```
File created:      6-Dec-94 10:55:40  {DSK}<project>lots>dec94>work>lots>freemenu.;3

  changes to:     (FNS FM.SKIPNEXT \FM.EDIT-FINDNEXT \FM.EDIT-ITEM \FM.EDIT-PREPARETOEDIT \FM.EDIT-INSERT
                       \FM.NUMBER-SETUP \FM.EDIT-SETUP \FM.NUMBER-CHANGESTATE)

previous date:     7-Sep-94 14:02:37  {DSK}<project>lots>dec94>work>lots>freemenu.;1

  Read Table:     INTERLISP

     Package:     INTERLISP

      Format:     XCCS
```

;;
;; Copyright (c) 1986, 1987, 1988, 1990, 1993, 1994 by Venue & Xerox Corporation. All rights reserved.

```
(RPAQQ FREEMENUCOMS
       [(PROP (FILETYPE MAKEFILE-ENVIRONMENT)
              FREEMENU)
        (COMS                                                         ; USER INTERFACE FUNCTIONS
              (MACROS FM.GROUPPROP FM.MENUPROP FM.NWAYPROP)
              (OPTIMIZERS FM.ITEMPROP)
              (PROP ARGNAMES FM.ITEMPROP FM.GROUPPROP FM.MENUPROP FM.NWAYPROP)
              (MACROS \FM.INSUREFM \FM.INSUREWINDOW)
                                                                      ; RUN TIME TYPE CHECKERS
              (FNS FREEMENU                                           ; ACCESSING FUNCTIONS
                   FM.ITEMPROP FM.GETITEM FM.GETSTATE                 ; CHANGING FUNCTIONS
                   FM.HIGHLIGHTITEM FM.CHANGELABEL FM.CHANGESTATE FM.RESETSTATE FM.RESETMENU FM.RESETSHAPE
                   FM.RESETGROUPS                                     ; MISC FUNCTIONS
                   FM.REDISPLAYITEM FM.REDISPLAYMENU FM.SHADE FM.EDITP FM.EDITITEM FM.ENDEDIT FM.SKIPNEXT
                   FM.WHICHITEM FM.TOPGROUPID))
        (COMS                                                         ; CREATION OF FREEMENUS
              (DECLARE%: DONTCOPY (MACROS \FM.ITEMPROP \FM.GROUPPROP \FM.NWAYPROP \FM.MAKEGROUP \FM.TOPGROUPPROP
                                         \FM.DTOPGROUPID \FM.DGROUPPROP \FM.DTOPGROUPPROP))
              (FNS                                                    ; FORMATTING
                   \FM.FORMAT \FM.FORMATBYROW \FM.FORMATBYCOLUMN \FM.FORMATBYGRID \FM.FORMATEXPLICIT
                   \FM.LAYOUTROW \FM.LAYOUTCOLUMN \FM.LAYOUTGRID \FM.JUSTIFYITEMS \FM.JUSTIFYGROUPS
                   \FM.PUSHGROUP                                      ; ERROR CHECKING
                   \FM.CHECKDESCRIPTION \FM.CHECKPROPS                ; CREATING
                   \FM.CREATEITEM \FM.GETREGIONS \FM.GETBITMAPS \FM.MAKEBITMAP \FM.READUSERDATA \FM.MAKELINKS
                   \FM.COLLECTNWAYS \FM.SETATTACHPOINT \FM.CREATEW \FM.STARTEDIT)
              (INITVARS (\FM.GROUP-ID-COUNTER 0))
              (GLOBALVARS \FM.GROUP-ID-COUNTER)
              (DECLARE%: DONTCOPY (MACROS \FM.MAKE-GROUP-ID \FM.SETUPPROPS \FM.SETFORMATPROPS \FM.CHECKFORBOX
                                         \FM.UPDATEFORBOX \FM.UPDATEGRID \FM.ITEMWIDTH \FM.ITEMHEIGHT
                                         \FM.ATTACHPOINT))
              (DECLARE%: DONTCOPY
                     (CONSTANTS (\FM.FORMAT-TYPES '(ROW COLUMN TABLE EXPLICIT))
                            (\FM.DEFAULTFORMAT 'ROW)
                                                                      ; format keywords
                            (\FM.GROUPSPEC 'GROUP)
                            (\FM.PROPSPEC 'PROPS)
                                                                      ; key words in description
                            (\FM.HJUSTIFY-SPECS '(LEFT CENTER RIGHT))
                            (\FM.VJUSTIFY-SPECS '(TOP MIDDLE BOTTOM))
                                                                      ; item justification keywords
                            (\FM.BOXSPACE 1)
                                                                      ; default number of bits between label and box
                            (\FM.ROWSPACE 2)
                            (\FM.COLUMNSPACE 10)
                                                                      ; default number of bits between formatted rows and columns
                            (\FM.ITEM-TYPES '(MOMENTARY TOGGLE 3STATE NWAY STATE NUMBER EDIT EDITSTART DISPLAY))
                                                                      ; known freemenu item types
                            (\FM.DESCRIPTION-PROPS '(TYPE LABEL LEFT BOTTOM ID GROUPID STATE INITSTATE FONT
                                                          BITMAP REGION MAXREGION MESSAGE USERDATA LINKS
                                                          SYSDOWNFN SYSMOVEDFN SYSSELECTEDFN DOWNFN HELDFN
                                                          MOVEDFN SELECTEDFN))
                                                                      ; properties in item description that don't become USERDATA
                            ))
              (RECORDS FREEMENUITEM))
        (COMS                                                         ; FREEMENU WINDOWS
              (DECLARE%: DONTCOPY (MACROS \FM.TRANSPOSE))
              (FNS \FM.OPENFN \FM.REDISPLAYMENU \FM.RESHAPEFN \FM.UNSCROLLWINDOW \FM.RESETCLIPPINGREGION
                   \FM.FILLWINDOW \FM.INITCORNERSFN \FM.TRANSPOSEHORZ \FM.TRANSPOSEVERT \FM.UPDATEGROUPEXTENT
                   \FM.WINDOWEXTENT \FM.UPDATEWINDOWEXTENT))
        (COMS                                                         ; MOUSE FUNCTIONS
              (DECLARE%: DONTCOPY (MACROS \FM.ONITEM \FM.CHECKREGION))
              (FNS \FM.WINDOWENTRYFN \FM.BUTTONEVENTFN \FM.RIGHTBUTTONFN \FM.DOSELECTION \FM.MENUHANDLER))
        (COMS                                                         ; ITEM SUPPORT FUNCTIONS
              (DECLARE%: DONTCOPY (MACROS \FM.DISPLAYBITMAP \FM.COERCEITEMPTR))
              (FNS \FM.GETITEMPROP \FM.PUTITEMPROP \FM.CGETITEMPROP \FM.CPUTITEMPROP \FM.DISPLAYITEM
                   \FM.HIGHLIGHTITEM \FM.CHANGELABEL \FM.CHANGESTATE \FM.ENDEDIT \FM.INSUREVISIBLE \FM.CLEARITEM
                   ))
        (COMS                                                         ; MOMENTARY ITEM FUNCTIONS
              (FNS \FM.MOMENTARY-SETUP \FM.MOMENTARY-SELECTEDFN))
        (COMS                                                         ; TOGGLE ITEM FUNCTIONS
              (FNS \FM.TOGGLE-SETUP \FM.TOGGLE-DOWNFN \FM.TOGGLE-SELECTEDFN \FM.TOGGLE-CHANGESTATE))
        (COMS                                                    ; 3STATE ITEM FUNCTIONS
              (FNS \FM.3STATE-SETUP \FM.3STATE-SETUPOFFBITMAP \FM.3STATE-DOWNFN \FM.3STATE-SELECTEDFN
                   \FM.3STATE-CHANGESTATE))
        (COMS                                                    ; STATE ITEM FUNCTIONS
              (FNS \FM.STATE-SETUP \FM.STATE-SELECTEDFN \FM.STATE-CHANGESTATE))
        (COMS                                                    ; NWAY ITEM FUNCTIONS
              (FNS \FM.NWAY-SETUP \FM.NWAY-MESSAGE \FM.NWAY-DOWNFN \FM.NWAY-MOVEDFN \FM.NWAY-SELECTEDFN
                   \FM.NWAY-CHANGESTATE))
        (COMS                                                    ; NUMBER ITEM FUNCTIONS
              (FNS \FM.NUMBER-SETUP \FM.NUMBER-MESSAGE \FM.NUMBER-SELECTEDFN \FM.NUMBER-CHANGESTATE))
        (COMS                                                    ; TITLE ITEM FUNCTIONS
              (FNS \FM.DISPLAY-SETUP))
        (COMS                                                    ; EDITSTART ITEM FUNCTIONS
              (FNS \FM.EDITSTART-SETUP \FM.EDITSTART-MESSAGE \FM.EDITSTART-SELECTEDFN))
        (COMS                                                    ; EDIT ITEMS
              (DECLARE%: DONTCOPY
                     (CONSTANTS (\FM.EDIT-TIMEOUT 100000)
                            (\FM.EDIT-RIGHTENDSPACE 5)
                            (\FM.EDIT-BLOCKSIZE 50)
                            (\FM.EDIT-CONTROLCHARS '(9 10 12 13))
                            (\FM.EDIT-CONTROLCHARSECHO 255)
                            (\FM.EDIT-WORDDELIMCHARS '(32 123 125 91 93 60 62 47 92 46 44 59 42 40 41 45))
                                                                ; space { } [ ] < > / \ . , ; * ( ) ---
                            ))
              (VARS (\FM.EDIT-TTBL))
              (GLOBALVARS \FM.EDIT-TTBL)
              (MACROS \FM.EDIT-MAXWIDTH \FM.EDIT-SCROLLAMOUNT)
              (FNS \FM.EDIT-SETUP \FM.EDIT-MESSAGE \FM.EDIT-SETUPTTBL \FM.EDIT-ITEM \FM.EDIT-PREPARETOEDIT
                   \FM.EDIT-FINDNEXT \FM.EDIT-FINDFIRST \FM.EDIT-BACKUP \FM.EDIT-WORDDELETE \FM.EDIT-INSERT
                   \FM.EDIT-DELETE \FM.EDIT-GETPOINTERINFO \FM.EDIT-MOVECARET \FM.EDIT-STRDELETE
                   \FM.EDIT-STRINSERT \FM.EDIT-UPDATEAFTERDELETE))
        (DECLARE%: DONTEVAL@LOAD DOEVAL@COMPILE DONTCOPY COMPILERVARS (ADDVARS (NLAMA)
                                                                             (NLAML)
                                                                             (LAMA FM.ITEMPROP])
```

(PUTPROPS **FREEMENU FILETYPE** CL:COMPILE-FILE)

(PUTPROPS **FREEMENU MAKEFILE-ENVIRONMENT** (:READTABLE "INTERLISP" :PACKAGE "INTERLISP"))

;; USER INTERFACE FUNCTIONS

(DECLARE%: EVAL@COMPILE

(PUTPROPS **FM.GROUPPROP MACRO**

```
          [ARGS                                                (* access macro to group props of window.
                                                               args (WINDOW GROUP PROP {VALUE}))
                (COND
                   ((NULL (CDDR ARGS))
                    (ERROR "Too few arguments to FM.GROUPPROP:" (CONS 'FM.GROUPPROP ARGS)))
                   [(CDDDR ARGS)
                    `(LET [(GROUP (CDR (FASSOC ,(CADR ARGS)
                                              (WINDOWPROP (\FM.INSUREWINDOW ,(CAR ARGS))
                                                     'FM.GROUPS]
                          (PROG1 (LISTGET GROUP ,(CADDR ARGS))
                              (LISTPUT GROUP ,(CADDR ARGS)
                                     ,(CADDDR ARGS)))]
                   (T `(LISTGET [CDR (FASSOC ,(CADR ARGS)
                                            (WINDOWPROP (\FM.INSUREWINDOW ,(CAR ARGS))
                                                   'FM.GROUPS]
                             ,(CADDR ARGS])
```

```
(PUTPROPS FM.MENUPROP MACRO [ARGS                              (* access macro to TOP group props of window.
                                                               args (WINDOW PROP {VALUE}))
                                 (COND
                                    ((NULL (CDR ARGS))
                                     (ERROR "Too few arguments to FM.MENUPROP:" (CONS 'FM.MENUPROP ARGS)))
                                    [(CDDR ARGS)
                                     `(LET [(GROUP (CDAR (WINDOWPROP (\FM.INSUREWINDOW %, (CAR ARGS))
                                                                'FM.GROUPS]
                                           (PROG1 (LISTGET GROUP %, (CADR ARGS))
                                               (LISTPUT GROUP %, (CADR ARGS)
                                                      %,
                                                      (CADDR ARGS)))]
                                    (T `(LISTGET (CDAR (WINDOWPROP (\FM.INSUREWINDOW %, (CAR ARGS))
                                                              'FM.GROUPS))
                                               %,
                                               (CADR ARGS])
```

```
(PUTPROPS FM.NWAYPROP MACRO [ARGS                              (* access macro to nway props. args
                                                               (WINDOW COLLECTION PROP {VALUE}))
                                 (COND
                                    ((NULL (CDDR ARGS))
                                     (ERROR "Too few arguments to FM.NWAYPROP:" (CONS 'FM.NWAYPROP ARGS)))
                                    [(CDDDR ARGS)
                                     `(LET [(NWAY (CDR (ASSOC ,(CADR ARGS)
                                                               (WINDOWPROP (\FM.INSUREWINDOW
                                                                              ,(CAR ARGS))
                                                                       'FM.NWAYS]
                                              (PROG1 (LISTGET NWAY ,(CADDR ARGS))
                                                  (LISTPUT NWAY ,(CADDR ARGS)
                                                         ,(CADDDR ARGS))))]
                                        (T `(LISTGET [CDR (ASSOC ,(CADR ARGS)
                                                                (WINDOWPROP (\FM.INSUREWINDOW
                                                                             ,(CAR ARGS))
                                                                      'FM.NWAYS]
                                                  ,(CADDR ARGS])
)

(DEFOPTIMIZER FM.ITEMPROP (&REST ARGS)
                          [COND
                             ((NULL (CDR ARGS))
                              (ERROR "Too few arguments to FM.ITEMPROP:" (CONS 'FM.ITEMPROP ARGS)))
                             ((NEQ (CAADR ARGS)
                                   'QUOTE)
                              'IGNOREMACRO)
                             ((CDDR ARGS)
                              (\FM.CPUTITEMPROP `(\FM.INSUREFM ,(CAR ARGS))
                                     (CADR ARGS)
                                     (CADDR ARGS)))
                             (T (\FM.CGETITEMPROP `(\FM.INSUREFM ,(CAR ARGS))
                                       (CADR ARGS])

(PUTPROPS FM.ITEMPROP ARGNAMES (ITEM PROP {VALUE}))

(PUTPROPS FM.GROUPPROP ARGNAMES (WINDOW GROUP PROP {VALUE}))

(PUTPROPS FM.MENUPROP ARGNAMES (WINDOW PROP {VALUE}))

(PUTPROPS FM.NWAYPROP ARGNAMES (WINDOW COLLECTION PROP {VALUE}))

(DECLARE%: EVAL@COMPILE

(PUTPROPS \FM.INSUREFM MACRO [ARGS

          (* args (ITEM WINDOW)%. Insure ITEM is freemenuitem. If WINDOW is supplied, then try to coerce item if necessary.)

                                      (COND
                                         [(CDR ARGS)                    (* WINDOW ARGUMENT SUPPLIED)
                                          `(COND
                                               ((type? FREEMENUITEM %, (CAR ARGS))
                                                %,
                                                (CAR ARGS))
                                               (T (IF [AND (LISTP %, (CAR ARGS))
                                                           (EQ \FM.GROUPSPEC (CAR %, (CAR ARGS]
                                                      THEN (ERROR "Can't describe a local item from top level:" %,
                                                                  (CAR ARGS))
                                                    ELSE (\FM.COERCEITEMPTR %, (CAR ARGS)
                                                                %,
                                                                (CADR ARGS]
                                         (T                             (* NO WINDOW SUPPLIED%: JUST TYPE CHECK ITEM)
                                            `(COND
                                                 ((type? FREEMENUITEM %, (CAR ARGS))
                                                  %,
                                                  (CAR ARGS))
                                                 (T (ERROR "Arg must be FreeMenuItem" %, (CAR ARGS])

(PUTPROPS \FM.INSUREWINDOW MACRO [(WINDOW)
                                      (COND
                                         ((AND (WINDOWP WINDOW)
                                               (WINDOWPROP WINDOW 'FM.ITEMS))
                                          WINDOW)
                                         (T (ERROR "Arg must be FreeMenu Window" WINDOW])
)

;; RUN TIME TYPE CHECKERS

(DEFINEQ

(FREEMENU
  [LAMBDA (DESCRIPTION TITLE BACKGROUND BORDER)                 (* jow "17-Apr-86 19:32")

          (* Create a freemenu from a description. \FM.FORMAT is the recursive formatter.
          The defaults are passed to it here. It returns a list of groups, the first of which is the entire menu.
          Each group is a property list, the first item being the ID of the group, with group properties following.)

    (SETQ DESCRIPTION (COPY DESCRIPTION))                       (* leave users description untouched)
    (LET ((WINDOW (\FM.CREATEW (\FM.FORMAT DESCRIPTION \FM.DEFAULTFORMAT DEFAULTFONT 0 0 \FM.ROWSPACE
                                      \FM.COLUMNSPACE)
                         TITLE BACKGROUND BORDER)))             (* \FM.SETATTACHPOINT (LISTGET
                                                                (CDAR WINDOW) (QUOTE ITEMS))
                                                                (fetch (REGION WIDTH) of (LISTGET
                                                                (CDAR WINDOW) (QUOTE REGION)))
```

```
                                                                    (fetch (REGION HEIGHT) of (LISTGET
                                                                    (CDAR WINDOW) (QUOTE REGION))))
          (\FM.MAKELINKS WINDOW)
          (\FM.COLLECTNWAYS WINDOW)
          (FM.RESETMENU WINDOW)
          WINDOW])
```

(**FM.ITEMPROP**

```
  [LAMBDA ARGPTR                                                    (* jow " 4-Apr-86 14:57")
    (COND
       [(ILESSP ARGPTR 2)
        (ERROR "Too few arguments to FM.ITEMPROP" (LIST 'FM.ITEMPROP (ARG ARGPTR 1]
       ((NOT (type? FREEMENUITEM (ARG ARGPTR 1)))
        (ERROR "FM.ITEMPROP arg must be FreeMenuItem:" (ARG ARGPTR 1)))
       ((EQ ARGPTR 2)
        (\FM.GETITEMPROP (ARG ARGPTR 1)
               (ARG ARGPTR 2)))
       (T (\FM.PUTITEMPROP (ARG ARGPTR 1)
                 (ARG ARGPTR 2)
                 (ARG ARGPTR 3])
```

(**FM.GETITEM**

```
  [LAMBDA (ID GROUP WINDOW)                                         (* jow "19-Apr-86 22:45")

              (* find an item in WINDOW based on GROUP and ID which is an item id or label, If GROUP is NIL, search whole menu.)

    (\FM.INSUREWINDOW WINDOW)
    (LET [(ITEMS (if GROUP
                     then (\FM.GROUPPROP WINDOW GROUP 'ITEMS)
                   else (WINDOWPROP WINDOW 'FM.ITEMS]
         (for ITEM in ITEMS thereis (OR (EQ ID (\FM.ITEMPROP ITEM 'ID))
                                        (EQUAL ID (\FM.ITEMPROP ITEM 'LABEL])
```

(**FM.GETSTATE**

```
  [LAMBDA (WINDOW)                                                  (* jow "18-Jun-86 16:29")

              (* programmer interface%: goes through all items and nway collections, returns a prop list of id / current state for any state
              items in the menu. The current state is the value of the STATE field, or for edit items, the LABEL.
              Don't include in state list if STATE is NIL.)

    (\FM.INSUREWINDOW WINDOW)
    (LET ((STATELIST (LIST NIL)))
         [for NWAY in (WINDOWPROP WINDOW 'FM.NWAYS) do (if (LISTGET (CDR NWAY)
                                                                   'STATE)
                                                           then (LCONC STATELIST (LIST (CAR NWAY)
                                                                                       (LISTGET (CDR NWAY)
                                                                                              'STATE]
         (for ITEM in (WINDOWPROP WINDOW 'FM.ITEMS)
            do (SELECTQ (\FM.ITEMPROP ITEM 'TYPE)
                   ((TOGGLE 3STATE STATE NWAY NUMBER)
                        [if (\FM.ITEMPROP ITEM 'STATE)
                            then (LCONC STATELIST (LIST (OR (\FM.ITEMPROP ITEM 'ID)
                                                            (\FM.ITEMPROP ITEM 'LABEL))
                                                        (\FM.ITEMPROP ITEM 'STATE])
                   (EDIT [LCONC STATELIST (LIST (\FM.ITEMPROP ITEM 'ID)
                                                (\FM.ITEMPROP ITEM 'LABEL])
                   NIL))
         (CAR STATELIST])
```

(**FM.HIGHLIGHTITEM**

```
  [LAMBDA (ITEM WINDOW)                                             (* jow "26-Jun-86 15:05")

              (* this is the user interface function for highlighting. Type check and coerce item, then call the real function)

    (\FM.INSUREWINDOW WINDOW)
    (SETQ ITEM (\FM.INSUREFM ITEM WINDOW))
    (\FM.HIGHLIGHTITEM ITEM WINDOW])
```

(**FM.CHANGELABEL**

```
  [LAMBDA (ITEM NEWLABEL WINDOW UPDATEFLG)                          ; Edited 28-Dec-87 17:08 by woz
                                                                    ; user interface to change the label of an item, and redisplay as
                                                                    ; necessary.
    (\FM.INSUREWINDOW WINDOW)
    (SETQ ITEM (\FM.INSUREFM ITEM WINDOW))
    (LET [(OLDREGION (\FM.ITEMPROP ITEM 'REGION]
         (\FM.CHANGELABEL ITEM NEWLABEL)                            ; fill in background

         ;; now put item back into its current state.  This only applies to particular type items (nway, toggle, 3state), so do the changestate directly,
         ;; rather than call changestate

         (SELECTQ (\FM.ITEMPROP ITEM 'TYPE)
              ((NWAY TOGGLE)                                        ; remember each nway item is handled as an individual toggle
                   (\FM.TOGGLE-CHANGESTATE ITEM (\FM.ITEMPROP ITEM 'STATE)))
              (3STATE (\FM.3STATE-CHANGESTATE ITEM (\FM.ITEMPROP ITEM 'STATE)))
              NIL)
        (if (OR UPDATEFLG (\FM.ITEMPROP ITEM 'CHANGELABELUPDATE))
            then                                                        ; update groups
                 (\FM.UPDATEGROUPEXTENT (WINDOWPROP WINDOW 'FM.GROUPS))
                 (WINDOWPROP WINDOW 'EXTENT (\FM.WINDOWEXTENT WINDOW))
                 (\FM.REDISPLAYMENU WINDOW)
          else                                                          ; just redisplay item
               (\FM.CLEARITEM ITEM WINDOW OLDREGION)                    ; fill in background
               (\FM.DISPLAYBITMAP ITEM (\FM.ITEMPROP ITEM 'BITMAP)
                      WINDOW])
```

(**FM.CHANGESTATE**

```
  [LAMBDA (X NEWSTATE WINDOW)                                           ; Edited 28-Dec-87 17:09 by woz

    ;; user interface to change the state of any (state) item or nway collection.  Redisplay the item if the window is open

    (\FM.INSUREWINDOW WINDOW)
    (if (ASSOC X (WINDOWPROP WINDOW 'FM.NWAYS))
        then                                                            ; X specifies an NWAY.  Changestate and redisplay.
             (LET [(OLDSTATE (\FM.NWAYPROP WINDOW X 'STATE]
                  (if NEWSTATE
                      then                                              ; NIL would mean deselect
                           (SETQ NEWSTATE (\FM.INSUREFM NEWSTATE WINDOW)))
                  (\FM.CHANGESTATE X NEWSTATE WINDOW)
                  (if OLDSTATE
                      then (\FM.DISPLAYBITMAP OLDSTATE (\FM.ITEMPROP OLDSTATE 'BITMAP)
                                  WINDOW))
                  (if NEWSTATE
                      then (\FM.DISPLAYBITMAP NEWSTATE (\FM.ITEMPROP NEWSTATE 'BITMAP)
                                  WINDOW)))
      else                                                              ; treat X as an item
           (SETQ X (\FM.INSUREFM X WINDOW))
           (IF (FMEMB (\FM.ITEMPROP X 'TYPE)
                      '(EDIT NUMBER))
               THEN
                    ;; do this because the doc says changestate works with edit items.  maybe a dumb idea.  need to let the main changelabel
                    ;; routine take care of the display stuff.

                    (FM.CHANGELABEL X NEWSTATE WINDOW)
             ELSE (\FM.CHANGESTATE X NEWSTATE WINDOW)
                  (\FM.DISPLAYBITMAP X (\FM.ITEMPROP X 'BITMAP)
                         WINDOW])
```

(**FM.RESETSTATE**

```
  [LAMBDA (X WINDOW)                                                    (* jow "24-Apr-86 21:27")
                                                                        (* Reset X, an item or nway collection, to its initial state and
                                                                        redisplay)
    (\FM.INSUREWINDOW WINDOW)
    (LET [(INITSTATE (if (ASSOC X (WINDOWPROP WINDOW 'FM.NWAYS))
                         then (\FM.NWAYPROP WINDOW X 'INITSTATE)
                       else (\FM.ITEMPROP (\FM.INSUREFM X WINDOW)
                                   'INITSTATE]
         (FM.CHANGESTATE X INITSTATE WINDOW])
```

(**FM.RESETMENU**

```
  [LAMBDA (WINDOW)                                                      (* jow "26-Jun-86 14:43")
                                                                        (* reset each item to its INITSTATE)
    (\FM.INSUREWINDOW WINDOW)
    (\FM.ENDEDIT WINDOW T)
    (for ITEM in (WINDOWPROP WINDOW 'FM.ITEMS) do (FM.RESETSTATE ITEM WINDOW))
    (for NWAY in (WINDOWPROP WINDOW 'FM.NWAYS) do (FM.RESETSTATE (CAR NWAY)
                                                                  WINDOW))
    (\FM.REDISPLAYMENU WINDOW])
```

(**FM.RESETSHAPE**

```
  [LAMBDA (WINDOW ALWAYSFLG)                                            (* jow "19-Apr-86 22:50")

            (* programmer way of reshaping a freemenu window to its minimal extent.
            If window is too small, it will be reshaped, without moving the lower left corner.
            If window is too big, it will only be reshaped if ALWAYSFLG is T.)

    (\FM.INSUREWINDOW WINDOW)
    (if (OR (ILESSP (WINDOWPROP WINDOW 'WIDTH)
                    (WINDOWPROP WINDOW 'FM.MINWIDTH))
            (ILESSP (WINDOWPROP WINDOW 'HEIGHT)
                    (WINDOWPROP WINDOW 'FM.MINHEIGHT))
            ALWAYSFLG)
        then (SHAPEW WINDOW (CREATEREGION (fetch (REGION LEFT) of (WINDOWPROP WINDOW 'REGION))
                                   (fetch (REGION BOTTOM) of (WINDOWPROP WINDOW 'REGION))
                                   (WIDTHIFWINDOW (WINDOWPROP WINDOW 'FM.MINWIDTH)
                                          (WINDOWPROP WINDOW 'BORDER))
                                   (HEIGHTIFWINDOW (WINDOWPROP WINDOW 'FM.MINHEIGHT)
                                          (WINDOWPROP WINDOW 'TITLE)
                                          (WINDOWPROP WINDOW 'BORDER])
```

```
(FM.RESETGROUPS
  [LAMBDA (WINDOW GROUPLIST REDISPLAYFLG)                    (* jow "26-Jun-86 14:45")
                                                             (* user interface to recalculating group extents.)

    (\FM.INSUREWINDOW WINDOW)
    (\FM.UPDATEGROUPEXTENT (WINDOWPROP WINDOW 'FM.GROUPS)
           GROUPLIST)
    (AND REDISPLAYFLG (\FM.REDISPLAYMENU WINDOW])

(FM.REDISPLAYITEM
  [LAMBDA (ITEM WINDOW)                                      (* jow "26-Jun-86 14:51")
                                                             (* user interface to displaying an item.)

    (\FM.INSUREWINDOW WINDOW)
    (SETQ ITEM (\FM.INSUREFM ITEM WINDOW))
    (\FM.DISPLAYBITMAP ITEM (\FM.ITEMPROP ITEM 'BITMAP)
           WINDOW])

(FM.REDISPLAYMENU
  [LAMBDA (WINDOW)                                           (* jow "26-Jun-86 14:45")
                                                             (* use \FM.REDISPLAYMENU, which has hooks for updating a
                                                             particular region.)

    (\FM.INSUREWINDOW WINDOW)
    (\FM.REDISPLAYMENU WINDOW])

(FM.SHADE
  [LAMBDA (X SHADE WINDOW)                                   (* jow "26-Jun-86 14:59")

             (* X is a group id or an item. Paint shade on top of group or item.)

    (\FM.INSUREWINDOW WINDOW)
    (LET [(REGION (OR (\FM.GROUPPROP WINDOW X 'REGION)
                      (\FM.ITEMPROP (\FM.INSUREFM X WINDOW)
                             'REGION]
         (if (AND REGION (OPENWP WINDOW))
             then (BLTSHADE (TEXTUREP SHADE)
                         WINDOW NIL NIL NIL NIL 'PAINT REGION])

(FM.EDITP
  [LAMBDA (WINDOW)                                           (* jow "19-Apr-86 22:52")
    (WINDOWPROP (\FM.INSUREWINDOW WINDOW)
           'FM.EDITITEM])

(FM.EDITITEM
  [LAMBDA (ITEM WINDOW CLEARFLG)                             (* jow "20-Oct-86 10:48")

;;; start editing at beginning of item.

    (\FM.INSUREWINDOW WINDOW)
    (SETQ ITEM (\FM.INSUREFM ITEM WINDOW))
    (\FM.ENDEDIT WINDOW T)
    [if CLEARFLG
        then                                                 ; hack to get EDIT-ITEM to clear item first.
             (SETQ CLEARFLG '(RIGHT]
    (if (OPENWP WINDOW)
        then (ADD.PROCESS `(\FM.STARTEDIT ',ITEM ',WINDOW ',CLEARFLG)
                    'NAME
                    'FREEMENU
                    'FREEMENU.PROCESS T])

(FM.ENDEDIT
  [LAMBDA (WINDOW WAITFLG)                                   (* jow "24-Apr-86 21:23")
    (\FM.INSUREWINDOW WINDOW)
    (\FM.ENDEDIT WINDOW WAITFLG])

(FM.SKIPNEXT
  [LAMBDA (WINDOW CLEARFLG)                                  ; Edited  6-Dec-94 10:53 by jds
    ;; SKIP FORWARD to the next editable item, and start editing there.  If CLEARFLG, clear the new item's contents as we move there.
    ;; This function needs to parallel processing in \EDITSTART-SELECTEDFN, to get the DONEFN right.
    (COND
       [(FM.EDITP WINDOW)
        ;; eval the EDITITEM change in the FREEMENU process, which must be the tty process if editing.  This works even if called from the
        ;; FREEMENU process, eg by LIMITCHARS
        (LET ((FM.PROCESS (TTY.PROCESS)))
             (COND
                [(PROCESSPROP FM.PROCESS 'FREEMENU.PROCESS)
                 (PROCESS.EVAL FM.PROCESS `(LET ((NEWITEM (\FM.EDIT-FINDNEXT)))
                                                (if NEWITEM
                                                  then (SETQ EDITITEM NEWITEM)
                                                       (if ',CLEARFLG
                                                           then (FM.CHANGELABEL EDITITEM "" WINDOW))
                                                       (\FM.EDIT-PREPARETOEDIT EDITITEM T)
                                                       (\FM.INSUREVISIBLE EDITITEM WINDOW)
                                                       (SETQ DONEFN (IF (EQ (\FM.ITEMPROP EDITITEM 'TYPE)
                                                                             'NUMBER)
                                                                        THEN (FUNCTION \FM.NUMBER-CHANGESTATE)
                                                                        ))
                                                else (\FM.ENDEDIT WINDOW]
                 (T (ERROR "Can't find freemenu process to do skip-next" FM.PROCESS]
         (T                                                    ; not editing, so start with first edit item in menu
            (LET ((EDITITEM (\FM.EDIT-FINDFIRST WINDOW)))
                 (COND
                    (EDITITEM (FM.EDITITEM EDITITEM WINDOW CLEARFLG])
```

(FM.WHICHITEM

```
  [LAMBDA (WINDOW POSorX Y)                                    (* jow "19-Apr-86 22:54")

            (* user interface to CHECKREGION. Return the item in WINDOW at
            (POSorX, Y) If WINDOW is NIL, use the window the cursor is in, and the cursor position in that window)

    (if (OR (WINDOWP WINDOW)
            (SETQ WINDOW (WHICHW)))
        then (COND
                ((POSITIONP POSorX)
                 (\FM.CHECKREGION WINDOW (fetch (POSITION XCOORD) of POSorX)
                        (fetch (POSITION YCOORD) of POSorX)))
                (POSorX (\FM.CHECKREGION WINDOW POSorX Y))
                (T (\FM.CHECKREGION WINDOW (LASTMOUSEX WINDOW)
                          (LASTMOUSEY WINDOW])
```

(FM.TOPGROUPID

```
  [LAMBDA (WINDOW)                                             (* jow "19-Apr-86 22:54")
                                                               (* grab id of top group)
    (\FM.DTOPGROUPID (WINDOWPROP (\FM.INSUREWINDOW WINDOW)
                            'FM.GROUPS])
)

;; CREATION OF FREEMENUS

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS \FM.ITEMPROP MACRO [ARGS

            (* access macro to FREEMENUITEM datatype. args (ITEM PROP {VALUE}) Doesnt force an INSUREFM on the item, so
            intended for internal use only. PROP must be a quoted literal.)

                                   (COND
                                      ((NULL (CDR ARGS))
                                       (ERROR "Too few arguments to \FM.ITEMPROP:" (CONS 'FM.ITEMPROP ARGS)))
                                      ((NOT (EQ (CAADR ARGS)
                                                'QUOTE))
                                       (ERROR "CANT USE \FM.ITEMPROP UNLESS PROP IS QUOTED"))
                                      ((CDDR ARGS)
                                       (\FM.CPUTITEMPROP (CAR ARGS)
                                              (CADR ARGS)
                                              (CADDR ARGS)))
                                      (T (\FM.CGETITEMPROP (CAR ARGS)
                                                (CADR ARGS])

(PUTPROPS \FM.GROUPPROP MACRO [ARGS

            (* internal access macro to group props of window. doesn't check for illegal args.
            args (WINDOW GROUP PROP {VALUE}))

                                    (COND
                                       ((NULL (CDDR ARGS))
                                        (ERROR "Too few arguments to FM.GROUPPROP:" (CONS 'FM.GROUPPROP ARGS))
                                        )
                                       [(CDDDR ARGS)
                                        `(LET [(GROUP (CDR (FASSOC %, (CADR ARGS)
                                                                  (WINDOWPROP %, (CAR ARGS)
                                                                         'FM.GROUPS]
                                              (PROG1 (LISTGET GROUP %, (CADDR ARGS))
                                                  (LISTPUT GROUP %, (CADDR ARGS)
                                                         %,
                                                         (CADDDR ARGS)))]
                                       (T `(LISTGET [CDR (FASSOC %, (CADR ARGS)
                                                                (WINDOWPROP %, (CAR ARGS)
                                                                       'FM.GROUPS]
                                                  %,
                                                   (CADDR ARGS])

(PUTPROPS \FM.NWAYPROP MACRO [ARGS                                 (* internal access macro to nway props.
                                                                   doesn't error check args. args (WINDOW COLLECTION PROP
                                        {VALUE}))
                                        (COND
                                           ((NULL (CDDR ARGS))
                                            (ERROR "Too few arguments to FM.NWAYPROP:" (CONS 'FM.NWAYPROP ARGS)))
                                           [(CDDDR ARGS)
                                            `(LET [(NWAY (CDR (ASSOC %, (CADR ARGS)
                                                                     (WINDOWPROP %, (CAR ARGS)
                                                                            'FM.NWAYS]
                                                  (PROG1 (LISTGET NWAY %, (CADDR ARGS))
                                                      (LISTPUT NWAY %, (CADDR ARGS)
                                                             %,
                                                             (CADDDR ARGS)))]
                                           (T `(LISTGET [CDR (ASSOC %, (CADR ARGS)
                                                                    (WINDOWPROP %, (CAR ARGS)
                                                                           'FM.NWAYS]
                                                     %,
                                                     (CADDR ARGS])

(PUTPROPS \FM.MAKEGROUP MACRO [ARGS                                (* access macro that will build group from
                                                                   (ID PROPS))
                                         `(CONS %, (CAR ARGS)
                                                %,
                                                (CADR ARGS])

(PUTPROPS \FM.TOPGROUPPROP MACRO [ARGS                             (* access macro to top group of window.
                                                                   args (WINDOW PROP {VALUE}))
                                            (COND
                                               ((NULL (CDR ARGS))
                                                (ERROR "BAD ARGS TO \FM.TOPGROUPPROP:" (CONS '\FM.TOPGROUPPROP
                                                                                             ARGS)))
                                               [(CDDR ARGS)
                                                `(LET [(GROUP (CDAR (WINDOWPROP (\FM.INSUREWINDOW %, (CAR ARGS))
                                                                           'FM.GROUPS]
                                                      (PROG1 (LISTGET GROUP %, (CADR ARGS))
                                                          (LISTPUT GROUP %, (CADR ARGS)
                                                                 %,
                                                                 (CADDR ARGS)))]
                                               (T `(LISTGET (CDAR (WINDOWPROP (\FM.INSUREWINDOW %, (CAR ARGS))
                                                                         'FM.GROUPS))
                                                         %,
                                                         (CADR ARGS])

(PUTPROPS \FM.DTOPGROUPID MACRO ((GROUP)
                                 (CAAR GROUP)))

(PUTPROPS \FM.DGROUPPROP MACRO [ARGS                               (* access macro to groups props directly.
                                                                   args (GROUPS GROUPID PROP {VALUE}))
                                          (COND
                                             ((NULL (CDDR ARGS))
                                              (ERROR "BAD ARGS TO \FM.DGROUPPROP" (CONS '\FM.DGROUPPROP ARGS)))
                                             [(CDDDR ARGS)
                                              `(LET [(GROUP (CDR (FASSOC %, (CADR ARGS)
                                                                         %,
                                                                         (CAR ARGS]
                                                    (PROG1 (LISTGET GROUP %, (CADDR ARGS))
                                                        (LISTPUT GROUP %, (CADDR ARGS)
                                                               %,
                                                               (CADDDR ARGS)))]
                                             (T `(LISTGET (CDR (FASSOC %, (CADR ARGS)
                                                                      %,
                                                                      (CAR ARGS)))
                                                       %,
                                                       (CADDR ARGS])

(PUTPROPS \FM.DTOPGROUPPROP MACRO [ARGS                            (* access macro to direct top group.
                                                                   args (GROUPS PROP {VALUE}))
                                             (COND
                                                ((NULL (CDR ARGS))
                                                 (ERROR "BAD ARGS TO \FM.DTOPGROUPPROP:" (CONS '\FM.DTOPGROUPPROP
                                                                                               ARGS)))
                                                [(CDDR ARGS)
                                                 `(PROG1 (LISTGET (CDAR %, (CAR ARGS))
                                                                %,
                                                                (CADR ARGS))
                                                      (LISTPUT (CDAR %, (CAR ARGS))
                                                             %,
                                                             (CADR ARGS)
                                                             %,
                                                             (CADDR ARGS)))]
                                                (T `(LISTGET (CDAR %, (CAR ARGS))
                                                          %,
                                                          (CADR ARGS])
)
```

```
)

(DEFINEQ

(\FM.FORMAT
  [LAMBDA (DESCRIPTION FORMAT FONT LEFT BOTTOM ROWSPACE COLUMNSPACE MOTHER ID PROPS)
                                                                     (* jow "28-Oct-86 18:42")

            (* recursive formatter. MOTHER is this groups mother group id, and ID is this groups id, and PROPS is this groups property
            list. Currently ID and PROPS are unspecified arguments, and are only set by SETUPPROPS
            (but they are available as format arguments for later versions???) Format description based on its requested format in
            PROPS. If the format type is not known, treat it as a user specified funtion to do the desired formatting, and apply it to the
            description. (NOT CURRENTLY) LEFT and BOTTOM specify the corner of the groups coordinate system, and the LEFT
            and BOTTOM menuprops in the group specify offsets in that system.
            If the group is boxed, then offset the group before formatting, and readjust the extent after formatting to account for the box.
            Return a group structure for this group.)

    (\FM.SETUPPROPS DESCRIPTION '(FORMAT FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID))
    (LET (GROUPS OLDCORNER BOXOFFSET EXTENT)
         (\FM.CHECKFORBOX)
         (SETQ GROUPS (SELECTQ FORMAT
                           (ROW (\FM.FORMATBYROW DESCRIPTION FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID PROPS))
                           (COLUMN (\FM.FORMATBYCOLUMN DESCRIPTION FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID
                                          PROPS))
                           (TABLE (\FM.FORMATBYGRID DESCRIPTION FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID PROPS))
                           (EXPLICIT (\FM.FORMATEXPLICIT DESCRIPTION FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID
                                            PROPS))
                           NIL))

            (* Hook for user defined format types -
            APPLY* (LISTGET PROPS (QUOTE FORMAT)) DESCRIPTION FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID)

         (\FM.JUSTIFYITEMS GROUPS)
         (\FM.JUSTIFYGROUPS GROUPS)
         (SETQ EXTENT (\FM.DTOPGROUPPROP GROUPS 'REGION))          (* UPDATEFORBOX macro uses EXTENT)
         (\FM.UPDATEFORBOX)
         GROUPS])

(\FM.FORMATBYROW
  [LAMBDA (DESCRIPTION FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID PROPS)
                                                                     (* jow "17-Apr-86 18:09")

            (* Called when row formatting is specified. ID and PROPS are passed from \FM.FORMAT.
            ID is id of this group, and thus passed to MOTHER of each row group.
            PROPS is the group proplist to build the group from. DESCRIPTION is a list of rows, each row a list of item descriptions and
            groups. Reverse the rows, then build from bottom up. Use \FM.LAYOUTROW to lay out the items in a row.)

    (LET ((EXTENT (CREATEREGION LEFT BOTTOM 0 0))
          (ITEMLIST (LIST NIL))
          (GROUPLIST (LIST NIL))
          (ROWIDS (LIST NIL))
          (ROWBOTTOM BOTTOM)
          GROUPS)
         (for ROW in (DREVERSE DESCRIPTION) do (SETQ GROUPS (\FM.LAYOUTROW ROW FONT LEFT ROWBOTTOM ROWSPACE
                                                                  COLUMNSPACE ID))
                                               (TCONC ROWIDS (\FM.DTOPGROUPID GROUPS))
                                               [LCONC ITEMLIST (REVERSE (\FM.DTOPGROUPPROP GROUPS 'ITEMS]
                                               (EXTENDREGION EXTENT (\FM.DTOPGROUPPROP GROUPS 'REGION))
                                               (LCONC GROUPLIST GROUPS)
                                               (add ROWBOTTOM (fetch (REGION HEIGHT) of (\FM.DTOPGROUPPROP
                                                                                         GROUPS
                                                                                         'REGION))
                                                    ROWSPACE))
         (LISTPUT PROPS 'ITEMS (DREVERSE (CAR ITEMLIST)))
         (LISTPUT PROPS 'REGION EXTENT)
         (LISTPUT PROPS 'DAUGHTERS (DREVERSE (CAR ROWIDS)))
         (CONS (\FM.MAKEGROUP ID PROPS)
               (CAR GROUPLIST])

(\FM.FORMATBYCOLUMN
  [LAMBDA (DESCRIPTION FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID PROPS)
                                                                     ; Edited 29-Dec-87 14:45 by woz

    ;; ID and PROPS are passed from \FM.FORMAT.  ID is this groups id, and is passed as the MOTHER of each column.  DESCRIPTION is a list of
    ;; columns, each column a list of items (top to bottom) and groups.  \FM.LAYOUTCOLUMN takes a column description and lays out the items.
    ;; Column formatting requires a second pass, to top justify the columns.  This is done by going through the GROUPLIST and pushing up each
    ;; column as necessary.

    (LET ((EXTENT (CREATEREGION LEFT BOTTOM 0 0))
          (ITEMLIST (LIST NIL))
          (GROUPLIST (LIST NIL))
          (COLUMNIDS (LIST NIL))
          (COLUMNLEFT LEFT)
          GROUPS)
         (for COL in DESCRIPTION do (SETQ GROUPS (\FM.LAYOUTCOLUMN COL FONT COLUMNLEFT BOTTOM ROWSPACE
                                                         COLUMNSPACE ID))
```

```
                                   (TCONC COLUMNIDS (\FM.DTOPGROUPID GROUPS))
                                   [LCONC ITEMLIST (COPY (\FM.DTOPGROUPPROP GROUPS 'ITEMS]
                                   (EXTENDREGION EXTENT (\FM.DTOPGROUPPROP GROUPS 'REGION))
                                   (LCONC GROUPLIST GROUPS)
                                   (add COLUMNLEFT (fetch (REGION WIDTH) of (\FM.DTOPGROUPPROP GROUPS 'REGION))
                                        COLUMNSPACE))
          (SETQ GROUPLIST (CAR GROUPLIST))                         ; list from LCONC pair
          [LET ((HEIGHT (fetch (REGION HEIGHT) of EXTENT))
                COLHEIGHT)
               (for COLID in (CAR COLUMNIDS) do                    ; go through each column, pushing up each item in the column.
                                            [SETQ COLHEIGHT (fetch (REGION HEIGHT) of (\FM.DGROUPPROP
                                                                                       GROUPLIST COLID
                                                                                       'REGION]
                                            (if (NEQ COLHEIGHT HEIGHT)
                                                then                ; column doesn't reach top, so push up
                                                     (\FM.PUSHGROUP COLID GROUPLIST (IDIFFERENCE HEIGHT
                                                                                          COLHEIGHT)
                                                            'UP]
          (LISTPUT PROPS 'ITEMS (CAR ITEMLIST))
          (LISTPUT PROPS 'REGION EXTENT)
          (LISTPUT PROPS 'DAUGHTERS (CAR COLUMNIDS))
          (CONS (\FM.MAKEGROUP ID PROPS)
                GROUPLIST])
```

(\FM.FORMATBYGRID

```
  [LAMBDA (DESCRIPTION FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID PROPS)
                                                                   (* jow " 9-May-86 16:05")
```

(* ID and PROPS are specified by \FM.FORMAT. ID is this groups id, and thus the MOTHER of each grid row.
DESCRIPTION is a list of rows, each row a list of item descriptions and groups.
Reverse the rows, then build from bottom up. GRID is the list of columns.
Ignore row and item offsets and make the first column LEFT. This is okay because can achieve the offset by group offset.
\FM.LAYOUTGRID formats each row, and also updates the column grid.
As a second pass, the items in each row are pushed right, to align them with the calculated grid.
This involves extending the extent to include any item/group that is on the last grid position, otherwise the item/group could
get justified out of the extent.)

```
    (LET ((EXTENT (CREATEREGION LEFT BOTTOM 0 0))
          (ITEMLIST (LIST NIL))
          (GROUPLIST (LIST NIL))
          (ROWIDS (LIST NIL))
          (GRID (TCONC (LIST NIL)
                       LEFT))
          (ROWBOTTOM BOTTOM)
          GROUPS ROWITEMS ROWREGION ROWDAUGHTERS ALIGNREGION BOX)
         (for ROWDESC in (REVERSE DESCRIPTION) do (SETQ GROUPS (\FM.LAYOUTGRID ROWDESC FONT GRID ROWBOTTOM
                                                                      ROWSPACE COLUMNSPACE ID))
                                                  (TCONC ROWIDS (\FM.DTOPGROUPID GROUPS))
                                                  [LCONC ITEMLIST (REVERSE (\FM.DTOPGROUPPROP GROUPS 'ITEMS]
                                                  (EXTENDREGION EXTENT (\FM.DTOPGROUPPROP GROUPS 'REGION))
                                                  (LCONC GROUPLIST GROUPS)
                                                  (add ROWBOTTOM (fetch (REGION HEIGHT) of (\FM.DTOPGROUPPROP
                                                                                            GROUPS
                                                                                            'REGION))
                                                       ROWSPACE))
         (SETQ GROUPLIST (CAR GROUPLIST))                          (* grab list from LCONC pair)
         (SETQ ROWIDS (DREVERSE (CAR ROWIDS)))
         (SETQ GRID (CAR GRID))
         (for ROWID in ROWIDS as ROWDESC in DESCRIPTION
            do (SETQ ROWREGION (\FM.DGROUPPROP GROUPLIST ROWID 'REGION))
               (SETQ ROWITEMS (\FM.DGROUPPROP GROUPLIST ROWID 'ITEMS))
               (SETQ ROWDAUGHTERS (\FM.DGROUPPROP GROUPLIST ROWID 'DAUGHTERS))
               (for ITEMDESC in ROWDESC as GRIDPOS in GRID
                  do (if (EQ \FM.GROUPSPEC (CAR ITEMDESC))
                         then (SETQ ALIGNREGION (\FM.DGROUPPROP GROUPLIST (CAR ROWDAUGHTERS)
                                                       'REGION))
                              (SETQ ROWITEMS (CDR (FMEMB [CAR (LAST (\FM.DGROUPPROP GROUPLIST (CAR ROWDAUGHTERS)
                                                                           'ITEMS]
                                                         ROWITEMS)))
                              (if (NEQ GRIDPOS (fetch (REGION LEFT) of ALIGNREGION))
                                  then                             (* need to align a group)
                                       (\FM.PUSHGROUP (CAR ROWDAUGHTERS)
                                              GROUPLIST
                                              (IDIFFERENCE GRIDPOS (fetch (REGION LEFT) of ALIGNREGION))
                                              'RIGHT))
                              (SETQ ROWDAUGHTERS (CDR ROWDAUGHTERS))
                                                                   (* point at next item and group)
                       else (SETQ ALIGNREGION (\FM.ITEMPROP (CAR ROWITEMS)
                                                     'MAXREGION))
                            (replace (REGION LEFT) of (\FM.ITEMPROP (CAR ROWITEMS)
                                                             'REGION)
                               with GRIDPOS)
                            (replace (REGION LEFT) of ALIGNREGION with GRIDPOS)
                            (SETQ ROWITEMS (CDR ROWITEMS))        (* point at next item))
                  finally (EXTENDREGION ROWREGION ALIGNREGION)
                        [if (SETQ BOX (\FM.DGROUPPROP GROUPLIST ROWID 'BOX))
```

```
                              then (add (fetch (REGION WIDTH) of ROWREGION)
                                        (IPLUS BOX (\FM.DGROUPPROP GROUPLIST ROWID 'BOXSPACE]
                         (EXTENDREGION EXTENT ROWREGION)))
          (LISTPUT PROPS 'ITEMS (DREVERSE (CAR ITEMLIST)))
          (LISTPUT PROPS 'REGION EXTENT)
          (LISTPUT PROPS 'DAUGHTERS ROWIDS)
          (CONS (\FM.MAKEGROUP ID PROPS)
                GROUPLIST])
```

(\FM.FORMATEXPLICIT

```
  [LAMBDA (DESCRIPTION FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID PROPS)
                                                                      (* jow "17-Apr-86 18:10")

          (* ID and PROPS are specified by \FM.FORMAT. For an explicitely formatted group, just check that the descriptions are
          valid, and figure out the groups extent. If the group is layed out in local coordinates, replace with menu coordinates.
          When a group is encountered within an explicitely formatted group, the LEFT and BOTTOM specs in the inside group locate
          its corner. If the outer group is expressed in group coordinates, then the corner of the outer group is passed on, so that the
          inner group will be in the same system.)

    (LET ((EXTENT (CREATEREGION LEFT BOTTOM 0 0))
          (ITEMLIST (LIST NIL))
          (GROUPLIST (LIST NIL))
          (SUBGROUPIDS (LIST NIL))
          (LOCAL (EQ (LISTGET PROPS 'COORDINATES)
                     'GROUP))
          X)                                                          (* X holds newly created group or item.)
         [for ITEMDESC in DESCRIPTION
            do (if (EQ \FM.GROUPSPEC (CAR ITEMDESC))
                   then                                               (* if item is a group, recurse)
                        (if LOCAL
                            then (SETQ X (\FM.FORMAT (CDR ITEMDESC)
                                                'EXPLICIT FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID))
                          else (SETQ X (\FM.FORMAT (CDR ITEMDESC)
                                              'EXPLICIT FONT 0 0 ROWSPACE COLUMNSPACE ID)))
                        (TCONC SUBGROUPIDS (\FM.DTOPGROUPID X))
                        [LCONC ITEMLIST (COPY (\FM.DTOPGROUPPROP X 'ITEMS]
                        (EXTENDREGION EXTENT (\FM.DTOPGROUPPROP X 'REGION))
                        (LCONC GROUPLIST X)
                 else (\FM.CHECKDESCRIPTION ITEMDESC)                 (* check description and left and bottom specs)
                      (if LOCAL
                          then                                        (* change group coord's into menu coord's)
                               (SETQ X (\FM.CREATEITEM ITEMDESC FONT LEFT BOTTOM ID))
                        else (SETQ X (\FM.CREATEITEM ITEMDESC FONT 0 0 ID)))
                      (TCONC ITEMLIST X)
                      (EXTENDREGION EXTENT (\FM.ITEMPROP X 'MAXREGION]
         (LISTPUT PROPS 'ITEMS (DREVERSE (CAR ITEMLIST)))
         (LISTPUT PROPS 'REGION EXTENT)
         (LISTPUT PROPS 'DAUGHTERS (CAR SUBGROUPIDS))
         (CONS (\FM.MAKEGROUP ID PROPS)
               (CAR GROUPLIST])
```

(\FM.LAYOUTROW

```
  [LAMBDA (ROW FONT LEFT BOTTOM ROWSPACE COLUMNSPACE MOTHER ID PROPS)
                                                                      (* jow "17-Apr-86 18:11")

          (* MOTHER mother group id. ID and PROPS belong to the group which is this row, and are currently unspecified on entry
          (later versions???) Layout the items in a row starting at LEFT and BOTTOM including any individual item offsets, leaving
          COLUMNSPACE bits between items in the row. Nested groups get default row format.
          Return a list of groups.)

    (\FM.SETUPPROPS ROW '(ID FONT LEFT BOTTOM COLUMNSPACE))
    (LET (OLDCORNER BOXOFFSET)
         (\FM.CHECKFORBOX)
         (LET ((EXTENT (CREATEREGION LEFT BOTTOM 0 0))
               (ITEMLIST (LIST NIL))
               (GROUPLIST (LIST NIL))
               (SUBGROUPIDS (LIST NIL))
               (GROUPLEFT LEFT)
               X)                                                     (* X holds newly created group or item)
              (for ITEMDESC in ROW do [if (EQ \FM.GROUPSPEC (CAR ITEMDESC))
                                          then (SETQ X (\FM.FORMAT (CDR ITEMDESC)
                                                              'ROW FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID))
                                               (TCONC SUBGROUPIDS (\FM.DTOPGROUPID X))
                                               [LCONC ITEMLIST (COPY (\FM.DTOPGROUPPROP X 'ITEMS]
                                               (EXTENDREGION EXTENT (\FM.DTOPGROUPPROP X 'REGION))
                                               (LCONC GROUPLIST X)
                                        else (\FM.CHECKDESCRIPTION ITEMDESC)
                                             (SETQ X (\FM.CREATEITEM ITEMDESC FONT LEFT BOTTOM ID))
                                             (TCONC ITEMLIST X)
                                             (EXTENDREGION EXTENT (\FM.ITEMPROP X 'MAXREGION]
                                     (SETQ LEFT (IPLUS GROUPLEFT (fetch (REGION WIDTH) of EXTENT)
                                                       COLUMNSPACE)))
              (\FM.UPDATEFORBOX)
              (LISTPUT PROPS 'ITEMS (CAR ITEMLIST))
              (LISTPUT PROPS 'REGION EXTENT)
```

```
               (LISTPUT PROPS 'DAUGHTERS (CAR SUBGROUPIDS))
               (CONS (\FM.MAKEGROUP ID PROPS)
                     (CAR GROUPLIST])
```

(\FM.LAYOUTCOLUMN

```
  [LAMBDA (COLUMN FONT LEFT BOTTOM ROWSPACE COLUMNSPACE MOTHER ID PROPS)
                                                                        (* jow "17-Apr-86 18:11")

            (* MOTHER is mother group id. ID and PROPS belong to the group which is this row, and are currently unspecified on entry
            (later versions???) Called by \FM.FORMATBYCOLUMN to layout the items in a column.
            The COLUMN is reversed, so that it is built from bottom up. Column starts at LEFT, BOTTOM, with ROWSPACE bits
            between items. Nested groups default to column format. The items are returned in the order that they are declared.)

    (\FM.SETUPPROPS COLUMN '(ID FONT LEFT BOTTOM ROWSPACE))
    (LET (OLDCORNER BOXOFFSET)
         (\FM.CHECKFORBOX)
         (LET ((EXTENT (CREATEREGION LEFT BOTTOM 0 0))
               (ITEMLIST (LIST NIL))
               (GROUPLIST (LIST NIL))
               (SUBGROUPIDS (LIST NIL))
               (GROUPBOTTOM BOTTOM)
               X)                                                       (* X holds newly created group or item)
              (for ITEMDESC in (DREVERSE COLUMN) do [if (EQ \FM.GROUPSPEC (CAR ITEMDESC))
                                                          then (SETQ X (\FM.FORMAT (CDR ITEMDESC)
                                                                              'COLUMN FONT LEFT BOTTOM ROWSPACE
                                                                              COLUMNSPACE ID))
                                                               (TCONC SUBGROUPIDS (\FM.DTOPGROUPID X))
                                                               [LCONC ITEMLIST (REVERSE (\FM.DTOPGROUPPROP
                                                                                         X
                                                                                         'ITEMS]
                                                               (EXTENDREGION EXTENT (\FM.DTOPGROUPPROP X
                                                                                           'REGION))
                                                               (LCONC GROUPLIST X)
                                                        else (\FM.CHECKDESCRIPTION ITEMDESC)
                                                             (SETQ X (\FM.CREATEITEM ITEMDESC FONT LEFT BOTTOM ID))
                                                             (TCONC ITEMLIST X)
                                                             (EXTENDREGION EXTENT (\FM.ITEMPROP X 'MAXREGION]
                                                      (SETQ BOTTOM (IPLUS GROUPBOTTOM (fetch (REGION HEIGHT)
                                                                                         of EXTENT)
                                                                          ROWSPACE)))
              (\FM.UPDATEFORBOX)
              (LISTPUT PROPS 'ITEMS (DREVERSE (CAR ITEMLIST)))
              (LISTPUT PROPS 'REGION EXTENT)
              (LISTPUT PROPS 'DAUGHTERS (DREVERSE (CAR SUBGROUPIDS)))
              (CONS (\FM.MAKEGROUP ID PROPS)
                    (CAR GROUPLIST])
```

(\FM.LAYOUTGRID

```
  [LAMBDA (ROW FONT GRID BOTTOM ROWSPACE COLUMNSPACE MOTHER ID PROPS)
                                                                        (* jow "24-Apr-86 23:15")

            (* MOTHER is mother group id. ID and PROPS belong to the group which is this row, and are currently unspecified on entry
            (later versions???) ROW is a list of item descriptions. Layout the items according to GRID, updating GRID as you go.
            GRID is a list (built in TCONC format) of column positions, ie the first number in the list is the left position of the first item in
            each row, and so on. GRID will always specify a first column. For each row, update GRID to accomodate the items in that
            row, by pushing the grid right as necessary for new items. Then \FM.FORMATBYGRID will use this grid to align all items by
            column.)

    (\FM.SETUPPROPS ROW '(ID FONT BOTTOM COLUMNSPACE))
    (LET ((GRIDLEN (FLENGTH (CAR GRID)))
          OLDCORNER BOXOFFSET)
         (if (LISTGET PROPS 'BOX)
             then                                                       (* offset group to allow for box. Like CHECKFORBOX;
                                                                        slightly different for GRID.)
                  (OR (LISTGET PROPS 'BOXSHADE)
                      (LISTPUT PROPS 'BOXSHADE BLACKSHADE))
                  (OR (LISTGET PROPS 'BOXSPACE)
                      (LISTPUT PROPS 'BOXSPACE \FM.BOXSPACE))
                  (SETQ OLDCORNER (CONS LEFT BOTTOM))
                  [SETQ BOXOFFSET (IPLUS (LISTGET PROPS 'BOX)
                                         (LISTGET PROPS 'BOXSPACE]
                  (\FM.UPDATEGRID 1 (IPLUS (CAAR GRID)
                                           BOXOFFSET))                  (* shift grid to account for box)
                  (add BOTTOM BOXOFFSET))
         (LET ((EXTENT (CREATEREGION (CAAR GRID)
                              BOTTOM 0 0))
               (ITEMLIST (LIST NIL))
               (GROUPLIST (LIST NIL))
               (SUBGROUPIDS (LIST NIL))
               (ITEMNUM 0)
               X GROUPREGION LEFT NEXTLEFT)
              (for ITEMDESC in ROW do (add ITEMNUM 1)
                                      (SETQ LEFT (CAR (FNTH (CAR GRID)
                                                            ITEMNUM)))
                                      (if (EQ \FM.GROUPSPEC (CAR ITEMDESC))
                                                 then (SETQ X (\FM.FORMAT (CDR ITEMDESC)
                                                                      'TABLE FONT LEFT BOTTOM ROWSPACE COLUMNSPACE ID))
                                                      (TCONC SUBGROUPIDS (\FM.DTOPGROUPID X))
                                                      [LCONC ITEMLIST (COPY (\FM.DTOPGROUPPROP X 'ITEMS]
                                                      (SETQ GROUPREGION (\FM.DTOPGROUPPROP X 'REGION))
                                                      (EXTENDREGION EXTENT GROUPREGION)
                                                      (LCONC GROUPLIST X)
                                                      (SETQ LEFT (fetch (REGION LEFT) of GROUPREGION))
                                                      (SETQ NEXTLEFT (IPLUS LEFT (fetch (REGION WIDTH) of GROUPREGION)
                                                                            COLUMNSPACE))
                                               else (\FM.CHECKDESCRIPTION ITEMDESC)
                                                    (SETQ X (\FM.CREATEITEM ITEMDESC FONT LEFT BOTTOM ID))
                                                    (TCONC ITEMLIST X)
                                                    (SETQ GROUPREGION (\FM.ITEMPROP X 'MAXREGION))
                                                    (EXTENDREGION EXTENT GROUPREGION)
                                                    (SETQ LEFT (fetch (REGION LEFT) of GROUPREGION))
                                                    (SETQ NEXTLEFT (IPLUS LEFT (fetch (REGION WIDTH) of GROUPREGION)
                                                                          COLUMNSPACE)))
                                             (\FM.UPDATEGRID ITEMNUM LEFT)
                                                                              (* mark where this one went)
                                             (\FM.UPDATEGRID (ADD1 ITEMNUM)
                                                    NEXTLEFT)                 (* mark where the next one will go)
                                             )
                 (\FM.UPDATEFORBOX)
                 (LISTPUT PROPS 'ITEMS (CAR ITEMLIST))
                 (LISTPUT PROPS 'REGION EXTENT)
                 (LISTPUT PROPS 'DAUGHTERS (CAR SUBGROUPIDS))
                 (CONS (\FM.MAKEGROUP ID PROPS)
                       (CAR GROUPLIST])
```

(\FM.JUSTIFYITEMS

```
  [LAMBDA (GROUPS GROUPID)                                                ; Edited  6-Sep-94 18:52 by jds

    ;; justify the items in group GROUPID, within that item's group's extent.  If GROUPID is nil, do top group.  This will descend into subgroups, and
    ;; justify those items within that group.

    (LET (EXTENT EXTENTLEFT EXTENTBOTTOM ITEMREGION ITEMMAXREGION ITEMWIDTH ITEMHEIGHT THISGROUP MOTHER)
         (OR GROUPID (SETQ GROUPID (CAAR GROUPS)))
         (PROG (($$LST1 (LISTGET (CDR (FASSOC GROUPID GROUPS))
                               'ITEMS))
                $$VAL ITEM)
           $$LP
               [SETQ ITEM (CAR (OR (LISTP $$LST1)
                                   (GO $$OUT]
               (COND
                  ([AND (NOT (LISTGET (FETCHFIELD '(FREEMENUITEM 24 POINTER)
                                             ITEM)
                                    'HJUSTIFY))
                        (NOT (LISTGET (FETCHFIELD '(FREEMENUITEM 24 POINTER)
                                             ITEM)
                                    'VJUSTIFY]
                   (GO $$ITERATE)))
               [COND
                  ((NEQ THISGROUP (fetch (FREEMENUITEM FM.GROUPID) of ITEM))
                   (SETQ THISGROUP (fetch (FREEMENUITEM FM.GROUPID) of ITEM))
                   [COND
                      [(EQ (LISTGET (CDR (FASSOC THISGROUP GROUPS))
                                  'CL:FORMAT)
                           'EXPLICIT)
                       (SETQ EXTENT (\FM.DGROUPPROP GROUPS THISGROUP 'REGION]
                      (T (SETQ MOTHER (LISTGET (CDR (FASSOC THISGROUP GROUPS))
                                             'MOTHER))
                         (SETQ EXTENT (LISTGET (CDR (FASSOC MOTHER GROUPS))
                                             'REGION]
                   (SETQ EXTENTLEFT (CAR EXTENT))
                   (SETQ EXTENTBOTTOM (CADR EXTENT]
               (SETQ ITEMREGION (fetch (FREEMENUITEM FM.REGION) of ITEM))
               (SETQ ITEMMAXREGION (fetch (FREEMENUITEM FM.MAXREGION) of ITEM))
               [COND
                  ((LISTGET (FETCHFIELD '(FREEMENUITEM 24 POINTER)
                                   ITEM)
                          'HJUSTIFY)
                   (SETQ ITEMWIDTH (CADDR ITEMMAXREGION))
                   (CAR (RPLACA ITEMREGION (SELECTQ (LISTGET (FETCHFIELD '(FREEMENUITEM 24 POINTER)
                                                                    ITEM)
                                                           'HJUSTIFY)
                                                (LEFT EXTENTLEFT)
                                                (CENTER (IPLUS EXTENTLEFT (IQUOTIENT (IDIFFERENCE (CADDR EXTENT)
                                                                                                 ITEMWIDTH)
                                                                                 2)))
                                                (RIGHT (IPLUS EXTENTLEFT (IDIFFERENCE (CADDR EXTENT)
                                                                                ITEMWIDTH)))
                                                NIL)))
                   (CAR (RPLACA ITEMMAXREGION (CAR ITEMREGION]
               [COND
                  ((LISTGET (FETCHFIELD '(FREEMENUITEM 24 POINTER)
                                   ITEM)
                        'VJUSTIFY)
                    (SETQ ITEMHEIGHT (CADDDR ITEMMAXREGION))
                    (CAR (RPLACA (CDR ITEMREGION)
                                 (SELECTQ (\FM.ITEMPROP ITEM 'VJUSTIFY)
                                      (TOP (IPLUS EXTENTBOTTOM (IDIFFERENCE (CADDDR EXTENT)
                                                                      ITEMHEIGHT)))
                                      (MIDDLE (IPLUS EXTENTBOTTOM (IQUOTIENT (IDIFFERENCE (CADDDR EXTENT)
                                                                                   ITEMHEIGHT)
                                                                         2)))
                                      (BOTTOM EXTENTBOTTOM)
                                      NIL)))
                    (CAR (RPLACA (CDR ITEMMAXREGION)
                                 (CADR ITEMREGION]
          $$ITERATE
              (SETQ $$LST1 (CDR $$LST1))
              (GO $$LP)
          $$OUT
              (RETURN $$VAL])
```

(\FM.JUSTIFYGROUPS

```
  [LAMBDA (GROUPS GROUPID)                                   ; Edited  6-Sep-94 19:22 by jds

             (* justify group GROUPID in GROUPS structure. This will descend into the daughter groups.
             If GROUPID is nil, start at the top.)

    (LET (EXTENT MOTHEREXTENT MOTHER HJUST VJUST)
         (OR GROUPID (SETQ GROUPID (\FM.DTOPGROUPID GROUPS)))
         (SETQ HJUST (\FM.DGROUPPROP GROUPS GROUPID 'HJUSTIFY))
         (SETQ VJUST (\FM.DGROUPPROP GROUPS GROUPID 'VJUSTIFY))
         [COND
            ((OR HJUST VJUST)
             (SETQ MOTHER (\FM.DGROUPPROP GROUPS GROUPID 'MOTHER))
             (SETQ MOTHEREXTENT (\FM.DGROUPPROP GROUPS MOTHER 'REGION))
             (SETQ EXTENT (\FM.DGROUPPROP GROUPS GROUPID 'REGION))
             (COND
                (HJUST (SELECTQ HJUST
                           (LEFT)
                           (RIGHT (\FM.PUSHGROUP GROUPID GROUPS (- (fetch (REGION RIGHT) of MOTHEREXTENT)
                                                                   (fetch (REGION RIGHT) of EXTENT))
                                         'RIGHT))
                           (CENTER)
                           NIL]
         (for DAUGHTER in (\FM.DGROUPPROP GROUPS GROUPID 'DAUGHTERS) do (\FM.JUSTIFYGROUPS GROUPS DAUGHTER])
```

(\FM.PUSHGROUP

```
  [LAMBDA (GROUPID GROUPS AMOUNT DIR)                        (* jow "12-Apr-86 18:25")

             (* GROUPS is freemenu groups structure, GROUPID is id of group in GROUPS to push.
             If GROUPID is NIL, then push top group. Push each item by AMOUNT in the DIR direction.
             Update the groups region. Currently this function only knows about pushing UP and RIGHT,)

    (OR GROUPID (SETQ GROUPID (\FM.DTOPGROUPID GROUPS)))
    (for ITEM in (\FM.DGROUPPROP GROUPS GROUPID 'ITEMS)
       do (SELECTQ DIR
              (UP (add (fetch (REGION BOTTOM) of (\FM.ITEMPROP ITEM 'REGION))
                       AMOUNT)
                  [replace (REGION BOTTOM) of (\FM.ITEMPROP ITEM 'MAXREGION) with (fetch (REGION BOTTOM)
                                                                                   of (\FM.ITEMPROP ITEM
                                                                                             'REGION])
              (RIGHT (add (fetch (REGION LEFT) of (\FM.ITEMPROP ITEM 'REGION))
                          AMOUNT)
                     [replace (REGION LEFT) of (\FM.ITEMPROP ITEM 'MAXREGION) with (fetch (REGION LEFT)
                                                                                    of (\FM.ITEMPROP
                                                                                        ITEM
                                                                                        'REGION])
              NIL))
    (SELECTQ DIR
        (UP (add (fetch (REGION BOTTOM) of (\FM.DGROUPPROP GROUPS GROUPID 'REGION))
                 AMOUNT))
        (RIGHT (add (fetch (REGION LEFT) of (\FM.DGROUPPROP GROUPS GROUPID 'REGION))
                    AMOUNT))
        NIL])
```

(\FM.CHECKDESCRIPTION

```
  [LAMBDA (ID)                                               (* jow "21-May-86 16:14")

             (* check the item description for errors. This is done before creating the item.
             The general errors are checked first, and then the type specific errors are checked.
             ALSO, if the item is boxed, fill out the description with all of the boxing info.)

    (LET [(LABEL (LISTGET ID 'LABEL))
          (TYPE (OR (LISTGET ID 'TYPE)
                    'MOMENTARY]                              (* ------------------------------ TYPE FIELD)
         (if (NOT (FMEMB TYPE \FM.ITEM-TYPES))
        then (ERROR "Invalid TYPE:" ID))                    (* ------------------------------ LABEL FIELD)
    (if (NOT (OR (AND LABEL (ATOM LABEL))
                 (STRINGP LABEL)
                 (BITMAPP LABEL)))
        then (ERROR "Invalid LABEL.  Atom, string, or bitmap expected:" ID))
                                                            (* ------------------------------ FIXP FIELDS)
    (for PROP in '(LEFT BOTTOM MAXWIDTH HAXHEIGHT BOX BOXSPACE)
       do (if [AND (LISTGET ID PROP)
                   (NOT (FIXP (LISTGET ID PROP]
              then (ERROR (CONCAT "Invalid " PROP ".  Fixp expected:")
                          ID)))                             (* ------------------------------ JUSTIFICATION FIELDS)
    (if (AND (LISTGET ID 'HJUSTIFY)
             (NOT (FMEMB (LISTGET ID 'HJUSTIFY)
                         \FM.HJUSTIFY-SPECS)))
        then (ERROR (CONCAT "Invalid HJUSTIFY.  One of " \FM.HJUSTIFY-SPECS " expected:" ID)))
    (if (AND (LISTGET ID 'VJUSTIFY)
             (NOT (FMEMB (LISTGET ID 'VJUSTIFY)
                         \FM.VJUSTIFY-SPECS)))
        then (ERROR (CONCAT "Invalid VJUSTIFY.  One of " \FM.VJUSTIFY-SPECS " expected:" ID)))
                                                            (* ------------------------------ TEXTURE FIELDS)
    (for PROP in '(BACKGROUND BOXSHADE) do (if [AND (LISTGET ID PROP)
                                                    (NOT (TEXTUREP (LISTGET ID PROP]
                                               then (ERROR (CONCAT "Invalid " PROP ".  Shade expected:")
                                                           ID)))
                                                            (* ------------------------------ HIGHLIGHT FIELD)
    (if [AND (LISTGET ID 'HIGHLIGHT)
             [NOT (ATOM (LISTGET ID 'HIGHLIGHT]
             [NOT (STRINGP (LISTGET ID 'HIGHLIGHT]
             (NOT (BITMAPP (LISTGET ID 'HIGHLIGHT]
        then (ERROR "Invalid HIGHLIGHT.  Texture or Label expected:" ID))
                                                            (* ------------------------------ FUNCTION FIELDS)
    (for PROP in '(SELECTEDFN DOWNFN HELDFN MOVEDFN)
       do (if [AND (LISTGET ID PROP)
                   (NOT (ATOM (LISTGET ID PROP)))
                   (NOT (LISTP (LISTGET ID PROP]
              then (ERROR (CONCAT "Invalid " PROP ".  Atomic function name expected:")
                          ID)))                             (* ------------------------------ TYPE SPECIFIC CHECKS)
    [if (LISTGET ID 'BOX)
        then                                                (* fill out box info in description)
             (OR (LISTGET ID 'BOXSHADE)
                 (LISTPUT ID 'BOXSHADE BLACKSHADE))
             (LISTPUT ID 'BOXOFFSET (IPLUS (LISTGET ID 'BOX)
                                           (OR (LISTGET ID 'BOXSPACE)
                                               \FM.BOXSPACE]
    (SELECTQ TYPE
        (3STATE (if [AND (LISTGET ID 'OFF)
                         [NOT (ATOM (LISTGET ID 'OFF]
                         [NOT (STRINGP (LISTGET ID 'OFF]
                         (NOT (BITMAPP (LISTGET ID 'OFF]
                    then (ERROR "Invalid OFF.  Texture or Label expected:" ID)))
        (STATE (if [AND (LISTGET ID 'CHANGESTATE)
                        (NOT (ATOM (LISTGET ID 'CHANGESTATE]
                   then (ERROR "Invalid CHANGESTATE  property.  Atomic function name expected:" ID))
               (if [AND (LISTGET ID 'MENUITEMS)
                        (NOT (LISTP (LISTGET ID 'MENUITEMS]
                   then (ERROR "Invalid MENUITEMS property.  List of items expected:" ID)))
        (NWAY (if (NOT (LISTGET ID 'COLLECTION))
                  then (ERROR "Unspecified COLLECTION for NWAY item:" ID)))
        (EDIT (if (BITMAPP LABEL)
                  then (ERROR "Edit item label must be string or atom." ID)))
        NIL])
```

(\FM.CHECKPROPS

```
  [LAMBDA (PROPS)                                           (* jow "28-Oct-86 18:37")
    (if (AND (LISTGET PROPS 'FORMAT)
             (NOT (FMEMB (LISTGET PROPS 'FORMAT)
                         \FM.FORMAT-TYPES)))
        then (ERROR "PROPS Error.  Invalid FORMAT:" PROPS))
    (for PROP in '(LEFT BOTTOM ROWSPACE COLUMNSPACE BOX BOXSPACE)
       do (if [AND (LISTGET PROPS PROP)
                   (NOT (FIXP (LISTGET PROPS PROP]
              then (ERROR (CONCAT "PROPS Error.  FIXP expected for " PROP " property:")
                          PROPS)))
    (for PROP in '(BOXSHADE BACKGROUND) do (if [AND (LISTGET PROPS PROP)
                                                    (NOT (TEXTUREP (LISTGET PROPS PROP]
                                               then (ERROR (CONCAT "PROPS Error.  TEXTURE expected for " PROP "
                                                                  property:")
                                                           PROPS])
```

(\FM.CREATEITEM

```
  [LAMBDA (ID FONTDEFAULT LEFT BOTTOM GROUPID)              (* jow "17-Apr-86 19:28")

          (* create an item at position LEFT and BOTTOM as specified by the formatter.
          Add item offsets given in the description to this position. Set the items region to the minimum of the label size and the max
          size specified.)
```

```
    (add LEFT (OR (LISTGET ID 'LEFT)
                  0))
    (add BOTTOM (OR (LISTGET ID 'BOTTOM)
                    0))
    (LET* [(TYPE (OR (LISTGET ID 'TYPE)
                     'MOMENTARY))
           (LABEL (LISTGET ID 'LABEL))
           (FONT (OR [AND (LISTGET ID 'FONT)
                          (APPLY* (FUNCTION FONTCREATE)
                                 (LISTGET ID 'FONT]
                     FONTDEFAULT))
           (REGIONS (\FM.GETREGIONS ID LEFT BOTTOM FONT))
           (BITMAPS (\FM.GETBITMAPS ID FONT (CAR REGIONS)
                           (CADR REGIONS)))
           (ITEM (create FREEMENUITEM
                        FM.TYPE _ TYPE
                        FM.LABEL _ LABEL
                        FM.ID _ (LISTGET ID 'ID)
                        FM.GROUPID _ GROUPID
                        FM.INITSTATE _ (LISTGET ID 'INITSTATE)
                        FM.FONT _ FONT
                        FM.BITMAP _ (CAR BITMAPS)
                        FM.HIGHLIGHT _ (CADR BITMAPS)
                        FM.REGION _ (CAR REGIONS)
                        FM.MAXREGION _ (CADDR REGIONS)
                        FM.MESSAGE _ (LISTGET ID 'MESSAGE)
                        FM.LINKS _ (OR (LISTGET ID 'LINKS)
                                       (LIST NIL))
                        FM.DOWNFN _ (OR (LISTGET ID 'DOWNFN)
                                        (FUNCTION NILL))
                        FM.HELDFN _ (OR (LISTGET ID 'HELDFN)
                                        (FUNCTION NILL))
                        FM.MOVEDFN _ (OR (LISTGET ID 'MOVEDFN)
                                         (FUNCTION NILL))
                        FM.SELECTEDFN _ (OR (LISTGET ID 'SELECTEDFN)
                                            (FUNCTION NILL]
          (\FM.READUSERDATA ITEM ID)
          (APPLY* (PACK* "\FM." TYPE "-SETUP")
                 ITEM REGIONS)                              (* pass REGIONS to setup fn, since might need highlightregion,
          etc.)
          ITEM])
```

(\FM.GETREGIONS

```
  [LAMBDA (ID LEFT BOTTOM FONT)                             (* jow "19-Apr-86 21:41")

          (* Called by the formatter to determine the region an item will occupy.
          LEFT and BOTTOM are the items proposed position, determined by the formatter.
          If the item is boxed, then the region is the region of the box, not the label in the box.
          Return a list containing the item region, the highlight region, and the max region.)

    (LET* [(WIDTH (\FM.ITEMWIDTH (LISTGET ID 'LABEL)
                         FONT))
           (HEIGHT (\FM.ITEMHEIGHT (LISTGET ID 'LABEL)
                          FONT))
           (HL (LISTGET ID 'HIGHLIGHT))
           (HIGHLIGHT (OR (AND (ATOM HL)
                               (NOT (TEXTUREP HL))
                               HL)
                          (BITMAPP HL)
                          (STRINGP HL)))
           (HIGHLIGHTWIDTH (OR (AND HIGHLIGHT (\FM.ITEMWIDTH HIGHLIGHT FONT))
                               0))
           (HIGHLIGHTHEIGHT (OR (AND HIGHLIGHT (\FM.ITEMHEIGHT HIGHLIGHT FONT))
                                0))
           (MAXWIDTH (OR (LISTGET ID 'MAXWIDTH)
                         (IMAX WIDTH HIGHLIGHTWIDTH)))
           (MAXHEIGHT (OR (LISTGET ID 'MAXHEIGHT)
                          (IMAX HEIGHT HIGHLIGHTHEIGHT)))
           (BOXOFFSET (AND (LISTGET ID 'BOXOFFSET)
                           (ITIMES 2 (LISTGET ID 'BOXOFFSET]
          (if BOXOFFSET
              then (SETQ WIDTH (IPLUS BOXOFFSET MAXWIDTH))
                   (SETQ HEIGHT (IPLUS BOXOFFSET MAXHEIGHT))
                   (LIST (CREATEREGION LEFT BOTTOM WIDTH HEIGHT)
                         (AND HIGHLIGHT (CREATEREGION LEFT BOTTOM WIDTH HEIGHT))
                         (CREATEREGION LEFT BOTTOM WIDTH HEIGHT))
            else (LIST (CREATEREGION LEFT BOTTOM (IMIN WIDTH MAXWIDTH)
                              (IMIN HEIGHT MAXHEIGHT))
                       (AND HIGHLIGHT (CREATEREGION LEFT BOTTOM (IMIN HIGHLIGHTWIDTH MAXWIDTH)
                                             (IMIN HIGHLIGHTHEIGHT MAXHEIGHT)))
                       (CREATEREGION LEFT BOTTOM MAXWIDTH MAXHEIGHT])
```

(\FM.GETBITMAPS

```
  [LAMBDA (ID FONT ITEMREGION HIGHLIGHTREGION)              (* jow "18-Apr-86 14:57")
```

```
                                                                   (* Figure out the items bitmap and highlighting requirements.)
    (LET ((BOX (OR (LISTGET ID 'BOX)
                   0))
          (BOXSHADE (LISTGET ID 'BOXSHADE))
          (HIGHLIGHT (LISTGET ID 'HIGHLIGHT))
          (WIDTH (fetch (REGION WIDTH) of ITEMREGION))
          (HEIGHT (fetch (REGION HEIGHT) of ITEMREGION))
          BITMAP HLBITMAP)
         (SETQ BITMAP (\FM.MAKEBITMAP (LISTGET ID 'LABEL)
                             FONT WIDTH HEIGHT ID))
         [COND
            ((OR (AND HIGHLIGHT (ATOM HIGHLIGHT)
                      (NOT (TEXTUREP HIGHLIGHT)))
                 (STRINGP HIGHLIGHT)
                 (BITMAPP HIGHLIGHT))                              (* highlight label specified.)
             (SETQ HLBITMAP (\FM.MAKEBITMAP HIGHLIGHT FONT (fetch (REGION WIDTH) of HIGHLIGHTREGION)
                                   (fetch (REGION HEIGHT) of HIGHLIGHTREGION)
                                   ID)))
            ((OR (TEXTUREP HIGHLIGHT)
                 (AND (LISTGET ID 'BOX)
                      (NEQ BOXSHADE BLACKSHADE)
                      (SETQ HIGHLIGHT BOXSHADE)))                  (* highlight texture was specified, or non-black box with default
                                                                   highlight (boxshade))
             (SETQ HLBITMAP (BITMAPCOPY BITMAP))
             (BLTSHADE HIGHLIGHT HLBITMAP BOX BOX (IDIFFERENCE WIDTH (ITIMES BOX 2))
                    (IDIFFERENCE HEIGHT (ITIMES BOX 2))
                    'PAINT))
            (T                                                     (* invert. Start with bitmap, and invert region inside box.)
               (SETQ HLBITMAP (BITMAPCOPY BITMAP))
               (BITBLT BITMAP BOX BOX HLBITMAP BOX BOX (IDIFFERENCE WIDTH (ITIMES BOX 2))
                      (IDIFFERENCE HEIGHT (ITIMES BOX 2))
                      'INVERT]
         (LIST BITMAP HLBITMAP])
```

(\FM.MAKEBITMAP

```
  [LAMBDA (LABEL FONT WIDTH HEIGHT ID)                             (* jow "18-Apr-86 14:29")
                                                                   (* use ID only for boxing info.)
    (LET ((BOX (LISTGET ID 'BOX))
          (BOXOFFSET (OR (LISTGET ID 'BOXOFFSET)
                         0))
          (BITMAP (BITMAPCREATE WIDTH HEIGHT))
          CLIPPINGREGION)
         [SETQ CLIPPINGREGION (CREATEREGION BOXOFFSET BOXOFFSET (IDIFFERENCE WIDTH (ITIMES BOXOFFSET 2))
                                     (IDIFFERENCE HEIGHT (ITIMES BOXOFFSET 2]
         (if BOX
             then                                                  (* check for boxed item)
                  (BLTSHADE (LISTGET ID 'BOXSHADE)
                         BITMAP)                                   (* do box and background)
                  (BLTSHADE WHITESHADE BITMAP BOX BOX (IDIFFERENCE WIDTH (ITIMES BOX 2))
                         (IDIFFERENCE HEIGHT (ITIMES BOX 2)))      (* copy box into HLBITMAP)
                  )
         (if (BITMAPP LABEL)
             then (BITBLT LABEL 0 0 BITMAP BOXOFFSET BOXOFFSET NIL NIL NIL NIL NIL CLIPPINGREGION)
           else (LET ((STREAM (DSPCREATE BITMAP)))
                     (DSPFONT FONT STREAM)
                     (DSPXPOSITION BOXOFFSET STREAM)
                     (DSPYPOSITION (IPLUS BOXOFFSET (FONTPROP FONT 'DESCENT))
                            STREAM)
                     (DSPCLIPPINGREGION CLIPPINGREGION STREAM)
                     (PRIN1 LABEL STREAM)))
         BITMAP])
```

(\FM.READUSERDATA

```
  [LAMBDA (ITEM DESCRIPTION)                                       (* jow "15-Apr-86 16:58")
                                                                   (* scans DESCRIPTION for user props.
                                                                   Add any prop/value pairs found to ITEM's userdata list.)
    (for X on DESCRIPTION by (CDDR X) do (if (NOT (FMEMB (CAR X)
                                                          \FM.DESCRIPTION-PROPS))
                                              then (LISTPUT (\FM.ITEMPROP ITEM 'USERDATA)
                                                          (CAR X)
                                                          (CADR X)])
```

(\FM.MAKELINKS

```
  [LAMBDA (WINDOW)                                                 (* jow "12-Apr-86 19:07")
                                                                   (* go through items and replace link requests with actual
                                                                   pointers)
    (for ITEM in (WINDOWPROP WINDOW 'FM.ITEMS) do (for LINKTAIL ITEMPTR on (CDR (\FM.ITEMPROP ITEM 'LINKS))
                                                     by (CDDR LINKTAIL) do (SETQ ITEMPTR (CAR LINKTAIL))
                                                                           (RPLACA LINKTAIL (\FM.COERCEITEMPTR
                                                                                             ITEMPTR WINDOW ITEM]
)
```

(\FM.COLLECTNWAYS

```
  [LAMBDA (WINDOW)                                                  (* jow "17-Apr-86 15:28")

             (* go through items in menu, building NWAYS structure. Select the first item in each collection.
             NWAYS structure is list of collection proplists, each beginning with id of collection, and containing STATE of collection, and
             other user props.)

    (LET ((NWAYS (LIST NIL))
          (NWAYIDS (LIST NIL))
          NWAYPROPS ITEMPTR)
         (for ITEM in (WINDOWPROP WINDOW 'FM.ITEMS)
            do (if [AND (EQ (\FM.ITEMPROP ITEM 'TYPE)
                              'NWAY)
                         (NOT (FMEMB (\FM.ITEMPROP ITEM 'COLLECTION)
                                     (CAR NWAYIDS]
                   then                                             (* this is the first nway of this collection)
                         (TCONC NWAYIDS (\FM.ITEMPROP ITEM 'COLLECTION))
                                                                    (* setup NWAYPROPS and STATE)
                         (if (\FM.ITEMPROP ITEM 'NWAYPROPS)
                             then (SETQ NWAYPROPS (\FM.ITEMPROP ITEM 'NWAYPROPS))
                                  (LISTPUT NWAYPROPS 'STATE ITEM)
                           else (SETQ NWAYPROPS (LIST 'STATE ITEM)))
                                                                    (* setup INITSTATE)
                         (if (LISTGET NWAYPROPS 'INITSTATE)
                             then                                   (* make link to specified INITSTATE item)
                                  (SETQ ITEMPTR (LISTGET NWAYPROPS 'INITSTATE))
                                  (LISTPUT NWAYPROPS 'INITSTATE (\FM.COERCEITEMPTR ITEMPTR WINDOW ITEM))
                           else                                     (* MAKE THIS ITEM THE INITSTATE)
                                (LISTPUT NWAYPROPS 'INITSTATE ITEM))
                         (TCONC NWAYS (\FM.MAKEGROUP (CADR NWAYIDS)
                                             NWAYPROPS))            (* this is the selected item)
                         (\FM.TOGGLE-CHANGESTATE ITEM T)))
         (WINDOWPROP WINDOW 'FM.NWAYS (CAR NWAYS])
```

(\FM.SETATTACHPOINT

```
  [LAMBDA (ITEMS WIDTH HEIGHT)                                      (* jow "12-Apr-86 18:02")
                                                                    (* figure out each items attach point based on its position in
                                                                    extent)
    (for ITEM in ITEMS do (\FM.ITEMPROP ITEM 'ATTACHPOINT (\FM.ATTACHPOINT ITEM WIDTH HEIGHT])
```

(\FM.CREATEW

```
  [LAMBDA (GROUPS TITLE BACKGROUND BORDER)                          ; Edited  6-Jan-87 18:32 by woz
                                                                    (* Create a freemenu window. Then setup the window with the
                                                                    necessary freemenu properties.)
    (LET* ([REGION (COPY (LISTGET (CDAR GROUPS)
                                  'REGION]
           (WINDOW (CREATEW (CREATEREGION (fetch (REGION LEFT) of REGION)
                                   (fetch (REGION BOTTOM) of REGION)
                                   (WIDTHIFWINDOW (fetch (REGION WIDTH) of REGION)
                                          BORDER)
                                   (HEIGHTIFWINDOW (fetch (REGION HEIGHT) of REGION)
                                          TITLE BORDER))
                          TITLE BORDER T)))
          (WINDOWPROP WINDOW 'WINDOWENTRYFN '\FM.WINDOWENTRYFN)
          (WINDOWPROP WINDOW 'BUTTONEVENTFN '\FM.BUTTONEVENTFN)
          (WINDOWPROP WINDOW 'RIGHTBUTTONFN '\FM.RIGHTBUTTONFN)
          (WINDOWPROP WINDOW 'REPAINTFN '\FM.REDISPLAYMENU)
          (WINDOWPROP WINDOW 'RESHAPEFN '\FM.RESHAPEFN)
          (WINDOWPROP WINDOW 'INITCORNERSFN '\FM.INITCORNERSFN)
          (WINDOWPROP WINDOW 'OPENFN '\FM.OPENFN)
          (WINDOWPROP WINDOW 'CLOSEFN '\FM.ENDEDIT)
          (WINDOWPROP WINDOW 'SHRINKFN '\FM.ENDEDIT)
          (WINDOWPROP WINDOW 'SCROLLFN 'SCROLLBYREPAINTFN)
          (WINDOWPROP WINDOW 'SCROLLEXTENTUSE '(LIMIT . LIMIT))
          (WINDOWPROP WINDOW 'EXTENT REGION)
          (WINDOWPROP WINDOW 'FM.MINWIDTH (fetch (REGION WIDTH) of REGION))
          (WINDOWPROP WINDOW 'FM.MINHEIGHT (fetch (REGION HEIGHT) of REGION))
          (WINDOWPROP WINDOW 'FM.BUSY NIL)
          (WINDOWPROP WINDOW 'FM.BACKGROUND BACKGROUND)
          (WINDOWPROP WINDOW 'FM.GROUPS GROUPS)
          (WINDOWPROP WINDOW 'FM.ITEMS (LISTGET (CDAR GROUPS)
                                               'ITEMS))
          WINDOW])
```

(\FM.STARTEDIT

```
  [LAMBDA (ITEM WINDOW CLEARFLG)                                    (* jow "17-Oct-86 18:35")
    (RESETLST
        (RESETSAVE NIL (LIST 'WINDOWPROP WINDOW 'FM.BUSY NIL))
        (WINDOWPROP WINDOW 'FM.BUSY T)
        (\FM.EDIT-ITEM ITEM WINDOW CLEARFLG T (if (EQ (\FM.ITEMPROP ITEM 'TYPE)
                                                      'NUMBER)
                                                  then (FUNCTION \FM.NUMBER-CHANGESTATE))))])

)
```

```
(RPAQ? \FM.GROUP-ID-COUNTER 0)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \FM.GROUP-ID-COUNTER)
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS \FM.MAKE-GROUP-ID MACRO (NIL (CONS (CL:INCF \FM.GROUP-ID-COUNTER)
                                               NIL)))

(PUTPROPS \FM.SETUPPROPS MACRO ((DESCRIPTION CHANGEPROPS)
                                (if (EQ \FM.PROPSPEC (CAAR DESCRIPTION))
                                    then (SETQ PROPS (CDAR DESCRIPTION))
                                         (RPLNODE2 DESCRIPTION (CDR DESCRIPTION))
                                                                (* yank props out of row)
                                         (\FM.CHECKPROPS PROPS)
                                         (\FM.SETFORMATPROPS CHANGEPROPS)
                                  else (SETQ PROPS (LIST 'ITEMS NIL))
                                       (SETQ ID (\FM.MAKE-GROUP-ID)))
                                (LISTPUT PROPS 'MOTHER MOTHER)))

(PUTPROPS \FM.SETFORMATPROPS MACRO ((CHANGEPROPS)
                                    (for PROP in CHANGEPROPS
                                       do (SELECTQ PROP
                                              (FORMAT [AND (LISTGET PROPS 'FORMAT)
                                                           (SETQ FORMAT (LISTGET PROPS 'FORMAT])
                                              (FONT [AND (LISTGET PROPS 'FONT)
                                                         (SETQ FONT (APPLY* (FUNCTION FONTCREATE)
                                                                           (LISTGET PROPS 'FONT])
                                              (LEFT (add LEFT (OR (LISTGET PROPS 'LEFT)
                                                                  0)))
                                              (BOTTOM (add BOTTOM (OR (LISTGET PROPS 'BOTTOM)
                                                                      0)))
                                              (ROWSPACE [AND (LISTGET PROPS 'ROWSPACE)
                                                             (SETQ ROWSPACE (LISTGET PROPS 'ROWSPACE])
                                              (COLUMNSPACE [AND (LISTGET PROPS 'COLUMNSPACE)
                                                                (SETQ COLUMNSPACE (LISTGET PROPS
                                                                                          'COLUMNSPACE])
                                              (ID (SETQ ID (OR (LISTGET PROPS 'ID)
                                                               (\FM.MAKE-GROUP-ID))))
                                              NIL))))

(PUTPROPS \FM.CHECKFORBOX MACRO [NIL (COND
                                         ((LISTGET PROPS 'BOX) (* offset group to allow for box.)
                                          (OR (LISTGET PROPS 'BOXSHADE)
                                              (LISTPUT PROPS 'BOXSHADE BLACKSHADE))
                                          (OR (LISTGET PROPS 'BOXSPACE)
                                              (LISTPUT PROPS 'BOXSPACE \FM.BOXSPACE))
                                          (SETQ OLDCORNER (CONS LEFT BOTTOM))
                                          [SETQ BOXOFFSET (IPLUS (LISTGET PROPS 'BOX)
                                                                 (LISTGET PROPS 'BOXSPACE]
                                          (add LEFT BOXOFFSET)
                                          (add BOTTOM BOXOFFSET])

(PUTPROPS \FM.UPDATEFORBOX MACRO [NIL (COND
                                          (BOXOFFSET            (* group is boxed%: readjust group region)
                                                 (replace (REGION LEFT) of EXTENT with (CAR OLDCORNER))
                                                 (replace (REGION BOTTOM) of EXTENT with (CDR OLDCORNER))
                                                 (add (fetch (REGION WIDTH) of EXTENT)
                                                      (ITIMES BOXOFFSET 2))
                                                 (add (fetch (REGION HEIGHT) of EXTENT)
                                                      (ITIMES BOXOFFSET 2])

(PUTPROPS \FM.UPDATEGRID MACRO [(NUM LEFT)
                                (if (IGREATERP NUM GRIDLEN)
                                    then                        (* add this col to grid)
                                         (TCONC GRID LEFT)
                                         (add GRIDLEN 1)
                                  else                          (* this col exists. check alignment)
                                       (LET ((GRIDTAIL (FNTH (CAR GRID)
                                                             NUM)))
                                            (COND
                                               ((IGREATERP LEFT (CAR GRIDTAIL))
                                                                (* push grid column over)
                                                (for TAIL on GRIDTAIL bind (AMOUNT _ (IDIFFERENCE LEFT
                                                                                              (CAR GRIDTAIL)))
                                                   do (add (CAR TAIL)
                                                           AMOUNT])

(PUTPROPS \FM.ITEMWIDTH MACRO ((LABEL FONT)
                               (if (BITMAPP LABEL)
                                   then (BITMAPWIDTH LABEL)
                                 else (STRINGWIDTH LABEL FONT))))
```

```
(PUTPROPS \FM.ITEMHEIGHT MACRO [(LABEL FONT)
                                (if (BITMAPP LABEL)
                                    then (BITMAPHEIGHT LABEL)
                                  else (FONTPROP FONT 'HEIGHT])

(PUTPROPS \FM.ATTACHPOINT MACRO [(ITEM WIDTH HEIGHT)
                                  (LET [(MAXREGION (\FM.ITEMPROP ITEM 'MAXREGION]
                                       (CONS [FIX (FPLUS 0.5 (FQUOTIENT (ITIMES (fetch (REGION WIDTH)
                                                                                    of MAXREGION)
                                                                              (fetch (REGION LEFT) of MAXREGION)
                                                                              )
                                                                    (IDIFFERENCE WIDTH (fetch (REGION WIDTH)
                                                                                           of MAXREGION]
                                             (FIX (FPLUS 0.5 (FQUOTIENT (ITIMES (fetch (REGION HEIGHT)
                                                                                    of MAXREGION)
                                                                              (fetch (REGION BOTTOM)
                                                                                 of MAXREGION))
                                                                    (IDIFFERENCE HEIGHT (fetch (REGION HEIGHT)
                                                                                            of MAXREGION])
)
)

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RPAQQ \FM.FORMAT-TYPES (ROW COLUMN TABLE EXPLICIT))

(RPAQQ \FM.DEFAULTFORMAT ROW)

(RPAQQ \FM.GROUPSPEC GROUP)

(RPAQQ \FM.PROPSPEC PROPS)

(RPAQQ \FM.HJUSTIFY-SPECS (LEFT CENTER RIGHT))

(RPAQQ \FM.VJUSTIFY-SPECS (TOP MIDDLE BOTTOM))

(RPAQQ \FM.BOXSPACE 1)

(RPAQQ \FM.ROWSPACE 2)

(RPAQQ \FM.COLUMNSPACE 10)

(RPAQQ \FM.ITEM-TYPES (MOMENTARY TOGGLE 3STATE NWAY STATE NUMBER EDIT EDITSTART DISPLAY))

(RPAQQ \FM.DESCRIPTION-PROPS (TYPE LABEL LEFT BOTTOM ID GROUPID STATE INITSTATE FONT BITMAP REGION MAXREGION
                                   MESSAGE USERDATA LINKS SYSDOWNFN SYSMOVEDFN SYSSELECTEDFN DOWNFN HELDFN
                                   MOVEDFN SELECTEDFN))

[CONSTANTS (\FM.FORMAT-TYPES '(ROW COLUMN TABLE EXPLICIT))
       (\FM.DEFAULTFORMAT 'ROW)
       (\FM.GROUPSPEC 'GROUP)
       (\FM.PROPSPEC 'PROPS)
       (\FM.HJUSTIFY-SPECS '(LEFT CENTER RIGHT))
       (\FM.VJUSTIFY-SPECS '(TOP MIDDLE BOTTOM))
       (\FM.BOXSPACE 1)
       (\FM.ROWSPACE 2)
       (\FM.COLUMNSPACE 10)
       (\FM.ITEM-TYPES '(MOMENTARY TOGGLE 3STATE NWAY STATE NUMBER EDIT EDITSTART DISPLAY))
       (\FM.DESCRIPTION-PROPS '(TYPE LABEL LEFT BOTTOM ID GROUPID STATE INITSTATE FONT BITMAP REGION MAXREGION
                                     MESSAGE USERDATA LINKS SYSDOWNFN SYSMOVEDFN SYSSELECTEDFN DOWNFN HELDFN
                                     MOVEDFN SELECTEDFN]
)
)

(DECLARE%: EVAL@COMPILE

(DATATYPE FREEMENUITEM
       (FM.TYPE FM.LABEL FM.ID FM.GROUPID FM.STATE FM.INITSTATE FM.FONT FM.BITMAP FM.HIGHLIGHT FM.REGION
              FM.MAXREGION FM.MESSAGE FM.USERDATA FM.LINKS FM.SYSDOWNFN FM.SYSMOVEDFN FM.SYSSELECTEDFN FM.DOWNFN
              FM.HELDFN FM.MOVEDFN FM.SELECTEDFN)
       FM.USERDATA _ (LIST NIL)
       FM.SYSDOWNFN _ (FUNCTION NILL)
       FM.SYSMOVEDFN _ (FUNCTION NILL)
       FM.SYSSELECTEDFN _ (FUNCTION NILL))
)

(/DECLAREDATATYPE 'FREEMENUITEM
       '(POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER
               POINTER POINTER POINTER POINTER POINTER POINTER POINTER POINTER)
       ;; ---field descriptor list elided by lister---
       '42)

;; FREEMENU WINDOWS
```

```
(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS \FM.TRANSPOSE MACRO [(POINT OLD NEW)
                                (FIX (FPLUS 0.5 (FQUOTIENT (FTIMES POINT NEW)
                                                       OLD])
)
)

(DEFINEQ

(\FM.OPENFN
  [LAMBDA (WINDOW)                                              ; Edited 19-Jan-87 17:58 by woz

;;; redisplay the menu when opening, because its state might have been changed while closed.  however, if we're under SHAPEW, then punt, and let the
;;; reshapefn redisplay, to avoid redundancy.

    (\FM.INSUREWINDOW WINDOW)
    (if (NOT (STKPOS 'SHAPEW1))
        then (\FM.REDISPLAYMENU WINDOW])

(\FM.REDISPLAYMENU
  [LAMBDA (WINDOW UPDATEREGION)                                 (* jow "26-Jun-86 14:43")
                                                                (* RIGHT NOW THIS IS DEPENDENT ON THE
                                                                ALIST/PROPLIST STRUCTURE OF GROUPS.)
    (if (OPENWP WINDOW)
        then (LET (REGION BOX BACKGROUND)
                  (\FM.RESETCLIPPINGREGION WINDOW)              (* back to full window)
                  (\FM.FILLWINDOW WINDOW (WINDOWPROP WINDOW 'FM.BACKGROUND))
                  [for GROUP in (WINDOWPROP WINDOW 'FM.GROUPS)
                     do                                         (* blast all boxes and backgrounds)
                        (SETQ GROUP (CDR GROUP))
                        (if (SETQ BOX (LISTGET GROUP 'BOX))
                            then (SETQ REGION (LISTGET GROUP 'REGION))
                                 (BLTSHADE (LISTGET GROUP 'BOXSHADE)
                                        WINDOW NIL NIL NIL NIL NIL REGION)
                                 (BLTSHADE (LISTGET GROUP 'BACKGROUND)
                                        WINDOW
                                        (IPLUS (fetch (REGION LEFT) of REGION)
                                               BOX)
                                        (IPLUS (fetch (REGION BOTTOM) of REGION)
                                               BOX)
                                        (IDIFFERENCE (fetch (REGION WIDTH) of REGION)
                                               (IPLUS BOX BOX))
                                        (IDIFFERENCE (fetch (REGION HEIGHT) of REGION)
                                               (IPLUS BOX BOX)))
                          elseif (SETQ BACKGROUND (LISTGET GROUP 'BACKGROUND))
                            then (BLTSHADE BACKGROUND WINDOW NIL NIL NIL NIL NIL (LISTGET GROUP 'REGION]
                  (for ITEM in (WINDOWPROP WINDOW 'FM.ITEMS) do (\FM.DISPLAYBITMAP ITEM (\FM.ITEMPROP ITEM
                                                                                          'BITMAP)
                                                                               WINDOW])

(\FM.RESHAPEFN
  [LAMBDA (WINDOW B OLDREGION)                                  (* jow "25-Apr-86 11:21")
    (if (NOT (WINDOWPROP WINDOW 'FM.DONTRESHAPE))
        then (\FM.ENDEDIT WINDOW T)
             (LET [(OLDWIDTH (fetch (REGION WIDTH) of OLDREGION))
                   (OLDHEIGHT (fetch (REGION HEIGHT) of OLDREGION))
                   (NEWWIDTH (WINDOWPROP WINDOW 'WIDTH))
                   (NEWHEIGHT (WINDOWPROP WINDOW 'HEIGHT))
                   (MINWIDTH (WINDOWPROP WINDOW 'FM.MINWIDTH))
                   (MINHEIGHT (WINDOWPROP WINDOW 'FM.MINHEIGHT]
                  (COND
                     ((AND (IGEQ OLDWIDTH MINWIDTH)
                           (IGREATERP NEWWIDTH MINWIDTH))
                      (\FM.TRANSPOSEHORZ WINDOW OLDWIDTH NEWWIDTH))
                     ((AND (IGREATERP OLDWIDTH MINWIDTH)
                           (ILEQ NEWWIDTH MINWIDTH))            (* transpose to minimal width)
                      (\FM.TRANSPOSEHORZ WINDOW OLDWIDTH MINWIDTH))
                     ((AND (ILESSP OLDWIDTH MINWIDTH)
                           (IGREATERP NEWWIDTH MINWIDTH))       (* transpose from minimal width)
                      (\FM.TRANSPOSEHORZ WINDOW MINWIDTH NEWWIDTH)))
                  (COND
                     ((AND (IGEQ OLDHEIGHT MINHEIGHT)
                           (IGREATERP NEWHEIGHT MINHEIGHT))
                      (\FM.TRANSPOSEVERT WINDOW OLDHEIGHT NEWHEIGHT))
                     ((AND (IGREATERP OLDHEIGHT MINHEIGHT)
                           (ILEQ NEWHEIGHT MINHEIGHT))          (* transpose to minimal height)
                      (\FM.TRANSPOSEVERT WINDOW OLDHEIGHT MINHEIGHT))
                     ((AND (ILESSP OLDHEIGHT MINHEIGHT)
                           (IGREATERP NEWHEIGHT MINHEIGHT))     (* transpose from minimal height)
                      (\FM.TRANSPOSEVERT WINDOW MINHEIGHT NEWHEIGHT)))
                  (\FM.UPDATEGROUPEXTENT (WINDOWPROP WINDOW 'FM.GROUPS))
```

```
                 (WINDOWPROP WINDOW 'EXTENT (\FM.WINDOWEXTENT WINDOW))
                                                                         (* grab new extent)
                 ))
     (\FM.UNSCROLLWINDOW WINDOW)
     (FM.REDISPLAYMENU WINDOW])
```

(\FM.UNSCROLLWINDOW

```
  [LAMBDA (WINDOW)                                                       (* jow "12-Apr-86 15:22")

             (* called after reshaping WINDOW; resets XOFFSET and YOFFSET to unscroll window Clipping region set back to copy of
             full WINDOW)

    (DSPXOFFSET [IPLUS (WINDOWPROP WINDOW 'BORDER)
                       (fetch (REGION LEFT) of (WINDOWPROP WINDOW 'REGION]
           WINDOW)
    (DSPYOFFSET [IPLUS (WINDOWPROP WINDOW 'BORDER)
                       (fetch (REGION BOTTOM) of (WINDOWPROP WINDOW 'REGION]
           WINDOW)
    (\FM.RESETCLIPPINGREGION WINDOW])
```

(\FM.RESETCLIPPINGREGION

```
  [LAMBDA (WINDOW)                                                       (* jow "10-Apr-86 21:52")
                                                                         (* reset the clipping region of WINDOW to the windows full
                                                                         expanse.)
    (DSPCLIPPINGREGION (CREATEREGION (IDIFFERENCE (IPLUS (fetch (REGION LEFT) of (WINDOWPROP WINDOW 'REGION))
                                                         (WINDOWPROP WINDOW 'BORDER))
                                                  (DSPXOFFSET NIL WINDOW))
                                     (IDIFFERENCE (IPLUS (fetch (REGION BOTTOM) of (WINDOWPROP WINDOW 'REGION))
                                                         (WINDOWPROP WINDOW 'BORDER))
                                            (DSPYOFFSET NIL WINDOW))
                                     (WINDOWPROP WINDOW 'WIDTH)
                                     (WINDOWPROP WINDOW 'HEIGHT))
           WINDOW])
```

(\FM.FILLWINDOW

```
  [LAMBDA (WINDOW SHADE)                                                 (* jow "11-Apr-86 11:51")

             (* fill entire window up to border with shade. Rely on clippingregion being full window on entry.
             Rely on border space is 2 bits.)

    (LET ((REGION (DSPCLIPPINGREGION NIL WINDOW)))
         (RESETLST
             (RESETSAVE NIL (LIST 'DSPCLIPPINGREGION REGION WINDOW))
             (DSPCLIPPINGREGION (CREATEREGION (IDIFFERENCE (fetch (REGION LEFT) of REGION)
                                                      2)
                                       (IDIFFERENCE (fetch (REGION BOTTOM) of REGION)
                                              2)
                                       (IPLUS 4 (fetch (REGION WIDTH) of REGION))
                                       (IPLUS 4 (fetch (REGION HEIGHT) of REGION)))
                    WINDOW)
             (DSPFILL NIL SHADE NIL WINDOW))])
```

(\FM.INITCORNERSFN

```
  [LAMBDA (WINDOW)                                                       (* jow " 3-Apr-86 23:35")

             (* called by SHAPEW to provide the initial corners of the reshape ghost box, in the form
             (x1 y1 x2 y2)%, where 1 is fixed and 2 is tracked. respond with the freemenus MINIMAL SHAPE leaving left, bottom as they
             are.)

    (LET [[LEFT (fetch (REGION LEFT) of (WINDOWPROP WINDOW 'REGION]
          (BOTTOM (fetch (REGION BOTTOM) of (WINDOWPROP WINDOW 'REGION]
         (LIST LEFT BOTTOM [IPLUS LEFT (WIDTHIFWINDOW (WINDOWPROP WINDOW 'FM.MINWIDTH)
                                              (WINDOWPROP WINDOW 'BORDER]
               (IPLUS BOTTOM (HEIGHTIFWINDOW (WINDOWPROP WINDOW 'FM.MINHEIGHT)
                                    (WINDOWPROP WINDOW 'TITLE)
                                    (WINDOWPROP WINDOW 'BORDER])
```

(\FM.TRANSPOSEHORZ

```
  [LAMBDA (WINDOW OLDWIDTH NEWWIDTH)                                     (* jow "12-Apr-86 18:27")
                                                                         (* transpose left point.)
    (for ITEM REGION in (WINDOWPROP WINDOW 'FM.ITEMS) do (SETQ REGION (\FM.ITEMPROP ITEM 'REGION))
                                                         (replace (REGION LEFT) of REGION
                                                            with (\FM.TRANSPOSE (fetch (REGION LEFT) of REGION)
                                                                        OLDWIDTH NEWWIDTH))
                                                         (replace (REGION LEFT) of (\FM.ITEMPROP ITEM 'MAXREGION)
                                                            with (fetch (REGION LEFT) of REGION])
```

(\FM.TRANSPOSEVERT

```
  [LAMBDA (WINDOW OLDHEIGHT NEWHEIGHT)                                   (* jow "12-Apr-86 18:27")
                                                                         (* transpose bottom point)
    (for ITEM REGION in (WINDOWPROP WINDOW 'FM.ITEMS) do (SETQ REGION (\FM.ITEMPROP ITEM 'REGION))
                                                              (replace (REGION BOTTOM) of REGION
                                                                 with (\FM.TRANSPOSE (fetch (REGION BOTTOM) of REGION)
                                                                             OLDHEIGHT NEWHEIGHT))
                                                              (replace (REGION BOTTOM) of (\FM.ITEMPROP ITEM
                                                                                                'MAXREGION)
                                                                 with (fetch (REGION BOTTOM) of REGION])
```

(\FM.UPDATEGROUPEXTENT

```
  [LAMBDA (GROUPS GROUPLIST)                                          (* jow "12-Apr-86 18:28")

                (* THIS DEPENDS ON THE ALIST/PROPLIST GROUP STRUCTURE.
                GROUPS is a freemenu group alist structure. GROUPLIST is a list of group id's to update the extent of.
                If GROUPLIST is NIL, then update top group.)

    [OR GROUPLIST (SETQ GROUPLIST (LIST (\FM.DTOPGROUPID GROUPS]
    (LET (GROUP REGION DAUGHTERS BOXOFFSET)
         (for ID in GROUPLIST do (SETQ GROUP (CDR (ASSOC ID GROUPS)))
                                 [SETQ REGION (LISTPUT GROUP 'REGION (COPYALL (\FM.ITEMPROP
                                                                               (CAR (LISTGET GROUP 'ITEMS))
                                                                               'MAXREGION]
                                 [if (SETQ DAUGHTERS (LISTGET GROUP 'DAUGHTERS))
                                     then                          (* update subgroups first)
                                          (\FM.UPDATEGROUPEXTENT GROUPS DAUGHTERS)
                                          (for SUBID in DAUGHTERS
                                             do (EXTENDREGION REGION (LISTGET (CDR (ASSOC SUBID GROUPS))
                                                                              'REGION]
                                 [for ITEM in (LISTGET GROUP 'ITEMS) do (EXTENDREGION REGION (\FM.ITEMPROP
                                                                                              ITEM
                                                                                              'MAXREGION]
                                 (if (LISTGET GROUP 'BOX)
                                     then [SETQ BOXOFFSET (IPLUS (LISTGET GROUP 'BOX)
                                                                 (LISTGET GROUP 'BOXSPACE]
                                          (add (fetch (REGION LEFT) of REGION)
                                               (MINUS BOXOFFSET))
                                          (add (fetch (REGION BOTTOM) of REGION)
                                               (MINUS BOXOFFSET))
                                          (add (fetch (REGION WIDTH) of REGION)
                                               (IPLUS BOXOFFSET BOXOFFSET))
                                          (add (fetch (REGION HEIGHT) of REGION)
                                               (IPLUS BOXOFFSET BOXOFFSET])
```

(\FM.WINDOWEXTENT

```
  [LAMBDA (WINDOW)                                                    (* jow "24-Apr-86 17:13")

                (* start with the top groups extent, assumed to be correct, and then extent to account for any infinite width items.
                return extended extent)

    (LET ([EXTENT (COPY (\FM.TOPGROUPPROP WINDOW 'REGION]
          REGION)
         (for ITEM in (WINDOWPROP WINDOW 'FM.ITEMS) when (\FM.ITEMPROP ITEM 'INFINITEWIDTH)
            do (SETQ REGION (\FM.ITEMPROP ITEM 'REGION))
               [replace (REGION WIDTH) of REGION with (\FM.ITEMWIDTH (\FM.ITEMPROP ITEM 'LABEL)
                                                                     (\FM.ITEMPROP ITEM 'FONT]
               (EXTENDREGION EXTENT REGION))
        EXTENT])
```

(\FM.UPDATEWINDOWEXTENT

```
  [LAMBDA (WINDOW)                                                    (* jow "25-Apr-86 11:38")

                (* CURRENTLY NEVER CALLED, BECAUSE PROBLEMS WITH RECALCULATING MINWIDTH, MINHEIGHT, BECAUSE
                THIS ALGORITHM JUST KEEPS ON ADDING.)

                (* update the window's extent to the menu's region. If the extent is not entirely visible, then menu has grown.
                Update MIN dimensions of menu to allow getting the entire menu visible again.)

    (WINDOWPROP WINDOW 'EXTENT (\FM.WINDOWEXTENT WINDOW))
    (LET [(EXTENT (WINDOWPROP WINDOW 'EXTENT]
         [if (IGREATERP (fetch (REGION WIDTH) of EXTENT)
                    (WINDOWPROP WINDOW 'WIDTH))
             then (WINDOWPROP WINDOW 'FM.MINWIDTH (IPLUS (WINDOWPROP WINDOW 'FM.MINWIDTH)
                                                         (IDIFFERENCE (fetch (REGION WIDTH) of EXTENT)
                                                                (WINDOWPROP WINDOW 'WIDTH]
         (if (IGREATERP (fetch (REGION HEIGHT) of EXTENT)
                    (WINDOWPROP WINDOW 'HEIGHT))
             then (WINDOWPROP WINDOW 'FM.MINHEIGHT (IPLUS (WINDOWPROP WINDOW 'FM.MINHEIGHT)
                                                          (IDIFFERENCE (fetch (REGION HEIGHT) of EXTENT)
                                                                 (WINDOWPROP WINDOW 'HEIGHT])
)
```

;; MOUSE FUNCTIONS

```
(DECLARE%: DONTCOPY
(DECLARE%: EVAL@COMPILE

(PUTPROPS \FM.ONITEM MACRO [(REGION X Y INFINITWIDTH)
                            (AND (IGEQ Y (fetch (REGION BOTTOM) of REGION))
                                 (IGEQ X (fetch (REGION LEFT) of REGION))
                                 [OR INFINITWIDTH (ILESSP X (IPLUS (fetch (REGION LEFT) of REGION)
                                                                   (fetch (REGION WIDTH) of REGION]
                                 (ILESSP Y (IPLUS (fetch (REGION BOTTOM) of REGION)
                                                  (fetch (REGION HEIGHT) of REGION])

(PUTPROPS \FM.CHECKREGION MACRO [(WINDOW X Y)
                                  (for (ITEM REGION) in (WINDOWPROP WINDOW 'FM.ITEMS)
                                     eachtime (SETQ REGION (\FM.ITEMPROP ITEM 'REGION))
                                     thereis (\FM.ONITEM REGION X Y (\FM.ITEMPROP ITEM 'INFINITEWIDTH])
)
)

(DEFINEQ

(\FM.WINDOWENTRYFN
  [LAMBDA (WINDOW)                                              (* jow "20-Oct-86 10:51")

;;; THIS SHOULD NEVER GET CALLED NOW, BECAUSE FREEMENU DUMPS THE EDIT WHEN IT LOSES THE TTY.

          (* called when buttonevent occurs while editing with the tty process somewhere else.
          should give the tty back to freemenu unless the event is right only and not on an item.
          in that case, just do the window command menu. don't worry here about calling buttoneventfn's, because once freemenu
          gets the tty back, the edit button handler will notice the event and act properly.)

    (if [AND (LASTMOUSESTATE (ONLY RIGHT))
             (NOT (AND (INSIDEP (DSPCLIPPINGREGION NIL WINDOW)
                                (LASTMOUSEX WINDOW)
                                (LASTMOUSEY WINDOW))
                       (\FM.CHECKREGION WINDOW (LASTMOUSEX WINDOW)
                                (LASTMOUSEY WINDOW]
        then (DOWINDOWCOM WINDOW)
      else (TTY.PROCESS (WINDOWPROP WINDOW 'PROCESS])

(\FM.BUTTONEVENTFN
  [LAMBDA (WINDOW)                                              (* jow "13-Nov-85 22:08")
    (TOTOPW WINDOW)
    (if (AND (NOT (WINDOWPROP WINDOW 'FM.BUSY))
             (LASTMOUSESTATE (NOT UP)))
        then                                                    (* ignore button up events and events when menu is busy)
             (\FM.MENUHANDLER WINDOW])

(\FM.RIGHTBUTTONFN
  [LAMBDA (WINDOW)                                              (* jow "10-Apr-86 22:38")

          (* if on an item, and not busy, then process the item selection, else do the window command menu.)

    (TOTOPW WINDOW)
    (if (AND (INSIDEP (DSPCLIPPINGREGION NIL WINDOW)
                    (LASTMOUSEX WINDOW)
                    (LASTMOUSEY WINDOW))
             (\FM.CHECKREGION WINDOW (LASTMOUSEX WINDOW)
                    (LASTMOUSEY WINDOW)))
        then                                                    (* valid item selected)
             (if (NOT (WINDOWPROP WINDOW 'FM.BUSY))
                 then (\FM.MENUHANDLER WINDOW))
      else                                                      (* not on item)
           (DOWINDOWCOM WINDOW])

(\FM.DOSELECTION
  [LAMBDA (ITEM WINDOW BUTTONS)                                 (* jow "17-Oct-86 17:06")

;;; run the selectedfns for this ITEM.  set busy flag accordingly.

    (RESETLST
        (RESETSAVE NIL (LIST 'WINDOWPROP WINDOW 'FM.BUSY NIL))
        (WINDOWPROP WINDOW 'FM.BUSY T)
        (APPLY* (\FM.ITEMPROP ITEM 'SYSSELECTEDFN)
               ITEM WINDOW BUTTONS)
        (BLOCK)
        (APPLY* (\FM.ITEMPROP ITEM 'SELECTEDFN)
               ITEM WINDOW BUTTONS)

        ;; return NIL so that result of process can't point to itself.

        NIL)])

(\FM.MENUHANDLER
  [LAMBDA (WINDOW SAMEPROCESS)                                  (* jow "20-Oct-86 10:48")
    (repeatuntil (MOUSESTATE UP) bind (TIMER _ (SETUPTIMER 0))
                                  ITEM LASTITEM BUTTONS PROMPTFLG
       do (SETQ BUTTONS (DECODEBUTTONS))
          (SETQ LASTITEM ITEM)
          (SETQ ITEM (\FM.CHECKREGION WINDOW (LASTMOUSEX WINDOW)
                            (LASTMOUSEY WINDOW)))
          (if ITEM
              then (COND
                       ((NOT LASTITEM)                           (* moved on new item)
                        (APPLY* (\FM.ITEMPROP ITEM 'SYSDOWNFN)
                               ITEM WINDOW BUTTONS)
                        (APPLY* (\FM.ITEMPROP ITEM 'DOWNFN)
                               ITEM WINDOW BUTTONS)
                        (SETUPTIMER MENUHELDWAIT TIMER)
                        (SETQ PROMPTFLG T))
                       ((NEQ LASTITEM ITEM)                      (* jump between items without dead interval.
                                                                 call last mouseoff, and new mousedown)
                        (APPLY* (\FM.ITEMPROP LASTITEM 'SYSMOVEDFN)
                               LASTITEM WINDOW BUTTONS)
                        (APPLY* (\FM.ITEMPROP LASTITEM 'MOVEDFN)
                               LASTITEM WINDOW BUTTONS)
                        (APPLY* (\FM.ITEMPROP ITEM 'SYSDOWNFN)
                               ITEM WINDOW BUTTONS)
                        (APPLY* (\FM.ITEMPROP ITEM 'DOWNFN)
                               ITEM WINDOW BUTTONS)
                        (SETUPTIMER MENUHELDWAIT TIMER)
                        (SETQ PROMPTFLG T))
                       ((AND PROMPTFLG (TIMEREXPIRED? TIMER))    (* held on item long enough)
                        (PRINTOUT (OR (WINDOWPROP WINDOW 'FM.PROMPTWINDOW)
                                      PROMPTWINDOW)
                               T
                               (if (STRINGP (\FM.ITEMPROP ITEM 'MESSAGE))
                                   then (\FM.ITEMPROP ITEM 'MESSAGE)
                                 else (APPLY* (\FM.ITEMPROP ITEM 'MESSAGE)
                                             ITEM WINDOW BUTTONS)))
                        (SETQ PROMPTFLG NIL)))
                   (APPLY* (\FM.ITEMPROP ITEM 'HELDFN)
                          ITEM WINDOW BUTTONS)
            elseif LASTITEM
              then                                               (* moved off item)
                   (APPLY* (\FM.ITEMPROP LASTITEM 'SYSMOVEDFN)
                          LASTITEM WINDOW BUTTONS)
                   (APPLY* (\FM.ITEMPROP LASTITEM 'MOVEDFN)
                          LASTITEM WINDOW BUTTONS))
       finally (SETQ LASTITEM ITEM)
             (SETQ ITEM (\FM.CHECKREGION WINDOW (LASTMOUSEX WINDOW)
                               (LASTMOUSEY WINDOW)))
             (if LASTITEM
                 then (COND
                          ((NEQ LASTITEM ITEM)                   (* moved off item)
                           (APPLY* (\FM.ITEMPROP LASTITEM 'SYSMOVEDFN)
                                  LASTITEM WINDOW BUTTONS)
                           (APPLY* (\FM.ITEMPROP LASTITEM 'MOVEDFN)
                                  LASTITEM WINDOW BUTTONS))
                          (ITEM (if SAMEPROCESS
                                    then (\FM.DOSELECTION ITEM WINDOW BUTTONS)
                                  else (ADD.PROCESS `(\FM.DOSELECTION ',ITEM ',WINDOW ',BUTTONS)
                                              'NAME
                                              'FREEMENU
                                              'FREEMENU.PROCESS T])

)

;; ITEM SUPPORT FUNCTIONS

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(PUTPROPS \FM.DISPLAYBITMAP MACRO [(ITEM BITMAP WINDOW)          ; take care of background shade and display the item
                                   (IF (OPENWP WINDOW)
                                       THEN (BLTSHADE (\FM.ITEMPROP ITEM 'BACKGROUND)
                                                   WINDOW NIL NIL NIL NIL NIL (\FM.ITEMPROP ITEM 'MAXREGION))
                                            (BITBLT BITMAP 0 0 WINDOW (fetch (REGION LEFT)
                                                                         of (\FM.ITEMPROP ITEM 'REGION))
                                                   (fetch (REGION BOTTOM) of (\FM.ITEMPROP ITEM 'REGION))
                                                   NIL NIL NIL 'PAINT])

(PUTPROPS \FM.COERCEITEMPTR MACRO [(ITEMPTR WINDOW ITEM)
                                   (LET (GROUPID ITEMID)
                                        [COND
                                           ((LISTP ITEMPTR)     (* pull apart)
                                            (SETQ GROUPID (CAR ITEMPTR))
                                            (SETQ ITEMID (CADR ITEMPTR]
                                        [COND
                                           ((EQ \FM.GROUPSPEC GROUPID)
                                                                 (* DOES NOT TYPE CHECK ITEM.
                                                                 IF USED ITEM MUST BE A FREEMENUITEM.)
```

```
                                                (SETQ GROUPID (\FM.ITEMPROP ITEM 'GROUPID]
                                            (OR (FM.GETITEM (OR ITEMID ITEMPTR)
                                                       GROUPID WINDOW)
                                                (ERROR "Could not find item:" ITEMPTR])
)
)

(DEFINEQ

(\FM.GETITEMPROP
  [LAMBDA (ITEM PROP)                                                   (* jow "11-Apr-86 11:40")
                                                                        (* BACKGROUND (fetch (FREEMENUITEM
                                                                        FM.BACKGROUND) of ITEM))
                                                                        (* ATTACHPOINT (fetch (FREEMENUITEM
                                                                        FM.ATTACHPOINT) of ITEM))
    (SELECTQ PROP
        (TYPE (fetch (FREEMENUITEM FM.TYPE) of ITEM))
        (LABEL (fetch (FREEMENUITEM FM.LABEL) of ITEM))
        (ID (fetch (FREEMENUITEM FM.ID) of ITEM))
        (GROUPID (fetch (FREEMENUITEM FM.GROUPID) of ITEM))
        (STATE (fetch (FREEMENUITEM FM.STATE) of ITEM))
        (INITSTATE (fetch (FREEMENUITEM FM.INITSTATE) of ITEM))
        (FONT (fetch (FREEMENUITEM FM.FONT) of ITEM))
        (BITMAP (fetch (FREEMENUITEM FM.BITMAP) of ITEM))
        (HIGHLIGHT (fetch (FREEMENUITEM FM.HIGHLIGHT) of ITEM))
        (REGION (fetch (FREEMENUITEM FM.REGION) of ITEM))
        (MAXREGION (fetch (FREEMENUITEM FM.MAXREGION) of ITEM))
        (MESSAGE (fetch (FREEMENUITEM FM.MESSAGE) of ITEM))
        (USERDATA (fetch (FREEMENUITEM FM.USERDATA) of ITEM))
        (LINKS (fetch (FREEMENUITEM FM.LINKS) of ITEM))
        (SYSDOWNFN (fetch (FREEMENUITEM FM.SYSDOWNFN) of ITEM))
        (SYSMOVEDFN (fetch (FREEMENUITEM FM.SYSMOVEDFN) of ITEM))
        (SYSSELECTEDFN
             (fetch (FREEMENUITEM FM.SYSSELECTEDFN) of ITEM))
        (DOWNFN (fetch (FREEMENUITEM FM.DOWNFN) of ITEM))
        (HELDFN (fetch (FREEMENUITEM FM.HELDFN) of ITEM))
        (MOVEDFN (fetch (FREEMENUITEM FM.MOVEDFN) of ITEM))
        (SELECTEDFN (fetch (FREEMENUITEM FM.SELECTEDFN) of ITEM))
        (LISTGET (fetch (FREEMENUITEM FM.USERDATA) of ITEM)
               PROP])

(\FM.PUTITEMPROP
  [LAMBDA (ITEM PROP VALUE)                                             (* jow "11-Apr-86 11:41")
                                                                        (* store new value in item field)
                                                                        (* BACKGROUND (PROG1 (fetch
                                                                        (FREEMENUITEM FM.BACKGROUND) of ITEM)
                                                                        (replace (FREEMENUITEM FM.BACKGROUND) of ITEM with
                                                                        VALUE)))
                                                                        (* ATTACHPOINT (PROG1 (fetch
                                                                        (FREEMENUITEM FM.ATTACHPOINT) of ITEM)
                                                                        (replace (FREEMENUITEM FM.ATTACHPOINT) of ITEM with
                                                                        VALUE)))
    (SELECTQ PROP
        (TYPE (ERROR "Can't change the TYPE of an item" VALUE))
        (LABEL (PROG1 (fetch (FREEMENUITEM FM.LABEL) of ITEM)
                   (replace (FREEMENUITEM FM.LABEL) of ITEM with VALUE)))
        (ID (PROG1 (fetch (FREEMENUITEM FM.ID) of ITEM)
                (replace (FREEMENUITEM FM.ID) of ITEM with VALUE)))
        (GROUPID (fetch (FREEMENUITEM FM.GROUPID) of ITEM)
                 (replace (FREEMENUITEM FM.GROUPID) of ITEM with VALUE))
        (STATE (PROG1 (fetch (FREEMENUITEM FM.STATE) of ITEM)
                   (replace (FREEMENUITEM FM.STATE) of ITEM with VALUE)))
        (INITSTATE (PROG1 (fetch (FREEMENUITEM FM.INITSTATE) of ITEM)
                       (replace (FREEMENUITEM FM.INITSTATE) of ITEM with VALUE)))
        (FONT (PROG1 (fetch (FREEMENUITEM FM.FONT) of ITEM)
                  (replace (FREEMENUITEM FM.FONT) of ITEM with VALUE)))
        (BITMAP (PROG1 (fetch (FREEMENUITEM FM.BITMAP) of ITEM)
                    (replace (FREEMENUITEM FM.BITMAP) of ITEM with VALUE)))
        (HIGHLIGHT (PROG1 (fetch (FREEMENUITEM FM.HIGHLIGHT) of ITEM)
                       (replace (FREEMENUITEM FM.HIGHLIGHT) of ITEM with VALUE)))
        (REGION (PROG1 (fetch (FREEMENUITEM FM.REGION) of ITEM)
                    (replace (FREEMENUITEM FM.REGION) of ITEM with VALUE)))
        (MAXREGION (PROG1 (fetch (FREEMENUITEM FM.MAXREGION) of ITEM)
                       (replace (FREEMENUITEM FM.MAXREGION) of ITEM with VALUE)))
        (MESSAGE (PROG1 (fetch (FREEMENUITEM FM.MESSAGE) of ITEM)
                     (replace (FREEMENUITEM FM.MESSAGE) of ITEM with VALUE)))
        (USERDATA (ERROR "Can't change the USERDATA of an item" VALUE))
        (LINKS (PROG1 (fetch (FREEMENUITEM FM.LINKS) of ITEM)
                   (replace (FREEMENUITEM FM.LINKS) of ITEM with VALUE)))
        (SYSDOWNFN (PROG1 (fetch (FREEMENUITEM FM.SYSDOWNFN) of ITEM)
                       (replace (FREEMENUITEM FM.SYSDOWNFN) of ITEM with VALUE)))
        (SYSMOVEDFN (PROG1 (fetch (FREEMENUITEM FM.SYSMOVEDFN) of ITEM)
                        (replace (FREEMENUITEM FM.SYSMOVEDFN) of ITEM with VALUE)))
        (SYSSELECTEDFN
             (PROG1 (fetch (FREEMENUITEM FM.SYSSELECTEDFN) of ITEM)
                 (replace (FREEMENUITEM FM.SYSSELECTEDFN) of ITEM with VALUE)))
        (DOWNFN (PROG1 (fetch (FREEMENUITEM FM.DOWNFN) of ITEM)
                    (replace (FREEMENUITEM FM.DOWNFN) of ITEM with VALUE)))
        (HELDFN (PROG1 (fetch (FREEMENUITEM FM.HELDFN) of ITEM)
                    (replace (FREEMENUITEM FM.HELDFN) of ITEM with VALUE)))
        (MOVEDFN (PROG1 (fetch (FREEMENUITEM FM.MOVEDFN) of ITEM)
                     (replace (FREEMENUITEM FM.MOVEDFN) of ITEM with VALUE)))
        (SELECTEDFN (PROG1 (fetch (FREEMENUITEM FM.SELECTEDFN) of ITEM)
                        (replace (FREEMENUITEM FM.SELECTEDFN) of ITEM with VALUE)))
        (PROG1 (LISTGET (fetch (FREEMENUITEM FM.USERDATA) of ITEM)
                      PROP)
            (LISTPUT (fetch (FREEMENUITEM FM.USERDATA) of ITEM)
                   PROP VALUE))])
```

(\FM.CGETITEMPROP

```
  [LAMBDA (ITEM PROP)                                          (* jow "12-Apr-86 16:13")
```

(* macro dispatch function for FM.ITEMPROP. ITEM is bound to the name of the item to be visited, and PROP is bound to the expression (QUOTE <FIELDNAME>)%. This function returns the appropriate fetchfield expression to be compiled. IF THE FREEMENUITEM RECORD IS CHANGED,THIS FUNCTION MUST BE CHANGED ACCORDINGLY)

(* BACKGROUND (BQUOTE (FETCH (FREEMENUITEM FM.BACKGROUND) OF (\FM.INSUREFM %, ITEM))))

(* ATTACHPOINT (BQUOTE (FETCH (FREEMENUITEM FM.ATTACHPOINT) OF (\FM.INSUREFM %, ITEM))))

```
    (SELECTQ (CADR PROP)
        (TYPE `(FETCH (FREEMENUITEM FM.TYPE) OF %, ITEM))
        (LABEL `(FETCH (FREEMENUITEM FM.LABEL) OF %, ITEM))
        (ID `(FETCH (FREEMENUITEM FM.ID) OF %, ITEM))
        (GROUPID `(FETCH (FREEMENUITEM FM.GROUPID) OF %, ITEM))
        (STATE `(FETCH (FREEMENUITEM FM.STATE) OF %, ITEM))
        (INITSTATE `(FETCH (FREEMENUITEM FM.INITSTATE) OF %, ITEM))
        (FONT `(FETCH (FREEMENUITEM FM.FONT) OF %, ITEM))
        (BITMAP `(FETCH (FREEMENUITEM FM.BITMAP) OF %, ITEM))
        (HIGHLIGHT `(FETCH (FREEMENUITEM FM.HIGHLIGHT) OF %, ITEM))
        (REGION `(FETCH (FREEMENUITEM FM.REGION) OF %, ITEM))
        (MAXREGION `(FETCH (FREEMENUITEM FM.MAXREGION) OF %, ITEM))
        (MESSAGE `(FETCH (FREEMENUITEM FM.MESSAGE) OF %, ITEM))
        (USERDATA `(FETCH (FREEMENUITEM FM.USERDATA) OF %, ITEM))
        (LINKS `(FETCH (FREEMENUITEM FM.LINKS) OF %, ITEM))
        (SYSDOWNFN `(FETCH (FREEMENUITEM FM.SYSDOWNFN) OF %, ITEM))
        (SYSMOVEDFN `(FETCH (FREEMENUITEM FM.SYSMOVEDFN) OF %, ITEM))
        (SYSSELECTEDFN
            `(FETCH (FREEMENUITEM FM.SYSSELECTEDFN) OF %, ITEM))
        (DOWNFN `(FETCH (FREEMENUITEM FM.DOWNFN) OF %, ITEM))
        (HELDFN `(FETCH (FREEMENUITEM FM.HELDFN) OF %, ITEM))
        (MOVEDFN `(FETCH (FREEMENUITEM FM.MOVEDFN) OF %, ITEM))
        (SELECTEDFN `(FETCH (FREEMENUITEM FM.SELECTEDFN) OF %, ITEM))
        `(LISTGET (FETCH (FREEMENUITEM FM.USERDATA) OF %, ITEM)
                (QUOTE %, (CADR PROP])
```

(\FM.CPUTITEMPROP

```
  [LAMBDA (ITEM PROP VALUE)                                    (* jow "12-Apr-86 16:10")
```

(* macro dispatch function for FM.ITEMPROP. ITEM is bound to the name of the item to be visited, PROP is bound to the expression ((QUOTE <FIELDNAME>)%, and VALUE is bound to the expression to be evaluated at run time to yield the newvalue.) This function returns the appropriate prog1 expression to be compiled, which will return the old value, and set the new value of an item prop. IF THE FREEMENUITEM RECORD IS CHANGED,THIS FUNCTION MUST BE CHANGED ACCORDINGLY)

(* BACKGROUND (BQUOTE (PROG1 (FETCH (FREEMENUITEM FM.BACKGROUND) OF (\FM.INSUREFM %, ITEM)) (REPLACE (FREEMENUITEM FM.BACKGROUND) OF %, ITEM WITH %, VALUE))))

(* ATTACHPOINT (BQUOTE (PROG1 (FETCH (FREEMENUITEM FM.ATTACHPOINT) OF (\FM.INSUREFM %, ITEM)) (REPLACE (FREEMENUITEM FM.ATTACHPOINT) OF %, ITEM WITH %, VALUE))))

```
    (SELECTQ (CADR PROP)
        (TYPE (ERROR "FreeMenuItem property TYPE not settable" (LIST 'FM.ITEMPROP ITEM PROP VALUE)))
        (LABEL `(PROG1 (FETCH (FREEMENUITEM FM.LABEL) OF %, ITEM)
                    (REPLACE (FREEMENUITEM FM.LABEL) OF %, ITEM WITH %, VALUE)))
        (ID `(PROG1 (FETCH (FREEMENUITEM FM.ID) OF %, ITEM)
                 (REPLACE (FREEMENUITEM FM.ID) OF %, ITEM WITH %, VALUE)))
        (GROUPID `(PROG1 (FETCH (FREEMENUITEM FM.GROUPID) OF %, ITEM)
                      (REPLACE (FREEMENUITEM FM.GROUPID) OF %, ITEM WITH %, VALUE)))
        (STATE `(PROG1 (FETCH (FREEMENUITEM FM.STATE) OF %, ITEM)
                    (REPLACE (FREEMENUITEM FM.STATE) OF %, ITEM WITH %, VALUE)))
        (INITSTATE `(PROG1 (FETCH (FREEMENUITEM FM.INITSTATE) OF %, ITEM)
                        (REPLACE (FREEMENUITEM FM.INITSTATE) OF %, ITEM WITH %, VALUE)))
        (FONT `(PROG1 (FETCH (FREEMENUITEM FM.FONT) OF %, ITEM)
                   (REPLACE (FREEMENUITEM FM.FONT) OF %, ITEM WITH %, VALUE)))
        (BITMAP `(PROG1 (FETCH (FREEMENUITEM FM.BITMAP) OF %, ITEM)
                     (REPLACE (FREEMENUITEM FM.BITMAP) OF %, ITEM WITH %, VALUE)))
        (HIGHLIGHT `(PROG1 (FETCH (FREEMENUITEM FM.HIGHLIGHT) OF %, ITEM)
                         (REPLACE (FREEMENUITEM FM.HIGHLIGHT) OF %, ITEM WITH %, VALUE)))
        (REGION `(PROG1 (FETCH (FREEMENUITEM FM.REGION) OF %, ITEM)
                      (REPLACE (FREEMENUITEM FM.REGION) OF %, ITEM WITH %, VALUE)))
        (MAXREGION `(PROG1 (FETCH (FREEMENUITEM FM.MAXREGION) OF %, ITEM)
                         (REPLACE (FREEMENUITEM FM.MAXREGION) OF %, ITEM WITH %, VALUE)))
        (MESSAGE `(PROG1 (FETCH (FREEMENUITEM FM.MESSAGE) OF %, ITEM)
                       (REPLACE (FREEMENUITEM FM.MESSAGE) OF %, ITEM WITH %, VALUE)))
        (USERDATA (ERROR "FreeMenuItem property USERDATA not settable" (LIST 'FM.ITEMPROP ITEM PROP VALUE)))
        (LINKS `(PROG1 (FETCH (FREEMENUITEM FM.LINKS) OF %, ITEM)
                     (REPLACE (FREEMENUITEM FM.LINKS) OF %, ITEM WITH %, VALUE)))
        (SYSDOWNFN `(PROG1 (FETCH (FREEMENUITEM FM.SYSDOWNFN) OF %, ITEM)
                         (REPLACE (FREEMENUITEM FM.SYSDOWNFN) OF %, ITEM WITH %, VALUE)))
        (SYSMOVEDFN `(PROG1 (FETCH (FREEMENUITEM FM.SYSMOVEDFN) OF %, ITEM)
                          (REPLACE (FREEMENUITEM FM.SYSMOVEDFN) OF %, ITEM WITH %, VALUE)))
        (SYSSELECTEDFN
            `(PROG1 (FETCH (FREEMENUITEM FM.SYSSELECTEDFN) OF %, ITEM)
                 (REPLACE (FREEMENUITEM FM.SYSSELECTEDFN) OF %, ITEM WITH %, VALUE)))
        (DOWNFN `(PROG1 (FETCH (FREEMENUITEM FM.DOWNFN) OF %, ITEM)
                      (REPLACE (FREEMENUITEM FM.DOWNFN) OF %, ITEM WITH %, VALUE)))
        (HELDFN `(PROG1 (FETCH (FREEMENUITEM FM.HELDFN) OF %, ITEM)
                      (REPLACE (FREEMENUITEM FM.HELDFN) OF %, ITEM WITH %, VALUE)))
        (MOVEDFN `(PROG1 (FETCH (FREEMENUITEM FM.MOVEDFN) OF %, ITEM)
                       (REPLACE (FREEMENUITEM FM.MOVEDFN) OF %, ITEM WITH %, VALUE)))
        (SELECTEDFN `(PROG1 (FETCH (FREEMENUITEM FM.SELECTEDFN) OF %, ITEM)
                          (REPLACE (FREEMENUITEM FM.SELECTEDFN) OF %, ITEM WITH %, VALUE)))
        `(PROG1 (LISTGET (FETCH (FREEMENUITEM FM.USERDATA) OF %, ITEM)
                       (QUOTE %, (CADR PROP)))
             (LISTPUT (FETCH (FREEMENUITEM FM.USERDATA) OF %, ITEM)
                    (QUOTE %, (CADR PROP))
                    %, VALUE))])
```

(\FM.DISPLAYITEM

```
  [LAMBDA (ITEM WINDOW)                                         (* jow "26-Jun-86 14:52")
    (\FM.DISPLAYBITMAP ITEM (\FM.ITEMPROP ITEM 'BITMAP)
           WINDOW])
```

(\FM.HIGHLIGHTITEM

```
  [LAMBDA (ITEM WINDOW BUTTONS)                                 (* jow "26-Jun-86 14:52")
    (\FM.DISPLAYBITMAP ITEM (\FM.ITEMPROP ITEM 'HIGHLIGHT)
           WINDOW])
```

(\FM.CHANGELABEL

```
  [LAMBDA (ITEM NEWLABEL)                                       ; Edited 28-Dec-87 17:03 by woz

    ;; change the items label.  NEWDESC is a description of the new item.  This includes the items USERDATA list, which has in it all of the boxing info
    ;; necessary for changing the label.  Do not redisplay

    (OR (OR (ATOM NEWLABEL)
            (STRINGP NEWLABEL)
            (BITMAPP NEWLABEL))
        (ERROR "CHANGELABEL Error. NEWLABEL must be an atom, string, or bitmap." NEWLABEL))
    (LET ((FONT (\FM.ITEMPROP ITEM 'FONT))
          [LEFT (fetch (REGION LEFT) of (\FM.ITEMPROP ITEM 'REGION]
          [BOTTOM (fetch (REGION BOTTOM) of (\FM.ITEMPROP ITEM 'REGION]
          [NEWDESC (APPEND (LIST 'LABEL NEWLABEL)
                          (\FM.ITEMPROP ITEM 'USERDATA]
          REGIONS BITMAPS)
         (SETQ REGIONS (\FM.GETREGIONS NEWDESC LEFT BOTTOM FONT))
         (SETQ BITMAPS (\FM.GETBITMAPS NEWDESC FONT (CAR REGIONS)
                              (CADR REGIONS)))
         (\FM.ITEMPROP ITEM 'LABEL NEWLABEL)
         (\FM.ITEMPROP ITEM 'REGION (CAR REGIONS))
         (\FM.ITEMPROP ITEM 'MAXREGION (CADDR REGIONS))
         (\FM.ITEMPROP ITEM 'BITMAP (CAR BITMAPS))
         (\FM.ITEMPROP ITEM 'HIGHLIGHT (CADR BITMAPS))
         (SELECTQ (\FM.ITEMPROP ITEM 'TYPE)
             (EDIT                                              ; use maxregion always
                   (\FM.ITEMPROP ITEM 'REGION (\FM.ITEMPROP ITEM 'MAXREGION)))
             (NUMBER                                            ; make state a number
                     (\FM.ITEMPROP ITEM 'REGION (\FM.ITEMPROP ITEM 'MAXREGION))
                     (\FM.NUMBER-CHANGESTATE ITEM NEWLABEL))
             (TOGGLE                                            ; reset state bitmaps
                     (\FM.ITEMPROP ITEM 'UNHIGHLIGHT (\FM.ITEMPROP ITEM 'BITMAP)))
             (3STATE                                            ; reset state bitmaps
                     (\FM.ITEMPROP ITEM 'UNHIGHLIGHT (\FM.ITEMPROP ITEM 'BITMAP))
                     (\FM.3STATE-SETUPOFFBITMAP ITEM))
             (NWAY                                              ; reset state bitmaps
                   (\FM.ITEMPROP ITEM 'UNHIGHLIGHT (\FM.ITEMPROP ITEM 'BITMAP)))
             NIL])
```

(\FM.CHANGESTATE

```
  [LAMBDA (X NEWSTATE WINDOW)                                   ; Edited 28-Dec-87 17:08 by woz

    ;; user interface to change the state of any (state) item or nway collection.  Redisplay the item if the window is open
    (if (ASSOC X (WINDOWPROP WINDOW 'FM.NWAYS))
        then                                                     ; X specifies an NWAY.  Changestate and redisplay.
             (\FM.NWAY-CHANGESTATE X NEWSTATE WINDOW)
      else                                                       ; treat X as an item
           (SELECTQ (\FM.ITEMPROP X 'TYPE)
               (TOGGLE (\FM.TOGGLE-CHANGESTATE X NEWSTATE))
               (3STATE (\FM.3STATE-CHANGESTATE X NEWSTATE))
               (STATE (\FM.STATE-CHANGESTATE X NEWSTATE WINDOW))
               NIL])

(\FM.ENDEDIT
  [LAMBDA (WINDOW WAITFLG)                                       (* jow " 4-Nov-86 16:09")

;;; used as a closefn for freemenu, as well as for ending edits as necessary during button events.  Will kill the edit process and wait as requested.  If
;;; editing, the freemenu process must be the ttyprocess.

    (if (FM.EDITP WINDOW)
        then (\CARET.DOWN)
             (SETUPTIMER 0 (WINDOWPROP WINDOW 'FM.EDIT-TIMER))
             (LET ((FM.PROCESS (TTY.PROCESS)))
                  (if (PROCESSPROP FM.PROCESS 'FREEMENU.PROCESS)
                      then (if (NEQ (THIS.PROCESS)
                                    FM.PROCESS)
                               then (PROCESS.RESULT FM.PROCESS WAITFLG))
                    else (ERROR "Can't find freemenu process to end editing" FM.PROCESS])

(\FM.INSUREVISIBLE
  [LAMBDA (ITEM WINDOW)                                          (* jow "25-Apr-86 12:04")
    (if [NOT (SUBREGIONP (DSPCLIPPINGREGION NIL WINDOW)
                    (\FM.ITEMPROP ITEM 'REGION]
        then                                                     (* not all of ITEM is visible%: ensure that left of item is in
                                                                 window)
             (SCROLLW WINDOW [FQUOTIENT (fetch (REGION LEFT) of (\FM.ITEMPROP ITEM 'REGION))
                                    (fetch (REGION WIDTH) of (WINDOWPROP WINDOW 'EXTENT]
                    0])

(\FM.CLEARITEM
  [LAMBDA (ITEM WINDOW REGION)                                   ; Edited 28-Dec-87 16:50 by woz

    ;; clear an item in the window.  If INFINITEWIDTH, then clear to edge of window.  Don't change the item.  REGION defaults to items current region,
    ;; and may be passed as an arg, in order to clear an 'old' region for the item.

    (if (OPENWP WINDOW)
        then [OR REGION (SETQ REGION (\FM.ITEMPROP ITEM 'REGION]
             (if (\FM.ITEMPROP ITEM 'INFINITEWIDTH)
                 then (BLTSHADE (\FM.ITEMPROP ITEM 'BACKGROUND)
                             WINDOW
                             (fetch (REGION LEFT) of REGION)
                             (fetch (REGION BOTTOM) of REGION)
                             NIL
                             (fetch (REGION HEIGHT) of REGION))
               else (BLTSHADE (\FM.ITEMPROP ITEM 'BACKGROUND)
                           WINDOW NIL NIL NIL NIL NIL REGION])

)

;; MOMENTARY ITEM FUNCTIONS

(DEFINEQ

(\FM.MOMENTARY-SETUP
  [LAMBDA (ITEM)                                                 (* jow "17-Apr-86 18:16")
    (OR (\FM.ITEMPROP ITEM 'MESSAGE)
        (\FM.ITEMPROP ITEM 'MESSAGE "Will select this item when you release the button."))
    (\FM.ITEMPROP ITEM 'SYSDOWNFN '\FM.HIGHLIGHTITEM)
    (\FM.ITEMPROP ITEM 'SYSMOVEDFN '\FM.DISPLAYITEM)
    (\FM.ITEMPROP ITEM 'SYSSELECTEDFN (FUNCTION \FM.MOMENTARY-SELECTEDFN])

(\FM.MOMENTARY-SELECTEDFN
  [LAMBDA (ITEM WINDOW BUTTONS)                                  (* jow "19-Apr-86 22:00")

            (* setup unhighlighting on the way out by puttin in a resetsave. we know we got called from \fm.doselection, which
            RESETLISTs.)

    (RESETSAVE NIL (LIST '\FM.DISPLAYITEM ITEM WINDOW])

)

;; TOGGLE ITEM FUNCTIONS

(DEFINEQ

(\FM.TOGGLE-SETUP
  [LAMBDA (ITEM REGIONS)                                           (* jow "18-Apr-86 12:22")
                                                                   (* toggle items initial state NIL)
    (OR (\FM.ITEMPROP ITEM 'MESSAGE)
        (\FM.ITEMPROP ITEM 'MESSAGE "Will toggle this item when you release the button."))
    (\FM.ITEMPROP ITEM 'SYSDOWNFN (FUNCTION \FM.TOGGLE-DOWNFN))
    (\FM.ITEMPROP ITEM 'SYSMOVEDFN (FUNCTION \FM.DISPLAYITEM))
    (\FM.ITEMPROP ITEM 'SYSSELECTEDFN (FUNCTION \FM.TOGGLE-SELECTEDFN))
                                                                   (* save unhighlighted looks.)
    (\FM.ITEMPROP ITEM 'UNHIGHLIGHT (\FM.ITEMPROP ITEM 'BITMAP))  (* save regions for state changes.)
    (if [AND (CADR REGIONS)
             (NOT (EQUAL (CADR REGIONS)
                         (\FM.ITEMPROP ITEM 'REGION]
        then (\FM.ITEMPROP ITEM 'OFFREGION (\FM.ITEMPROP ITEM 'REGION))
             (\FM.ITEMPROP ITEM 'ONREGION (CADR REGIONS])
```

(\FM.TOGGLE-DOWNFN

```
  [LAMBDA (ITEM WINDOW BUTTONS)                                    (* jow "12-Apr-86 18:08")

         (* display the other state in the window. Can't just invert item in window, because "highlight" may be shade other than black.)

    (if (\FM.ITEMPROP ITEM 'STATE)
        then (\FM.DISPLAYBITMAP ITEM (\FM.ITEMPROP ITEM 'UNHIGHLIGHT)
                    WINDOW)
      else (\FM.DISPLAYBITMAP ITEM (\FM.ITEMPROP ITEM 'HIGHLIGHT)
                  WINDOW])
```

(\FM.TOGGLE-SELECTEDFN

```
  [LAMBDA (ITEM WINDOW BUTTONS)                                    (* jow "12-Apr-86 16:54")
                                                                   (* change item to new state. display already updated)
    (if (\FM.ITEMPROP ITEM 'STATE)
        then (\FM.TOGGLE-CHANGESTATE ITEM NIL)
      else (\FM.TOGGLE-CHANGESTATE ITEM T])
```

(\FM.TOGGLE-CHANGESTATE

```
  [LAMBDA (ITEM NEWSTATE)                                          (* jow "18-Apr-86 12:22")
    (\FM.ITEMPROP ITEM 'STATE NEWSTATE)
    (if NEWSTATE
        then (\FM.ITEMPROP ITEM 'BITMAP (\FM.ITEMPROP ITEM 'HIGHLIGHT))
             [AND (\FM.ITEMPROP ITEM 'ONREGION)
                  (\FM.ITEMPROP ITEM 'REGION (\FM.ITEMPROP ITEM 'ONREGION]
      else (\FM.ITEMPROP ITEM 'BITMAP (\FM.ITEMPROP ITEM 'UNHIGHLIGHT))
           (AND (\FM.ITEMPROP ITEM 'OFFREGION)
                (\FM.ITEMPROP ITEM 'REGION (\FM.ITEMPROP ITEM 'OFFREGION])
)
```

;; 3STATE ITEM FUNCTIONS

```
(DEFINEQ
```

(\FM.3STATE-SETUP

```
  [LAMBDA (ITEM REGIONS)                                           (* jow "18-Apr-86 14:40")
    (OR (\FM.ITEMPROP ITEM 'MESSAGE)
        (\FM.ITEMPROP ITEM 'MESSAGE "Will change item to this state when you release the button."))
    (\FM.ITEMPROP ITEM 'SYSDOWNFN (FUNCTION \FM.3STATE-DOWNFN))
    (\FM.ITEMPROP ITEM 'SYSMOVEDFN (FUNCTION \FM.DISPLAYITEM))
    (\FM.ITEMPROP ITEM 'SYSSELECTEDFN (FUNCTION \FM.3STATE-SELECTEDFN))
                                                                   (* save the unhighlighted bitmap.)
    (\FM.ITEMPROP ITEM 'UNHIGHLIGHT (\FM.ITEMPROP ITEM 'BITMAP))  (* save regions for state changes.)
    (if [AND (CADR REGIONS)
             (NOT (EQUAL (CADR REGIONS)
                         (\FM.ITEMPROP ITEM 'REGION]
        then (\FM.ITEMPROP ITEM 'NEUTRALREGION (\FM.ITEMPROP ITEM 'REGION))
             (\FM.ITEMPROP ITEM 'ONREGION (CADR REGIONS)))
    (\FM.3STATE-SETUPOFFBITMAP ITEM])
```

(\FM.3STATE-SETUPOFFBITMAP

```
  [LAMBDA (ITEM)                                                   (* jow "24-Apr-86 23:01")
                                                                   (* used by 3state items to setup bitmap with OFF looks.)
    (LET* ((OFF (\FM.ITEMPROP ITEM 'OFF))
           (BOX (OR (\FM.ITEMPROP ITEM 'BOX)
                    0))
           (FONT (\FM.ITEMPROP ITEM 'FONT))
           (OFFREGION (\FM.ITEMPROP ITEM 'REGION))
           ID OFFBITMAP)
          (COND
             ((OR (AND OFF (ATOM OFF)
                       (NOT (TEXTUREP OFF)))
                  (STRINGP OFF)
                  (BITMAPP OFF))                                   (* new label specified. make anew)

         (* Set REGION of OFF looks%: build item description that has OFF has its HIGHLIGHT prop.
         Then pass to GETREGIONS (so lie a bit) to get the region of the OFF looks.)
```

```
            [SETQ ID (COPY (\FM.ITEMPROP ITEM 'USERDATA]
            (LISTPUT ID 'HIGHLIGHT OFF)
            (SETQ OFFREGION (CADR (\FM.GETREGIONS ID (fetch (REGION LEFT) of OFFREGION)
                                         (fetch (REGION BOTTOM) of OFFREGION)
                                         FONT)))
            (SETQ OFFBITMAP (\FM.MAKEBITMAP OFF FONT (fetch (REGION WIDTH) of OFFREGION)
                                   (fetch (REGION HEIGHT) of OFFREGION)
                                   ID))
            (\FM.ITEMPROP ITEM 'HIGHLIGHT (LIST (\FM.ITEMPROP ITEM 'HIGHLIGHT)
                                                OFFBITMAP))
            (if [NOT (EQUAL OFFREGION (\FM.ITEMPROP ITEM 'REGION]
                then                                        (* different region for OFF looks. Save regions for changing
                                                            state)
                    (\FM.ITEMPROP ITEM 'NEUTRALREGION (\FM.ITEMPROP ITEM 'REGION))
                    (\FM.ITEMPROP ITEM 'OFFREGION OFFREGION)
                    (EXTENDREGION (\FM.ITEMPROP ITEM 'MAXREGION)
                           OFFREGION)))
           ((TEXTUREP OFF)                                  (* paint shade on label)
            [SETQ OFFBITMAP (BITMAPCOPY (\FM.ITEMPROP ITEM 'BITMAP]
            (BLTSHADE OFF OFFBITMAP BOX BOX (IDIFFERENCE (fetch (REGION WIDTH) of OFFREGION)
                                                   (ITIMES BOX 2))
                   (IDIFFERENCE (fetch (REGION HEIGHT) of OFFREGION)
                          (ITIMES BOX 2))
                   'PAINT)
            (\FM.ITEMPROP ITEM 'HIGHLIGHT (LIST (\FM.ITEMPROP ITEM 'HIGHLIGHT)
                                                OFFBITMAP)))
           (T                                               (* default%: draw slash on label)
              [SETQ OFFBITMAP (BITMAPCOPY (\FM.ITEMPROP ITEM 'BITMAP]
              (LET ((STREAM (DSPCREATE OFFBITMAP)))
                   (DRAWLINE 0 0 (SUB1 (fetch (REGION WIDTH) of OFFREGION))
                          (IDIFFERENCE (fetch (REGION HEIGHT) of OFFREGION)
                                 2)
                          2
                          'REPLACE STREAM)
                   (\FM.ITEMPROP ITEM 'HIGHLIGHT (LIST (\FM.ITEMPROP ITEM 'HIGHLIGHT)
                                                       OFFBITMAP])
```

(\FM.3STATE-DOWNFN

```
  [LAMBDA (ITEM WINDOW BUTTONS)                             (* jow "16-Apr-86 17:58")

           (* called when mouse down over 3state item. rotates the state of ITEM on the screen.
           The order is OFF -
           NIL -
           T)

    (SELECTQ (\FM.ITEMPROP ITEM 'STATE)
         (OFF                                               (* OFF to NIL)
              (\FM.DISPLAYBITMAP ITEM (\FM.ITEMPROP ITEM 'UNHIGHLIGHT)
                     WINDOW))
         (T                                                 (* T to OFF)
            (\FM.DISPLAYBITMAP ITEM (CADR (\FM.ITEMPROP ITEM 'HIGHLIGHT))
                   WINDOW))
         (NIL                                               (* NIL to T)
              (\FM.DISPLAYBITMAP ITEM (CAR (\FM.ITEMPROP ITEM 'HIGHLIGHT))
                     WINDOW))
         NIL])
```

(\FM.3STATE-SELECTEDFN

```
  [LAMBDA (ITEM WINDOW BUTTONS)                             (* jow "12-Apr-86 18:30")

           (* called when 3state item selected. rotates the state of ITEM and its bitmap.
           The order is OFF -
           NIL -
           T)

    (SELECTQ (\FM.ITEMPROP ITEM 'STATE)
         (OFF                                               (* OFF to NIL)
              (\FM.3STATE-CHANGESTATE ITEM NIL))
         (T                                                 (* T to OFF)
            (\FM.3STATE-CHANGESTATE ITEM 'OFF))
         (NIL                                               (* NIL to T)
              (\FM.3STATE-CHANGESTATE ITEM T))
         NIL])
```

(\FM.3STATE-CHANGESTATE

```
  [LAMBDA (ITEM NEWSTATE)                                   (* jow "18-Apr-86 15:19")
    (\FM.ITEMPROP ITEM 'STATE NEWSTATE)
    (SELECTQ NEWSTATE
         (OFF                                               (* to OFF)
              [\FM.ITEMPROP ITEM 'BITMAP (CADR (\FM.ITEMPROP ITEM 'HIGHLIGHT]
              [AND (\FM.ITEMPROP ITEM 'OFFREGION)
                   (\FM.ITEMPROP ITEM 'REGION (\FM.ITEMPROP ITEM 'OFFREGION])
         (T                                                 (* to T)
            [\FM.ITEMPROP ITEM 'BITMAP (CAR (\FM.ITEMPROP ITEM 'HIGHLIGHT]
            [AND (\FM.ITEMPROP ITEM 'ONREGION)
                 (\FM.ITEMPROP ITEM 'REGION (\FM.ITEMPROP ITEM 'ONREGION])
         (NIL                                                        (* to NIL)
              (\FM.ITEMPROP ITEM 'BITMAP (\FM.ITEMPROP ITEM 'UNHIGHLIGHT))
              [AND (\FM.ITEMPROP ITEM 'NEUTRALREGION)
                   (\FM.ITEMPROP ITEM 'REGION (\FM.ITEMPROP ITEM 'NEUTRALREGION])
         NIL])

)
```

;; STATE ITEM FUNCTIONS

```
(DEFINEQ

(\FM.STATE-SETUP
  [LAMBDA (ITEM)                                                     (* jow "28-Oct-86 18:55")

              (* The item's state is initialized to the first of the menu items. The subitems list is replaced with a menu of those items.)

    (OR (\FM.ITEMPROP ITEM 'MESSAGE)
        (\FM.ITEMPROP ITEM 'MESSAGE "Will let you select a value from a pop up menu."))
    (\FM.ITEMPROP ITEM 'SYSDOWNFN '\FM.HIGHLIGHTITEM)
    (\FM.ITEMPROP ITEM 'SYSMOVEDFN '\FM.DISPLAYITEM)
    (\FM.ITEMPROP ITEM 'SYSSELECTEDFN (FUNCTION \FM.STATE-SELECTEDFN))
    (if (\FM.ITEMPROP ITEM 'MENUITEMS)
        then                                                         (* build menu as specified)
             (LET [(MENU.ITEMS (\FM.ITEMPROP ITEM 'MENUITEMS))
                   (MENU.FONT (APPLY (FUNCTION FONTCREATE)
                                     (\FM.ITEMPROP ITEM 'MENUFONT]
                  (\FM.ITEMPROP ITEM 'STATE (OR (\FM.ITEMPROP ITEM 'INITSTATE)
                                                (CAR MENU.ITEMS)))
                  (\FM.ITEMPROP ITEM 'CHANGESTATE
                         (create MENU
                                 ITEMS _ MENU.ITEMS
                                 MENUFONT _ MENU.FONT
                                 CENTERFLG _ T
                                 TITLE _ (OR (\FM.ITEMPROP ITEM 'MENUTITLE)
                                             (\FM.ITEMPROP ITEM 'LABEL])

(\FM.STATE-SELECTEDFN
  [LAMBDA (ITEM WINDOW BUTTONS)                                      (* jow "12-Apr-86 18:30")

              (* Setup highlighting on the way out, to account for CHANGESTATE function and user selectedfn.
              If CHANGESTATE is an atom, treat as function name to be applied to ITEM WINDOW BUTTONS, which must return the
              new state (any atom, string, or bitmap) If CHANGESTATE is a menu, pop it up to select new state.
              If CHANGESTATE returns NIL, don't change state)

    (RESETSAVE NIL (LIST '\FM.DISPLAYITEM ITEM WINDOW))
    (LET [(NEWSTATE (COND
                       [(type? MENU (\FM.ITEMPROP ITEM 'CHANGESTATE))
                        (MENU (\FM.ITEMPROP ITEM 'CHANGESTATE]
                       ((\FM.ITEMPROP ITEM 'CHANGESTATE)
                        (APPLY* (\FM.ITEMPROP ITEM 'CHANGESTATE)
                               ITEM WINDOW BUTTONS]
         (if NEWSTATE
             then (\FM.STATE-CHANGESTATE ITEM NEWSTATE WINDOW])

(\FM.STATE-CHANGESTATE
  [LAMBDA (ITEM NEWSTATE WINDOW)                                     (* jow "12-Apr-86 18:31")

              (* changing the state of a STATE item simply changes the label of its display item.)

    (\FM.ITEMPROP ITEM 'STATE NEWSTATE)
    (LET [(DISPLAYITEM (LISTGET (\FM.ITEMPROP ITEM 'LINKS)
                              'DISPLAY]
         (if DISPLAYITEM
             then (FM.CHANGELABEL DISPLAYITEM NEWSTATE WINDOW])

)
```

;; NWAY ITEM FUNCTIONS

```
(DEFINEQ

(\FM.NWAY-SETUP
  [LAMBDA (ITEM REGIONS)                                             (* jow "24-Apr-86 21:53")
    (OR (\FM.ITEMPROP ITEM 'MESSAGE)
        (\FM.ITEMPROP ITEM 'MESSAGE (FUNCTION \FM.NWAY-MESSAGE)))
    (\FM.ITEMPROP ITEM 'SYSDOWNFN (FUNCTION \FM.NWAY-DOWNFN))
    (\FM.ITEMPROP ITEM 'SYSMOVEDFN (FUNCTION \FM.NWAY-MOVEDFN))
    (\FM.ITEMPROP ITEM 'SYSSELECTEDFN (FUNCTION \FM.NWAY-SELECTEDFN))
    (\FM.ITEMPROP ITEM 'UNHIGHLIGHT (\FM.ITEMPROP ITEM 'BITMAP)) (* save regions for state changes.)
    (if [AND (CADR REGIONS)
            (NOT (EQUAL (CADR REGIONS)
                        (\FM.ITEMPROP ITEM 'REGION]
        then (\FM.ITEMPROP ITEM 'OFFREGION (\FM.ITEMPROP ITEM 'REGION))
             (\FM.ITEMPROP ITEM 'ONREGION (CADR REGIONS])

(\FM.NWAY-MESSAGE
  [LAMBDA (ITEM WINDOW BUTTONS)                                (* jow "24-Apr-86 22:07")
    (IF (\FM.NWAYPROP WINDOW (\FM.ITEMPROP ITEM 'COLLECTION)
               'DESELECT)
        THEN (SELECTQ (CAR BUTTONS)
                  (RIGHT "Will turn off this NWAY collection.")
                  ((LEFT MIDDLE)
                      "Will select this item from its NWAY collection.")
                  NIL)
      ELSE "Will select this item from its NWAY collection."])

(\FM.NWAY-DOWNFN
  [LAMBDA (ITEM WINDOW BUTTONS)                                (* jow "12-Apr-86 18:16")
    (LET* [[NWAY (CDR (ASSOC (\FM.ITEMPROP ITEM 'COLLECTION)
                             (WINDOWPROP WINDOW 'FM.NWAYS]
           (STATE (LISTGET NWAY 'STATE]
          (if STATE
              then                                             (* an item is currently selected%: unhighlight it)
                   (\FM.DISPLAYBITMAP STATE (\FM.ITEMPROP STATE 'UNHIGHLIGHT)
                          WINDOW))
          (if [NOT (AND (EQ (CAR BUTTONS)
                            'RIGHT)
                        (LISTGET NWAY 'DESELECT]
              then                                             (* highlight this item unless deselect group.)
                   (\FM.DISPLAYBITMAP ITEM (\FM.ITEMPROP ITEM 'HIGHLIGHT)
                          WINDOW])

(\FM.NWAY-MOVEDFN
  [LAMBDA (ITEM WINDOW BUTTONS)                                (* jow "12-Apr-86 18:16")
    (LET* [[NWAY (CDR (ASSOC (\FM.ITEMPROP ITEM 'COLLECTION)
                             (WINDOWPROP WINDOW 'FM.NWAYS]
           (STATE (LISTGET NWAY 'STATE]
          (if STATE
              then                                             (* there is an item currently selected to redisplay)
                   (\FM.DISPLAYBITMAP STATE (\FM.ITEMPROP STATE 'BITMAP)
                          WINDOW))
          (if [NOT (AND (EQ (CAR BUTTONS)
                            'RIGHT)
                        (LISTGET NWAY 'DESELECT]
              then                                             (* this item was highlighted by downfn, so redisplay.)
                   (\FM.DISPLAYBITMAP ITEM (\FM.ITEMPROP ITEM 'BITMAP)
                          WINDOW])

(\FM.NWAY-SELECTEDFN
  [LAMBDA (ITEM WINDOW BUTTONS)                                (* jow "19-Apr-86 23:07")
    (if (AND (EQ (CAR BUTTONS)
                 'RIGHT)
             (\FM.NWAYPROP WINDOW (\FM.ITEMPROP ITEM 'COLLECTION)
                    'DESELECT))
        then                                                   (* group deselected)
             (\FM.NWAY-CHANGESTATE (\FM.ITEMPROP ITEM 'COLLECTION)
                    NIL WINDOW)
      else                                                     (* new item selected)
           (\FM.NWAY-CHANGESTATE (\FM.ITEMPROP ITEM 'COLLECTION)
                  ITEM WINDOW])

(\FM.NWAY-CHANGESTATE
  [LAMBDA (COLLECTION NEWSTATE WINDOW)                         (* jow "19-Apr-86 23:09")
    (LET [(STATE (\FM.NWAYPROP WINDOW COLLECTION 'STATE]
         (if (NEQ STATE NEWSTATE)
             then                                              (* actually have something to change)
                  (if STATE
                      then                                     (* STATE item is unselected)
                           (\FM.TOGGLE-CHANGESTATE STATE NIL))
                  (if NEWSTATE
                      then (\FM.TOGGLE-CHANGESTATE NEWSTATE T))
                  (\FM.NWAYPROP WINDOW COLLECTION 'STATE NEWSTATE])
)

;; NUMBER ITEM FUNCTIONS

(DEFINEQ

(\FM.NUMBER-SETUP
  [LAMBDA (ITEM)                                               ; Edited  5-Dec-94 15:48 by jds
```

```
     ;; This is EDIT-SETUP with number specifics added.
     (OR \FM.EDIT-TTBL (\FM.EDIT-SETUPTTBL))                          ; since have edit item, setup term table
     (\FM.ITEMPROP ITEM 'REGION (\FM.ITEMPROP ITEM 'MAXREGION))      ; always sensitive on maxregion
     [COND
        ([AND (\FM.ITEMPROP ITEM 'BOX)
              (NOT (\FM.ITEMPROP ITEM 'MAXWIDTH]                      ; boxing implies maxwidth
         (\FM.ITEMPROP ITEM 'MAXWIDTH (IDIFFERENCE (fetch (REGION WIDTH) of (\FM.ITEMPROP ITEM 'REGION))
                                            (ITIMES 2 (\FM.ITEMPROP ITEM 'BOXOFFSET]
     (COND
        [(\FM.ITEMPROP ITEM 'MAXWIDTH)                                ; setup stopwidth
         (\FM.ITEMPROP ITEM 'LABELMAXWIDTH (\FM.ITEMPROP ITEM 'MAXWIDTH]
        (T                                                            ; make item infinite
           (\FM.ITEMPROP ITEM 'INFINITEWIDTH T)))
     (OR (\FM.ITEMPROP ITEM 'MESSAGE)
         (\FM.ITEMPROP ITEM 'MESSAGE (FUNCTION \FM.NUMBER-MESSAGE)))
     (\FM.ITEMPROP ITEM 'INITSTATE (\FM.ITEMPROP ITEM 'LABEL))
     [COND
        [(FMEMB (\FM.ITEMPROP ITEM 'NUMBERTYPE)
                '(FLOAT FLOATP))
         (\FM.ITEMPROP ITEM 'SYSLIMITCHARS
                '(+ - 1 2 3 4 5 6 7 8 9 0 %.]
        (T (\FM.ITEMPROP ITEM 'SYSLIMITCHARS '(+ - 1 2 3 4 5 6 7 8 9 0]
     (\FM.ITEMPROP ITEM 'SYSSELECTEDFN (FUNCTION \FM.NUMBER-SELECTEDFN]))
```

(\FM.NUMBER-MESSAGE

```
  [LAMBDA (ITEM WINDOW BUTTONS)                                       (* jow "24-Apr-86 22:06")
     (SELECTQ (CAR BUTTONS)
         (RIGHT "Will clear this number, then start editing.")
         ((LEFT MIDDLE)
              "Will start editing this number at this position.")
         NIL])
```

(\FM.NUMBER-SELECTEDFN

```
  [LAMBDA (ITEM WINDOW BUTTONS)                                       (* jow "17-Oct-86 18:36")
     (\FM.EDIT-ITEM ITEM WINDOW BUTTONS NIL (FUNCTION \FM.NUMBER-CHANGESTATE])
```

(\FM.NUMBER-CHANGESTATE

```
  [LAMBDA (ITEM)                                                      ; Edited  6-Dec-94 10:02 by jds
     (\FM.ITEMPROP ITEM 'STATE (COND
                                  ([NOT (EQUAL "" (\FM.ITEMPROP ITEM 'LABEL]
                                   (NUMBERP (MKATOM (\FM.ITEMPROP ITEM 'LABEL])
)
```

;; TITLE ITEM FUNCTIONS

```
(DEFINEQ
```

(\FM.DISPLAY-SETUP

```
  [LAMBDA (ITEM)                                                      (* jow "17-Apr-86 18:17")
     (OR (\FM.ITEMPROP ITEM 'MESSAGE)
         (\FM.ITEMPROP ITEM 'MESSAGE ""])
)
```

;; EDITSTART ITEM FUNCTIONS

```
(DEFINEQ
```

(\FM.EDITSTART-SETUP

```
  [LAMBDA (ITEM)                                                      (* jow "24-Apr-86 22:00")
     (OR (\FM.ITEMPROP ITEM 'MESSAGE)
         (\FM.ITEMPROP ITEM 'MESSAGE (FUNCTION \FM.EDITSTART-MESSAGE)))
     (\FM.ITEMPROP ITEM 'SYSDOWNFN '\FM.HIGHLIGHTITEM)
     (\FM.ITEMPROP ITEM 'SYSMOVEDFN '\FM.DISPLAYITEM)
     (\FM.ITEMPROP ITEM 'SYSSELECTEDFN (FUNCTION \FM.EDITSTART-SELECTEDFN]))
```

(\FM.EDITSTART-MESSAGE

```
  [LAMBDA (ITEM WINDOW BUTTONS)                                       (* jow "24-Apr-86 22:04")
     (SELECTQ (CAR BUTTONS)
         (RIGHT "Will clear first, then start editing.")
         ((LEFT MIDDLE)
              "Will start editing.")
         NIL])
```

(\FM.EDITSTART-SELECTEDFN

```
  [LAMBDA (ITEM WINDOW BUTTONS)                                       ; Edited 28-Dec-87 17:28 by woz
                                                                      ; start editing at the beginning of item in the EDIT link.
     (\FM.DISPLAYITEM ITEM WINDOW)
     (LET [(EDITITEM (LISTGET (\FM.ITEMPROP ITEM 'LINKS)
                             'EDIT]
```

```
          (if (type? FREEMENUITEM EDITITEM)
              then (\FM.ITEMPROP ITEM 'SELECTEDFN (FUNCTION NILL))
                                                                         ; insure editstart item won't have selectedfn side effect, because
                                                                         ; end of edit is not well defined
                   (\FM.INSUREVISIBLE EDITITEM WINDOW)
                   (\FM.EDIT-ITEM EDITITEM WINDOW BUTTONS T (IF (EQ (\FM.ITEMPROP EDITITEM 'TYPE)
                                                                   'NUMBER)
                                                              THEN (FUNCTION \FM.NUMBER-CHANGESTATE])
)

;; EDIT ITEMS

(DECLARE%: DONTCOPY

(DECLARE%: EVAL@COMPILE

(RPAQQ \FM.EDIT-TIMEOUT 100000)

(RPAQQ \FM.EDIT-RIGHTENDSPACE 5)

(RPAQQ \FM.EDIT-BLOCKSIZE 50)

(RPAQQ \FM.EDIT-CONTROLCHARS (9 10 12 13))

(RPAQQ \FM.EDIT-CONTROLCHARSECHO 255)

(RPAQQ \FM.EDIT-WORDDELIMCHARS (32 123 125 91 93 60 62 47 92 46 44 59 42 40 41 45))

[CONSTANTS (\FM.EDIT-TIMEOUT 100000)
       (\FM.EDIT-RIGHTENDSPACE 5)
       (\FM.EDIT-BLOCKSIZE 50)
       (\FM.EDIT-CONTROLCHARS '(9 10 12 13))
       (\FM.EDIT-CONTROLCHARSECHO 255)
       (\FM.EDIT-WORDDELIMCHARS '(32 123 125 91 93 60 62 47 92 46 44 59 42 40 41 45]
)
)

(RPAQQ \FM.EDIT-TTBL NIL)

(DECLARE%: DOEVAL@COMPILE DONTCOPY

(GLOBALVARS \FM.EDIT-TTBL)
)

(DECLARE%: EVAL@COMPILE

(PUTPROPS \FM.EDIT-MAXWIDTH MACRO [NIL (OR LABELMAXWIDTH (IPLUS (WINDOWPROP WINDOW 'WIDTH)
                                                                (fetch (REGION LEFT) of (DSPCLIPPINGREGION NIL
                                                                                                 WINDOW))
                                                                (MINUS LEFT])

(PUTPROPS \FM.EDIT-SCROLLAMOUNT MACRO (NIL (IQUOTIENT (WINDOWPROP WINDOW 'WIDTH)
                                                       2)))
)

(DEFINEQ

(\FM.EDIT-SETUP
  [LAMBDA (ITEM)                                                        ; Edited  5-Dec-94 15:44 by jds

    ;; LABELMAXWIDTH is maximum width string can reach.  Right now it is set to MAXWIDTH, leaving no right end space.

    (OR \FM.EDIT-TTBL (\FM.EDIT-SETUPTTBL))                             ; since have edit item, setup term table
    (\FM.ITEMPROP ITEM 'REGION (\FM.ITEMPROP ITEM 'MAXREGION))          ; always sensitive on maxregion
    [COND
       ([AND (\FM.ITEMPROP ITEM 'BOX)
             (NOT (\FM.ITEMPROP ITEM 'MAXWIDTH]                         ; boxing implies maxwidth
        (\FM.ITEMPROP ITEM 'MAXWIDTH (IDIFFERENCE (fetch (REGION WIDTH) of (\FM.ITEMPROP ITEM 'REGION))
                                            (ITIMES 2 (\FM.ITEMPROP ITEM 'BOXOFFSET]
    (COND
       [(\FM.ITEMPROP ITEM 'MAXWIDTH)                                   ; setup stopwidth
        (\FM.ITEMPROP ITEM 'LABELMAXWIDTH (\FM.ITEMPROP ITEM 'MAXWIDTH]
       (T                                                               ; make item infinite
          (\FM.ITEMPROP ITEM 'INFINITEWIDTH T)))
    (OR (\FM.ITEMPROP ITEM 'MESSAGE)
        (\FM.ITEMPROP ITEM 'MESSAGE (FUNCTION \FM.EDIT-MESSAGE)))
    (\FM.ITEMPROP ITEM 'INITSTATE (\FM.ITEMPROP ITEM 'LABEL))
    (\FM.ITEMPROP ITEM 'SYSSELECTEDFN (FUNCTION \FM.EDIT-ITEM])

(\FM.EDIT-MESSAGE
  [LAMBDA (ITEM WINDOW BUTTONS)                                         (* jow "24-Apr-86 22:05")
    (SELECTQ (CAR BUTTONS)
         (RIGHT "Will clear first, then start editing.")
         ((LEFT MIDDLE)
             "Will start editing at this position.")
         NIL])
```

```
(\FM.EDIT-SETUPTTBL
  [LAMBDA NIL                                                  (* jow "21-Aug-86 12:50")

            (* creates a new term table in \FM.TTBL with no line buffering or control character echoing.)

    (SETQ \FM.EDIT-TTBL (COPYTERMTABLE 'ORIG))
    (ECHOMODE NIL \FM.EDIT-TTBL)
    (for CC from 0 to 31 do (ECHOCONTROL CC 'REAL \FM.EDIT-TTBL])

(\FM.EDIT-ITEM
  [LAMBDA (EDITITEM WINDOW BUTTONS STARTFLG DONEFN)            ; Edited  6-Dec-94 10:16 by jds

;;; called when an edit item gets selected.  If STARTFLG is T, start editing the item at the beginning, rather than at the current mouse position.

    (DECLARE (SPECVARS DONEFN EDITITEM WINDOW))
    (COND
       ((EQ (CAR BUTTONS)
            'RIGHT)
        (FM.CHANGELABEL EDITITEM "" WINDOW)))
    (LET ((TIMEOUT (SETUPTIMER 0))
          FONT BITMAP LEFT DISPLAYLEFT DISPLAYBOTTOM BOXOFFSET CHCODE CHARWIDTH MAXWIDTH STRINGPTR TAILPTR
          MOUSEX MOUSEY ITEM ITEMWIDTH LIMITCHARS SYSLIMITCHARS ECHOCHAR STREAM)
         (RESETLST                                             ; setup system
             (RESETSAVE (SETTERMTABLE \FM.EDIT-TTBL))
             (RESETSAVE (TTYDISPLAYSTREAM WINDOW))
             (RESETSAVE (CURSOR T))
             (RESETSAVE NIL (LIST 'WINDOWPROP WINDOW 'FM.EDITITEM NIL))
             (RESETSAVE NIL (LIST 'WINDOWPROP WINDOW 'FM.EDIT-TIMER NIL))
             (RESETSAVE NIL (LIST 'WINDOWPROP WINDOW 'PROCESS NIL))
             (\FM.EDIT-PREPARETOEDIT EDITITEM STARTFLG)            ; setup item info
             (RESETSAVE (TTY.PROCESS (THIS.PROCESS)))          ; grab the tty last, so won't have it unless the menu thinks it's
                                                               ; editing.
             [do (SETUPTIMER \FM.EDIT-TIMEOUT TIMEOUT)

                 ;; wait for something interesting to happen. while waiting, call tty fns to make caret flash, etc,

                 (until (OR (MOUSESTATE (NOT UP))
                            (READP)
                            (TIMEREXPIRED? TIMEOUT)
                            (NOT (TTY.PROCESSP)))
                    do (\TTYBACKGROUND))
                 [COND
                    [(NOT (TTY.PROCESSP))
                     (RETURN (AND DONEFN (APPLY* DONEFN EDITITEM WINDOW]
                    [(LASTMOUSESTATE (NOT UP))
                     (SETQ BUTTONS (DECODEBUTTONS))
                     (SETQ MOUSEX (LASTMOUSEX WINDOW))
                     (SETQ MOUSEY (LASTMOUSEY WINDOW))
                     (SETQ ITEM (\FM.CHECKREGION WINDOW MOUSEX MOUSEY))
                     (COND
                        ((EQ ITEM EDITITEM)
                         (SELECTQ (CAR BUTTONS)
                             (LEFT (\FM.EDIT-MOVECARET))
                             (RIGHT (\FM.EDIT-DELETE))
                             NIL))
                        (ITEM                                  ; run new buttonfn, in THIS PROCESS.*
                              (\CARET.DOWN)
                              (AND DONEFN (SETQ BUTTONS (APPLY* DONEFN EDITITEM WINDOW)))
                                                               ; just reuse BUTTONS to hold result DONEFN
                              (\FM.MENUHANDLER WINDOW T)
                              (RETURN BUTTONS))
                        (T                                     ; let other button events run
                           (BLOCK]
                    ((READP)
                     (SETQ CHCODE (\GETKEY))
                     (SELECTQ CHCODE
                         (530                                  ; SKIP-NEXT key
                              (AND DONEFN (APPLY* DONEFN EDITITEM WINDOW))
                              (FM.SKIPNEXT WINDOW))
                         (562                                  ; SHIFT-SKIP-NEXT key means clear first
                              (AND DONEFN (APPLY* DONEFN EDITITEM WINDOW))
                              (FM.SKIPNEXT WINDOW T))
                         (SELECTQ (GETSYNTAX CHCODE \FM.EDIT-TTBL)
                             (CHARDELETE                       ; backup char,
                                          (\FM.EDIT-BACKUP))
                             (WORDDELETE                       ; delete word
                                          (\FM.EDIT-WORDDELETE))
                             (\FM.EDIT-INSERT]
                 (COND
                    ((TIMEREXPIRED? TIMEOUT)
                     (RETURN (AND DONEFN (APPLY* DONEFN EDITITEM WINDOW])])

(\FM.EDIT-PREPARETOEDIT
  [LAMBDA (EDITITEM STARTFLG)                                  ; Edited  5-Dec-94 15:46 by jds
```

;;; called to prepare edit info, displaystream, and window for EDITITEM. References variables bound in FM.EDIT-ITEM.

```
    (\FM.ITEMPROP EDITITEM 'SELECTEDFN (FUNCTION NILL))          ; insure edit item won't have selectedfn side effect, because end
                                                                 ; of edit is not well defined
    (WINDOWPROP WINDOW 'FM.EDITITEM EDITITEM)
    (WINDOWPROP WINDOW 'FM.EDIT-TIMER TIMEOUT)
    (SETQ BOXOFFSET (OR (\FM.ITEMPROP EDITITEM 'BOXOFFSET)
                        0))
    (SETQ FONT (\FM.ITEMPROP EDITITEM 'FONT))
    (SETQ BITMAP (\FM.ITEMPROP EDITITEM 'BITMAP))
    (SETQ ITEMWIDTH (STRINGWIDTH (\FM.ITEMPROP EDITITEM 'LABEL)
                           FONT))
    (SETQ MAXWIDTH (\FM.ITEMPROP EDITITEM 'MAXWIDTH))
    (SETQ LIMITCHARS (\FM.ITEMPROP EDITITEM 'LIMITCHARS))
    (SETQ SYSLIMITCHARS (\FM.ITEMPROP EDITITEM 'SYSLIMITCHARS))
    (SETQ ECHOCHAR (\FM.ITEMPROP EDITITEM 'ECHOCHAR))            (* setup edit pointer info)
    (SETQ STREAM (DSPCREATE BITMAP))
    (LET ((REGION (\FM.ITEMPROP EDITITEM 'REGION))
          POINTER)
         (SETQ DISPLAYLEFT (fetch (REGION LEFT) of REGION))
         (SETQ DISPLAYBOTTOM (fetch (REGION BOTTOM) of REGION))
         (SETQ LEFT (IPLUS DISPLAYLEFT BOXOFFSET))
         [SETQ POINTER (COND
                          (STARTFLG (CONS 1 0))
                          (T (\FM.EDIT-GETPOINTERINFO (\FM.ITEMPROP EDITITEM 'LABEL)
                                    FONT LEFT (LASTMOUSEX WINDOW]
         (SETQ STRINGPTR (CAR POINTER))
         (SETQ TAILPTR (IPLUS BOXOFFSET (CDR POINTER)))          ; setup window x and y position, so caret it right place
         (DSPXPOSITION (IPLUS LEFT (CDR POINTER))
                WINDOW)
         (DSPYPOSITION (IPLUS DISPLAYBOTTOM (FONTPROP FONT 'DESCENT)
                              BOXOFFSET)
                WINDOW)                                          ; setup edit stream, used for printing inserted characters to the
                                                                 ; bitmap
         (DSPXPOSITION TAILPTR STREAM)
         (DSPYPOSITION (IPLUS (FONTPROP FONT 'DESCENT)
                              BOXOFFSET)
                STREAM)
         (DSPFONT FONT STREAM])
```

(\FM.EDIT-FINDNEXT

```
  [LAMBDA NIL                                                    ; Edited  5-Dec-94 15:33 by jds

                (* find the next edit item in the freemenu after ITEM. Return NIL if there isn't another one.)

    (for I in [CDR (FMEMB EDITITEM (WINDOWPROP WINDOW 'FM.ITEMS] thereis (FMEMB (\FM.ITEMPROP I 'TYPE)
                                                                              '(EDIT NUMBER])
```

(\FM.EDIT-FINDFIRST

```
  [LAMBDA (WINDOW)                                               (* jow "18-Jun-86 17:01")
                                                                 (* start editing the first edit item in the menu.)
    (for I in (WINDOWPROP WINDOW 'FM.ITEMS) thereis (EQ (\FM.ITEMPROP I 'TYPE)
                                                        'EDIT])
```

(\FM.EDIT-BACKUP

```
  [LAMBDA NIL                                                    (* jow "24-Apr-86 16:23")
                                                                 (* backup 1 character, if possible)
    (if (IGREATERP STRINGPTR 1)
        then (SETQ STRINGPTR (SUB1 STRINGPTR))
             (SETQ CHARWIDTH (CHARWIDTH (NTHCHARCODE (\FM.ITEMPROP EDITITEM 'LABEL)
                                               STRINGPTR)
                                    FONT))
             (RELMOVETO (MINUS CHARWIDTH)
                    0 WINDOW)
             (RELMOVETO (MINUS CHARWIDTH)
                    0 STREAM)
             (if (ILESSP (DSPXPOSITION NIL WINDOW)
                        (fetch (REGION LEFT) of (DSPCLIPPINGREGION NIL WINDOW)))
                 then (SCROLLW WINDOW (\FM.EDIT-SCROLLAMOUNT)
                             0)                                  (* about to backup off window%: scroll.)
                      )
             (BITBLT BITMAP TAILPTR BOXOFFSET BITMAP (IDIFFERENCE TAILPTR CHARWIDTH)
                    BOXOFFSET
                    (IPLUS BOXOFFSET ITEMWIDTH (MINUS TAILPTR))
                    (FONTPROP FONT 'HEIGHT))
             (\FM.ITEMPROP EDITITEM 'LABEL (\FM.EDIT-STRDELETE (\FM.ITEMPROP EDITITEM 'LABEL)
                                                 STRINGPTR STRINGPTR))
             (SETQ ITEMWIDTH (IDIFFERENCE ITEMWIDTH CHARWIDTH))
             (SETQ TAILPTR (IDIFFERENCE TAILPTR CHARWIDTH))
             (\FM.EDIT-UPDATEAFTERDELETE])
```

(\FM.EDIT-WORDDELETE

```
  [LAMBDA NIL                                                  (* jow "24-Apr-86 16:54")

                (* called on ^W. The list \FM.EDIT-WRODDELIMCHARS specifies a list of character codes that stop word delete.
                Backup over any number of these chars, then any number of non-delim chars, until get to another delim char, leaving that
                char in the string.)

    (if (NEQ STRINGPTR 1)
        then (LET ((END (SUB1 STRINGPTR))
                   (STRING (\FM.ITEMPROP EDITITEM 'LABEL))
                   (ENDTAILPTR BOXOFFSET))
                  (while (AND (NEQ END 1)
                              (FMEMB (NTHCHARCODE STRING (SUB1 END))
                                     \FM.EDIT-WORDDELIMCHARS))
                     do                                        (* move END back to the farthest sequential delim char)
                        (SETQ END (SUB1 END)))
                  (while (AND (NEQ END 1)
                              (NOT (FMEMB (NTHCHARCODE STRING (SUB1 END))
                                          \FM.EDIT-WORDDELIMCHARS)))
                     do                                        (* move END back to the farthest sequential non-delim char)
                        (SETQ END (SUB1 END)))                 (* now END is pointing to the farthest char to be deleted)
                  [if (NEQ END 1)
                      then (SETQ ENDTAILPTR (IPLUS BOXOFFSET (STRINGWIDTH (SUBSTRING STRING 1 (SUB1 END))
                                                                          FONT]
                  (BITBLT BITMAP TAILPTR BOXOFFSET BITMAP ENDTAILPTR BOXOFFSET (IPLUS BOXOFFSET ITEMWIDTH
                                                                                      (MINUS TAILPTR))
                         (FONTPROP FONT 'HEIGHT))
                  (\FM.ITEMPROP EDITITEM 'LABEL (\FM.EDIT-STRDELETE STRING END (SUB1 STRINGPTR)))
                  (SETQ ITEMWIDTH (STRINGWIDTH (\FM.ITEMPROP EDITITEM 'LABEL)
                                         FONT))
                  (SETQ STRINGPTR END)
                  (SETQ TAILPTR ENDTAILPTR)
                  (DSPXPOSITION (IPLUS LEFT TAILPTR)
                         WINDOW)
                  (DSPXPOSITION TAILPTR STREAM)
                  (if (ILESSP (DSPXPOSITION NIL WINDOW)
                              (fetch (REGION LEFT) of (DSPCLIPPINGREGION NIL WINDOW)))
                      then (SCROLLW WINDOW (\FM.EDIT-SCROLLAMOUNT)
                                  0)                           (* about to backup off window%: scroll.)
                           )
                  (\FM.EDIT-UPDATEAFTERDELETE])
```

(\FM.EDIT-INSERT

```
  [LAMBDA NIL                                                  ; Edited  5-Dec-94 15:42 by jds

    ;; Insert a character into an EDIT or NUMBER freemenu item.  Before inserting, check against LIMITCHARS, the user-specified character limit, and
    ;; against SYSLIMITCHARS, the system-provided limit.  For example, on NUMBER items, SYSLIMITCHARS limits type-in to numbers, but the
    ;; user's LMIITCHARS function might also enable CR as a skip-next character.

    (COND
       ([AND [OR (NOT LIMITCHARS)
                 (AND (LISTP LIMITCHARS)
                      (FMEMB (CHARACTER CHCODE)
                             LIMITCHARS))
                 (AND (ATOM LIMITCHARS)
                      (APPLY* LIMITCHARS EDITITEM WINDOW (CHARACTER CHCODE]
             (OR (NOT SYSLIMITCHARS)
                 (AND (LISTP SYSLIMITCHARS)
                      (FMEMB (CHARACTER CHCODE)
                             SYSLIMITCHARS))
                 (AND (ATOM SYSLIMITCHARS)
                      (APPLY* SYSLIMITCHARS EDITITEM WINDOW (CHARACTER CHCODE]
                                                               ; insert a single character, CHCODE into the string
        (SETQ CHARWIDTH (CHARWIDTH CHCODE FONT))
        (COND
           ((OR (NOT MAXWIDTH)
                (ILEQ (IPLUS ITEMWIDTH CHARWIDTH)
                      MAXWIDTH))                               ; i am going to insert
            (RELMOVETO CHARWIDTH 0 WINDOW)
            (COND
               ([IGREATERP (DSPXPOSITION NIL WINDOW)
                       (IPLUS (fetch (REGION LEFT) of (DSPCLIPPINGREGION NIL WINDOW))
                              (fetch (REGION WIDTH) of (DSPCLIPPINGREGION NIL WINDOW]
                                                               ; about to type off window: scroll back.
                (add (fetch (REGION WIDTH) of (WINDOWPROP WINDOW 'EXTENT))
                     \FM.EDIT-BLOCKSIZE)
                (SCROLLW WINDOW (MINUS (\FM.EDIT-SCROLLAMOUNT))
                       0)))
            (COND
               ((IGREATERP (IPLUS ITEMWIDTH CHARWIDTH)
                       (BITMAPWIDTH BITMAP))                   ; current bitmap too small, make new one.  This won't get done if
                                                               ; item is boxed.
                (\FM.ITEMPROP EDITITEM 'BITMAP (BITMAPCREATE (IPLUS (BITMAPWIDTH BITMAP)
                                                                    \FM.EDIT-BLOCKSIZE)
                                                      (BITMAPHEIGHT BITMAP)))
                (BITBLT BITMAP 0 0 (\FM.ITEMPROP EDITITEM 'BITMAP)
                       0 0)
                (SETQ BITMAP (\FM.ITEMPROP EDITITEM 'BITMAP))
                (DSPDESTINATION BITMAP STREAM)))                    ; now insert character into bitmap
            (BITBLT BITMAP TAILPTR BOXOFFSET BITMAP (IPLUS TAILPTR CHARWIDTH)
                   BOXOFFSET
                   (IPLUS BOXOFFSET ITEMWIDTH (MINUS TAILPTR))
                   (FONTPROP FONT 'HEIGHT))
            (SETQ ITEMWIDTH (IPLUS ITEMWIDTH CHARWIDTH))
            (COND
               ((FMEMB CHCODE \FM.EDIT-CONTROLCHARS)                ; for CR, LF, TAB, etc, echo non control action char
                (PRIN1 (OR ECHOCHAR (CHARACTER \FM.EDIT-CONTROLCHARSECHO))
                       STREAM))
               (T (PRIN1 (OR ECHOCHAR (CHARACTER CHCODE))
                         STREAM)))
            (\CARET.DOWN)
            (BITBLT BITMAP 0 0 WINDOW DISPLAYLEFT DISPLAYBOTTOM MAXWIDTH)
            (\FM.ITEMPROP EDITITEM 'LABEL (\FM.EDIT-STRINSERT (\FM.ITEMPROP EDITITEM 'LABEL)
                                                 (CHARACTER CHCODE)
                                                 STRINGPTR))
            (SETQ STRINGPTR (ADD1 STRINGPTR))
            (SETQ TAILPTR (IPLUS TAILPTR CHARWIDTH))
            (EXTENDREGION (WINDOWPROP WINDOW 'EXTENT)
                   (CREATEREGION LEFT 0 (IPLUS ITEMWIDTH BOXOFFSET)
                          0])
```

(\FM.EDIT-DELETE

```
  [LAMBDA NIL                                                       (* jow "10-Jun-86 16:12")

             (* Called when a right button event occurs in ITEM's region, while it is being edited.
             Delete the substring of the items string starting at the current position, and ending at the position of MOUSEX, inclusive.)

    (\CARET.DOWN)
    (while (MOUSESTATE (NOT UP)) bind (REGION _ (\FM.ITEMPROP EDITITEM 'REGION))
                                      (INFINITEWIDTH _ (\FM.ITEMPROP EDITITEM 'INFINITEWIDTH))
                                      (BOTTOM _ (IPLUS BOXOFFSET DISPLAYBOTTOM))
                                      (HEIGHT _ (FONTPROP FONT 'HEIGHT))
                                      (PIVOT _ (IPLUS DISPLAYLEFT TAILPTR))
                                      END POINTER OLDPOINTER MOVEDOFF
       eachtime (SETQ MOUSEX (LASTMOUSEX WINDOW))
             (SETQ MOUSEY (LASTMOUSEY WINDOW))
       do (if (\FM.ONITEM REGION MOUSEX MOUSEY INFINITEWIDTH)
              then (SETQ OLDPOINTER POINTER)
                   (SETQ POINTER (\FM.EDIT-GETPOINTERINFO (\FM.ITEMPROP EDITITEM 'LABEL)
                                        FONT LEFT MOUSEX))
                   [if (OR MOVEDOFF (NOT (EQUAL POINTER OLDPOINTER)))
                       then (SETQ MOVEDOFF NIL)
                            (SETQ END (IPLUS LEFT (CDR POINTER)))
                            (BITBLT BITMAP 0 0 WINDOW DISPLAYLEFT DISPLAYBOTTOM MAXWIDTH)
                            (if (IGREATERP END PIVOT)
                                then                                (* highlight from pivot to end)
                                     (BLTSHADE BLACKSHADE WINDOW PIVOT BOTTOM (IDIFFERENCE END PIVOT)
                                            HEIGHT
                                            'INVERT)
                              else                                  (* highlight from end to pivot)
                                   (BLTSHADE BLACKSHADE WINDOW END BOTTOM (IDIFFERENCE PIVOT END)
                                          HEIGHT
                                          'INVERT]
            elseif (NOT MOVEDOFF)
              then (BITBLT BITMAP 0 0 WINDOW DISPLAYLEFT DISPLAYBOTTOM MAXWIDTH)
                   (SETQ MOVEDOFF T))
       finally (if (AND (\FM.ONITEM REGION MOUSEX MOUSEY INFINITEWIDTH)
                        (NEQ (CAR POINTER)
                             STRINGPTR))
                   then (if (IGREATERP END PIVOT)
                            then                                    (* from current to right%: pointers and xpositions remain the
                                                                    same)
                                 (BITBLT BITMAP (IPLUS BOXOFFSET (CDR POINTER))
                                        BOXOFFSET BITMAP TAILPTR BOXOFFSET (IPLUS BOXOFFSET ITEMWIDTH (MINUS
                                                                                                      TAILPTR
                                                                                                      ))
                                        HEIGHT)
                                 [\FM.ITEMPROP EDITITEM 'LABEL (\FM.EDIT-STRDELETE (\FM.ITEMPROP EDITITEM
                                                                                           'LABEL)
                                                                      STRINGPTR
                                                                      (SUB1 (CAR POINTER]
                          else                                      (* from current to left%:)
                               (BITBLT BITMAP TAILPTR BOXOFFSET BITMAP (IPLUS BOXOFFSET (CDR POINTER))
                                      BOXOFFSET
                                      (IPLUS BOXOFFSET ITEMWIDTH (MINUS TAILPTR))
                                      HEIGHT)
                               (\FM.ITEMPROP EDITITEM 'LABEL (\FM.EDIT-STRDELETE (\FM.ITEMPROP EDITITEM 'LABEL)
                                                                    (CAR POINTER)
                                                                    (SUB1 STRINGPTR)))
                               (SETQ STRINGPTR (CAR POINTER))
                               (SETQ TAILPTR (IPLUS BOXOFFSET (CDR POINTER)))
                               (DSPXPOSITION END WINDOW)
                               (DSPXPOSITION TAILPTR STREAM))
                        (SETQ ITEMWIDTH (STRINGWIDTH (\FM.ITEMPROP EDITITEM 'LABEL)
                                                FONT))
                          (\FM.EDIT-UPDATEAFTERDELETE])

(\FM.EDIT-GETPOINTERINFO
  [LAMBDA (STRING FONT LEFT MOUSEX)                                (* jow "22-Apr-86 14:58")

             (* calculate string pointer and tail pointer from mouse location within string.
             Assume mousex in window coordinates, not REGION coordinates.
             Return as dotted pair (stringptr . tailptr) -- Each character is sensitive 2 bits to the left to allow for mousing between chars)

    (SETQ MOUSEX (IDIFFERENCE MOUSEX LEFT))
    (LET ((PTR))
         (for N (WIDTH _ -2) from 1 to (NCHARS STRING) do (add WIDTH (CHARWIDTH (NTHCHARCODE STRING N)
                                                                                FONT))
                                                          (if (IGREATERP WIDTH MOUSEX)
                                                              then (SETQ PTR N)
                                                                   (RETURN)))
         (if PTR
             then                                                  (* mouse at PTR in string)
                  (CONS PTR (STRINGWIDTH (OR (SUBSTRING STRING 1 (SUB1 PTR))
                                             "")
                                   FONT))
           else                                                    (* mouse at end of string)
                (CONS (ADD1 (NCHARS STRING))
                      (STRINGWIDTH STRING FONT])

(\FM.EDIT-MOVECARET
  [LAMBDA NIL                                                      (* jow "10-Sep-86 10:33")

             (* mouse event has occured at MOUSEX in ITEM's region while editing.
             Move the edit caret to that position)

    (\CARET.DOWN)
    (SETQ POINTER (\FM.EDIT-GETPOINTERINFO (\FM.ITEMPROP EDITITEM 'LABEL)
                         FONT LEFT MOUSEX))
    (DSPXPOSITION (IPLUS LEFT (CDR POINTER))
           WINDOW)                                                 (* move caret)
    (SETQ STRINGPTR (CAR POINTER))                                 (* update edit pointers)
    (SETQ TAILPTR (IPLUS BOXOFFSET (CDR POINTER)))
    (DSPXPOSITION TAILPTR STREAM])

(\FM.EDIT-STRDELETE
  [LAMBDA (STRING N M)                                             (* jow "17-Jul-85 00:29")

             (* delete from characters N through M of STRING. no bounds checks are made on N and M.
             returns a new string)

    (CONCAT (OR (SUBSTRING STRING 1 (SUB1 N))
                "")
           (OR (SUBSTRING STRING (ADD1 M)
                      (NCHARS STRING))
               ""])

(\FM.EDIT-STRINSERT
  [LAMBDA (STRING CHAR N)                                          (* jow "17-Jul-85 00:40")

             (* return new string with CHAR inserted as new character at position N.
             just appends CHAR if N is 1 greater than nchars)

    (CONCAT (OR (SUBSTRING STRING 1 (SUB1 N))
                "")
           CHAR
           (OR (SUBSTRING STRING N (NCHARS STRING))
               ""])

(\FM.EDIT-UPDATEAFTERDELETE
  [LAMBDA NIL                                                      (* jow "10-Jun-86 16:09")
                                                                   (* called to update the screen after a delete has occured.)
    (\CARET.DOWN)
    (BLTSHADE WHITESHADE BITMAP (IPLUS BOXOFFSET ITEMWIDTH)
           BOXOFFSET
           (IDIFFERENCE (BITMAPWIDTH BITMAP)
                  (IPLUS ITEMWIDTH BOXOFFSET BOXOFFSET))
           (FONTPROP FONT 'HEIGHT))                                (* whiteout to rightmargin)
    (BITBLT BITMAP 0 0 WINDOW DISPLAYLEFT DISPLAYBOTTOM MAXWIDTH])
)

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

(ADDTOVAR NLAMA )
(ADDTOVAR NLAML )

(ADDTOVAR LAMA FM.ITEMPROP)
)

(PUTPROPS FREEMENU COPYRIGHT ("Venue & Xerox Corporation" 1986 1987 1988 1990 1993 1994))
```

FUNCTION INDEX

MACRO INDEX

CONSTANT INDEX

PROPERTY INDEX

VARIABLE INDEX

RECORD INDEX

OPTIMIZER INDEX