL 1))
                                                             NIL 1 T)
                                                  NIL T))
        (WINDOWPROP WINDOW 'MINSIZE (FUNCTION MENUWMINSIZEFN))
        (WINDOWPROP WINDOW 'MAXSIZE (FUNCTION MENUWMINSIZEFN))
        (WINDOWADDPROP WINDOW 'RESHAPEFN (FUNCTION MENUWRESHAPEFN))
        (RETURN WINDOW])
```

(**MENUWMINSIZEFN**

```
  [LAMBDA (MENUW)                                                   ; Edited 14-Jan-99 17:16 by rmk:
    ;; returns the minimum size of a menu window.
    (PROG ([MENU (CAR (WINDOWPROP MENUW 'MENU]
           (TITLE? (WINDOWPROP MENUW 'TITLE))
           TITLERELATEDVAR BORDERSIZE OUTLINESIZE MINWIDTH)
          (SETQ BORDERSIZE (ITIMES (fetch (MENU MENUBORDERSIZE) of MENU)
                                   2))
          (SETQ OUTLINESIZE (ITIMES (IPLUS (fetch (MENU MENUOUTLINESIZE) of MENU)
                                           (WINDOWPROP MENUW 'BORDER))
                                    2))
          (SETQ MINWIDTH (ITIMES (IPLUS (MAXMENUITEMWIDTH MENU)
                                        BORDERSIZE 2)
                                 (fetch (MENU MENUCOLUMNS) of MENU)))
                                                                    ; The minimum width of the window takes into account the
                                                                    ; contents of the menu and its title
          [COND
             ((SETQ TITLERELATEDVAR (fetch (MENU TITLE) of MENU))
              (SETQ MINWIDTH (IMAX MINWIDTH (STRINGWIDTH TITLERELATEDVAR (SETQ TITLERELATEDVAR (MENUTITLEFONT
                                                                                                 MENU]
          (RETURN (CONS (WIDTHIFWINDOW MINWIDTH (WINDOWPROP MENUW 'BORDER))
                        (HEIGHTIFWINDOW (IPLUS (ITIMES (fetch (MENU MENUROWS) of MENU)
                                                       (IPLUS BORDERSIZE (MAXMENUITEMHEIGHT MENU)))
                                               (COND
                                                  (TITLERELATEDVAR (FONTPROP TITLERELATEDVAR 'HEIGHT))
                                                  (T 0)))
                                        TITLE?
                                        (WINDOWPROP MENUW 'BORDER])
```

(**MENUWRESHAPEFN**

```
  [LAMBDA (WINDOW OLDIMAGE OLDREGION)                               (* hdj " 6-Feb-85 15:50")
                                                                    (* This function takes care of size adjustments whenever the
                                                                    main window is reshaped.)
    (PROG ([MENU (CAR (WINDOWPROP WINDOW 'MENU]
           INTREGION USABLEWIDTH USABLEHEIGHT NROWS NCOLUMNS XTRWIDTH XTRHEIGHT BORDER)
          (OR MENU (RETURN))
          (DELETEMENU MENU NIL WINDOW)
          (SETQ BORDER (ITIMES 2 (fetch (MENU MENUOUTLINESIZE) of MENU)))
          (SETQ USABLEWIDTH (IDIFFERENCE (fetch (REGION WIDTH) of (SETQ INTREGION (DSPCLIPPINGREGION NIL WINDOW))
                                                                  )
                                         BORDER))
          [SETQ USABLEHEIGHT (IDIFFERENCE (fetch (REGION HEIGHT) of INTREGION)
                                          (COND
                                             ((fetch (MENU TITLE) of MENU)
                                              (IPLUS (FONTPROP (MENUTITLEFONT MENU)
                                                               'HEIGHT)
                                                     BORDER))
                                             (T BORDER]             (* calculate the largest item size that fits and the amount left
                                                                    over.)
          (SETQ XTRWIDTH (IDIFFERENCE USABLEWIDTH (ITIMES [replace ITEMWIDTH of MENU
                                                             with (IQUOTIENT USABLEWIDTH (SETQ NCOLUMNS
                                                                                               (fetch MENUCOLUMNS
                                                                                                  of MENU]
                                                                                         NCOLUMNS)))
          (SETQ XTRHEIGHT (IDIFFERENCE USABLEHEIGHT (ITIMES [replace ITEMHEIGHT of MENU
                                                               with (IQUOTIENT USABLEHEIGHT
                                                                               (SETQ NROWS (fetch MENUROWS
                                                                                              of MENU]
                                                                                       NROWS)))
          (UPDATE/MENU/IMAGE MENU)

          (* black out the window so the extra part of the window will not stand out.)

          (DSPFILL NIL BLACKSHADE 'REPLACE WINDOW)                  (* put the menu image centered in the window)
          (ADDMENU MENU WINDOW (create POSITION
                                       XCOORD _ (IQUOTIENT XTRWIDTH 2)
                                       YCOORD _ (IQUOTIENT XTRHEIGHT 2)))
          (SHOWSHADEDITEMS MENU WINDOW)
          (RETURN WINDOW])

)
```

```
(DEFINEQ

(GETPROMPTWINDOW
  [LAMBDA (MAINWINDOW %#LINES FONT DONTCREATE)                    ; Edited 22-Jan-88 15:20 by woz

    ;; makes sure that MAINWINDOW has an attached promptwindow and returns it.  If one already exists, it is shaped to be at least #LINES high.  If
    ;; FONT is NIL, the font of the main window is used for the promptwindow.

    (PROG ((PWINDOWPROP (WINDOWPROP MAINWINDOW 'PROMPTWINDOW))
           PWINDOW HEIGHT PAGEFULLFN)
          [COND
             (DONTCREATE (RETURN (CAR PWINDOWPROP]
          [SETQ FONT (COND
                        (FONT (FONTCREATE FONT))
                        (T (DSPFONT NIL (OR (CAR PWINDOWPROP)
                                            MAINWINDOW]
          (COND
             [%#LINES (COND
                         ((EQ %#LINES T)                          ; Infinitely expandable window
                          (SETQ PAGEFULLFN (FUNCTION \PROMPTWINDOW.PAGEFULLFN))
                          (SETQ %#LINES 1))
                         [(STRINGP %#LINES)                       ; Big enough for this string
                          (LET ([MAINWIDTH (fetch (REGION WIDTH) of (OR (CAR PWINDOWPROP)
                                                                        (WINDOWREGION MAINWINDOW]
                                (STRWIDTH (STRINGWIDTH %#LINES FONT)))
                               (SETQ %#LINES (IQUOTIENT (IPLUS STRWIDTH (SUB1 MAINWIDTH))
                                                        MAINWIDTH]
                         ((FIXP %#LINES))
                         (T (\ILLEGAL.ARG %#LINES]
             (T (SETQ %#LINES 1)))
          (COND
             [PWINDOWPROP (SETQ PWINDOW (CAR PWINDOWPROP))
                     (COND
                        ((NOT (OPENWP PWINDOW))
                         (REATTACHPROMPTWINDOW MAINWINDOW PWINDOW)))
                     (COND
                        ((IGREATERP %#LINES (CDR PWINDOWPROP))    ; Window exists, but not big enough
                         (\PROMPTWINDOW.EXPAND PWINDOWPROP %#LINES]
             (T (SETQ PWINDOW (CREATEW [create REGION
                                               LEFT _ 0
                                               BOTTOM _ 0
                                               WIDTH _ (fetch (REGION WIDTH) of (WINDOWREGION MAINWINDOW))
                                               HEIGHT _ (SETQ HEIGHT (HEIGHTIFWINDOW (TIMES %#LINES
                                                                                            (FONTPROP
                                                                                             FONT
                                                                                             'HEIGHT]
                                       NIL NIL T))
                (DSPSCROLL T PWINDOW)
                (DSPFONT FONT PWINDOW)
                (WINDOWPROP PWINDOW 'PAGEFULLFN 'NILL)
                (REATTACHPROMPTWINDOW MAINWINDOW PWINDOW)
                (WINDOWPROP MAINWINDOW 'PROMPTWINDOW (CONS PWINDOW %#LINES))
                (WINDOWPROP PWINDOW 'OPENFN (FUNCTION \PROMPTWINDOW.OPENFN))
                (WINDOWPROP PWINDOW 'PASSTOMAINCOMS '(CLOSEW BURYW REDISPLAYW MOVEW SHAPEW SHRINKW
                                                            HARDCOPYIMAGEW))
                (\PROMPTWINDOW.SET.HEIGHT PWINDOW HEIGHT)
                (OPENW PWINDOW)))
          (AND PAGEFULLFN (WINDOWPROP PWINDOW 'PAGEFULLFN PAGEFULLFN))
          (RETURN PWINDOW])

(\PROMPTWINDOW.EXPAND
  [LAMBDA (PWINDOWPROP %#LINES)                                   (* bvm%: " 2-May-86 14:59")

          (* * Expand the PWINDOWPROP = (window . nlines) to be %#LINES high)

    (LET* [(PWINDOW (CAR PWINDOWPROP))
           (HEIGHT (HEIGHTIFWINDOW (TIMES %#LINES (FONTPROP PWINDOW 'HEIGHT]
          (SHAPEW PWINDOW (create REGION using (WINDOWPROP PWINDOW 'REGION)
                                               HEIGHT _ HEIGHT))
          (RPLACD PWINDOWPROP %#LINES)
          (\PROMPTWINDOW.SET.HEIGHT PWINDOW HEIGHT])

(\PROMPTWINDOW.SET.HEIGHT
  [LAMBDA (PWINDOW HEIGHT)                                        (* bvm%: " 2-May-86 14:57")

          (* * Sets prompt window's height to be HEIGHT -- makes window inflexible and coerces it onto screen if it is off)

    (LET [(OBSCUREDHEIGHT (IDIFFERENCE SCREENHEIGHT (fetch (REGION TOP) of (WINDOWPROP PWINDOW 'REGION]
         [COND
            ((ILESSP OBSCUREDHEIGHT 0)                            (* Promptwindow off screen at top, so slip window group down
                                                                     to make it visible)
             (RELMOVEW (MAINWINDOW PWINDOW)
                    (create POSITION
                            XCOORD _ 0
                            YCOORD _ OBSCUREDHEIGHT]
```

```
         (WINDOWPROP PWINDOW 'MINSIZE (CONS 0 HEIGHT))
         (WINDOWPROP PWINDOW 'MAXSIZE (CONS 64000 HEIGHT])
```

(\PROMPTWINDOW.OPENFN

```
  [LAMBDA (WINDOW)                                          (* bvm%: "11-Nov-84 15:52")

          (* * Called when WINDOW is opened. WINDOW had been closed, and hence detached, from its main window, but perhaps
          somebody still had a handle on it and is now printing to it. Look for an open window whose promptwindow is this window.)

    (OR (WINDOWPROP WINDOW 'MAINWINDOW)
        (for MAINW in (OPENWINDOWS) bind PWINDOWPROP when (AND (SETQ PWINDOWPROP (WINDOWPROP MAINW 'PROMPTWINDOW))
                                                               (EQ (CAR PWINDOWPROP)
                                                                   WINDOW))
         do (RETURN (REATTACHPROMPTWINDOW MAINW WINDOW])
```

(\PROMPTWINDOW.PAGEFULLFN

```
  [LAMBDA (WINDOW)                                          (* bvm%: " 2-May-86 14:59")

          (* * Called to automatically expand a prompt window)

    (LET* ((PWINDOWPROP (WINDOWPROP (MAINWINDOW WINDOW)
                                    'PROMPTWINDOW))
           (%#LINES (CDR PWINDOWPROP)))
          (AND %#LINES (\PROMPTWINDOW.EXPAND PWINDOWPROP (ADD1 %#LINES])
```

(REATTACHPROMPTWINDOW

```
  [LAMBDA (MAINWINDOW PWINDOW)                              ; Edited  5-Sep-91 19:25 by jds
    ;; Reattach a prompt window th the main window; -preserve PASSTOMAINCOMS rather than nuking them.
    (LET [(OLDPASSTOMAINCOMS (WINDOWPROP PWINDOW 'PASSTOMAINCOMS]
         (ATTACHWINDOW PWINDOW MAINWINDOW 'TOP 'JUSTIFY)
         (WINDOWPROP PWINDOW 'PASSTOMAINCOMS OLDPASSTOMAINCOMS])
```

(REMOVEPROMPTWINDOW

```
  [LAMBDA (MAINWINDOW)                                      (* rrb "23-Oct-85 13:56")
    (PROG (PWINDOW)
      LP  [COND
             ((SETQ PWINDOW (WINDOWPROP MAINWINDOW 'PROMPTWINDOW NIL))
              (WINDOWDELPROP (SETQ PWINDOW (CAR PWINDOW))
                     'OPENFN
                     (FUNCTION \PROMPTWINDOW.OPENFN))
              (DETACHWINDOW PWINDOW)
              (RETURN (CLOSEW PWINDOW]
          (COND
             ((NEQ MAINWINDOW (SETQ MAINWINDOW (MAINWINDOW MAINWINDOW)))
              (GO LP])

)

(DECLARE%: DONTCOPY DOEVAL@COMPILE

(DECLARE%: EVAL@COMPILE

(RECORD RESHAPINGWINDOWDATA (ATTACHEDW (ATEDGE . ATWHEREONEDGE)
                                   (ATMINX . ATMINY)
                                   (ATMAXX . ATMAXY)
                                   (ATXSIZE . ATYSIZE)
                                   (ATNOWX . ATNOWY)))
)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS WindowMenu WindowTitleDisplayStream WBorder WindowMenuCommands)
)
)


(DEFGLOBALVAR *ATTACHED-WINDOW-COMMAND-SYNONYMS*
   (LIST (CONS '\INTERACTIVE.CLOSEW 'CLOSEW)
         (CONS 'HARDCOPYIMAGEW.TOPRINTER 'HARDCOPYIMAGEW)
         (CONS 'HARDCOPYIMAGEW.TOFILE 'HARDCOPYIMAGEW))
   "used by attachwindows to associate window command substitutes with their original name, eg
   \interactive.closew with closew.  Must be maintained as an alist, with each entry of the form (new-com .
   old-com).")

(PUTPROPS ATTACHEDWINDOW COPYRIGHT ("Venue & Xerox Corporation" 1983 1984 1985 1986 1987 1988 1990 1991 1992
                                    1995 1999))
```

FUNCTION INDEX

VARIABLE INDEX

RECORD INDEX