# FREE MENU

Free Menus are powerful and flexible menus that are useful for applications that need menus with different types of items, including command items, state items, and editable items. A Free Menu lives in a window, which can be opened and closed as desired, or attached as a control menu to the application window.

## Making a Free Menu

A Free Menu is built from a description of the contents and layout of the menu. As a Free Menu is simply a group of items, a Free Menu Description is simply a specification of a group of items. Each group has properties associated with it, as does each Free Menu Item. These properties specify the format of the items in the group, and the behavior of each item. The function FREEMENU takes a Free Menu Description, and returns a closed window with the Free Menu in it.

Probably the easiest way to make a Free Menu is to define your own function which calls FREEMENU with the Free Menu Description right there in your function. This function can then also set up the Free Menu window as required by the application. The Free Menu Description is then saved as part of your function when you save your application.

Alternatively, you can save the Free Menu Description as a variable in your file, and then just call FREEMENU with the name of the variable. This may be a more difficult alternative if you want to use the backquote facility to built your Free Menu Description. See the section **Free Menu Item Descriptions**.

### Free Menu Formatting

A Free Menu can be formatted in one of four ways. The items in any group can be automatically layed out in rows, in columns, or in a table, or else the application can specify the exact location of each item in the group. Additionally, Free Menu keeps track of the region that a group of items occupies, and items can be justified within that region. This way an item can be automatically positioned at one of the nine justification locations, top-left, top-center, top-right, middle-left, etc.

### Free Menu Description

A Free Menu Description, specifying a group of items, is a list structure. The first thing in the list is an optional list of the properties of this group of items, in the form:

```
(PROPS <PROP> <VALUE> <PROP> <VALUE> ...)
```

The key word PROPS determines whether or not the optional group props list is specified. The section **Free Menu Group Properties** describes each group property. For now, the important property is FORMAT. The type of formatting determines the syntax of the rest of the Free Menu Description, in a very simple way.

When using EXPLICIT formatting, the rest of the description is any number of Item Descriptions, which have LEFT and BOTTOM properties specifying the position of the item in the menu. The syntax is:

```
((PROPS FORMAT EXPLICIT ...)
 <ITEM DESCRIPTION>
 <ITEM DESCRIPTION> ...)
```

When using ROW or TABLE formatting, the rest of the description is any number of item groups, each group corresponding to a row in the menu. These groups are identical in *syntax* to an EXPLICIT group description, with an optional PROPS list and then any number of Item Descriptions, except that the items need not have LEFT and BOTTOM properties, as the location of each item is figured out by the formatter. But the order of the rows and items is important. The menu is layed out top to bottom by row, and left to right within each row. The syntax is (the comments are not part of the description):

```
((PROPS FORMAT ROW ...)           ; props of this group
 (<ITEM DESCRIPTION>                ; items in first row
  <ITEM DESCRIPTION> ...)
 ((PROPS ...)                    ; props of second row
  <ITEM DESCRIPTION>              ; items in second row
  <ITEM DESCRIPTION> ...))
```

When using COLUMN formatting, the syntax is identical to that of ROW formatting. However each group of items corresponds to a column in the menu, rather than a row. The menu is layed out left to right by column, top to bottom within each column.

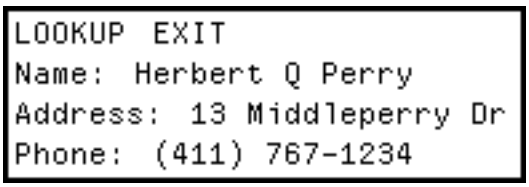
Finally, a Free Menu Description can have recursively nested groups. Anywhere the description can take an Item Description, it can take a group, marked by the key word GROUP. A nested group inherits all of the properties of its mother group, by default. However, any of these properties can be overridden in the nested groups PROPS list, including the FORMAT. The syntax is:

```
(                          ; no PROPS list, default row format
 (<ITEM DESCRIPTION>                          ; first in row
  (GROUP                  ; nested group, second in row
     (PROPS FORMAT COLUMN...)          ; optional props
     (<ITEM DESCRIPTION> ...)            ; first column
     (<ITEM DESCRIPTION> ...))
  <ITEM DESCRIPTION>))                    ; third in row
```

Here is an example of a simple Free Menu Description, for a menu which might provide access to a simple data base:

```
(((LABEL LOOKUP SELECTEDFN MYLOOKUPFN) (LABEL EXIT SELECTEDFN MYEXITFN))
 ((LABEL Name: TYPE DISPLAY) (LABEL "" TYPE EDIT ID NAME))
 ((LABEL Address: TYPE DISPLAY) (LABEL "" TYPE EDIT ID ADDRESS))
 ((LABEL Phone: TYPE DISPLAY)
  (LABEL "" TYPE EDIT LIMITCHARS MYPHONEP ID PHONE)))
```

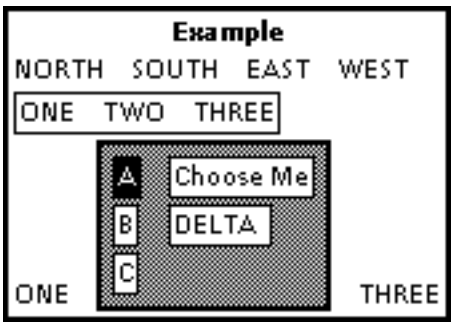
This menu has two command buttons, LOOKUP and EXIT, and three edit fields, with ID's NAME, PHONE, and ADDRESS. The Edit items are initialized to the empty string, as in this example they need no other initial value. The user could click after the Name: prompt, type a person's name, and then press the LOOKUP button. This would cause the function MYLOOKUPFN to be called, which could look at the NAME Edit item, lookup that name in the data base, and then fill in the rest of the fields appropriately. Note that the PHONE item has MYPHONEP as a LIMITCHARS function. This function would be called when editing the phone number, in order to restrict input to a valid phone number. After looking up Perry, the Free Menu might look like:



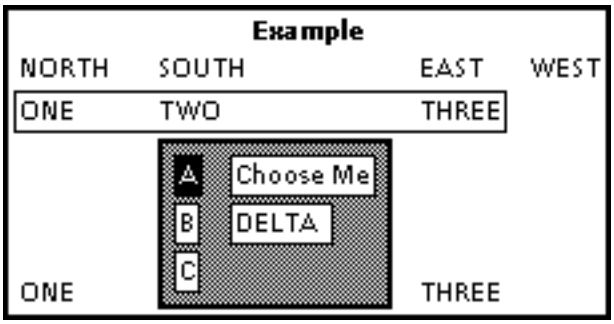

Here is a more complicated example:

```
((PROPS FONT (MODERN 10))
 ((LABEL Example FONT (MODERN 10 BOLD) HJUSTIFY CENTER))
 ((LABEL NORTH) (LABEL SOUTH) (LABEL EAST) (LABEL WEST))
 ((PROPS ID ROW3 BOX 1)
  (LABEL ONE) (LABEL TWO) (LABEL THREE))
 ((PROPS ID ROW4)
  (LABEL ONE ID ALPHA)
  (GROUP (PROPS FORMAT COLUMN BACKGROUND 23130 BOX 2 BOXSPACE 4)
         ((TYPE NWAY LABEL A BOX 1 COLLECTION COL1 NWAYPROPS (DESELECT T))
          (TYPE NWAY LABEL B BOX 1 COLLECTION COL1)
          (TYPE NWAY LABEL C BOX 1 COLLECTION COL1))
         ((TYPE STATE LABEL "Choose Me" BOX 1 MENUITEMS (BRAVO DELTA)
           INITSTATE DELTA LINKS (DISPLAY (GROUP ALPHA)))
          (TYPE DISPLAY ID ALPHA LABEL "" BOX 1 MAXWIDTH 35)))
   (LABEL THREE)))
```

which will produce the following Free Menu:



And if the Free Menu were formatted as a Table, instead of in Rows, it would look like:




## Free Menu Group Properties

Each group has properties. Most group properties are relevant, and should be set, in the group's PROPS list in the Free Menu Description. User properties can freely be included in the PROPS list. A few other properties are setup by the formatter. After the Free Menu is created, group properties can be accessed by the macro FM.GROUPPROP or FM.MENUPROP.

ID — The identifier of this group. Setting the group ID is desirable, for example, if the application needs to get handles on items in particular groups, or access group properties.

FORMAT — One of ROW, COLUMN, TABLE, or EXPLICIT. The default is ROW.

FONT — A font description of the form (FAMILY SIZE FACE), or a FONTDESCRIPTOR data type. This will be the default font for each item in this group. The default font of the top group is the value of the variable DEFAULTFONT.

COORDINATES — One of GROUP, or MENU. This property applies only to Explicit formatting. If GROUP, then the items in the explicit group are positioned in coordinates reletive to the lower left corner of the group, as determined by the mother group. If MENU, which is the default, then the items are positioned reletive to the lower left corner of the menu.

LEFT — Specifies a left offset for this group, pushing the group to the right.

BOTTOM — Specifies a bottom offset for this group, pushing the group up.

ROWSPACE — The number of bits between rows in this group.

COLUMNSPACE — The number of bits between columns in this group.

BOX — The number of bits in the box around this group of items.

BOXSHADE — The shade of the box.

BOXSPACE — The number of bits between the box and the items.

BACKGROUND — The background shade of this group. Nested groups will inherit this background shade, but items in this group and nested groups will not. This is because in general it is difficult to read text on a

background, so items appear on white background by default. This can be overridden by the BACKGROUND Item Property.

### Other Group Properties

The following group properties are setup and maintained by Free Menu. The application should probably not change any of these properties.

ITEMS — A list of the items in the group.

REGION — The region that is the extent of the items in the group.

MOTHER — The ID of the group that is the mother of this group.

DAUGHTERS — A list of ID of groups which are daughters to this group.

## Free Menu Items

Each Free Menu Item is stored as an instance of the Data Type FREEMENUITEM. Free Menu Items can be thought of as objects, each item having its own particular properties, such as its type, label, and mouse event functions. A number of useful item types, described in the section **Free Menu Item Types**, are predefined by Free Menu. New types of items can be defined by the application, using Display items as a base.

Each Free Menu Item is created from a Free Menu Item Description when the Free Menu is created.

### Free Menu Item Descriptions

A Free Menu Item Description is a list in property list format, specifying the properties of the item. For example:

```
(LABEL Refetch SELECTEDFN MY.REFETCHFN)
```

describes a command (Momentary) item labelled 'Refetch', with the function MY.REFETCHFN to be called when the item is selected.

None of the property values in an item description are evaluated. When constructing Free Menu descriptions that incorporate evaluated expressions, for example labels that are bitmaps, it is helpful to use the backquote facility. For example, if the value of the variable MYBITMAP is a bitmap, then

```
(FREEMENU `(((LABEL A) (LABEL ,MYBITMAP))))
```

would create a Free Menu of one row, with two items in that row, the second of which has the value of MYBITMAP as its label.

### Free Menu Item Properties

The following Free Menu Item Properties can be set in the Item Description. Any other properties given in an Item Description will be treated as user properties, and will be saved on the USERDATA property of the item.

TYPE — The type of the item. Choose from one of the Free Menu Item type keywords MOMENTARY, TOGGLE, 3STATE, STATE, NWAY, EDITSTART, EDIT, NUMBER, or DISPLAY. The default is MOMENTARY.

| | |
|---|---|
| LABEL | An atom, string, or bit map. Bit maps are always copied, so that the original won't be changed. This property must be specified for every item. |
| FONT | The font that the item will appear in. The default is the font specified for the group that this item is in. Can be a font description of the form (FAMILY SIZE FACE), or a FONTDESCRIPTOR data type. |
| ID | May be used to specify a unique identifier for this item, but is not necessary. |
| LEFT and BOTTOM | When Row, Column, or Table formatting, these specify offsets, pushing the item right and up, respectively, from where the formatter would have put the item. In Explicit formatting, these are the actual coordinates of the item, in the coordinate system given by the group's COORDINATES property. |
| HJUSTIFY | One of LEFT, CENTER, or RIGHT. Specifies that this item is to be horizontally justified within the extent of its group. Note that the main group, as opposed to the smaller row or column group, is used. |
| VJUSTIFY | One of TOP, MIDDLE, or BOTTOM. Specifies that this item is to be vertically justified. |
| HIGHLIGHT | Specifies the highlighted looks of the item, that is, how the item changes when a mouse event occurs on it. See the section **Free Menu Item Highlighting**, below. |
| MESSAGE | A string that will be printed in the prompt window after a mouse button is held down over this item for MENUHELDWAIT milliseconds. Or, if an atom, treat as a function to get the message. The function is applied to ITEM WINDOW BUTTONS, and should return a string. The default is a message appropriate to the type of the item. |
| INITSTATE | The initial state of the item. This is only appropriate to TOGGLE, 3STATE, and STATE items. |
| MAXWIDTH | The width allowed for this item. The formatter will leave enough space after the item for the item to grow to this width without collisions. |
| MAXHEIGHT | Similar to MAXWIDTH, but in the vertical dimension. |
| BOX | The number of bits in the box around this item. Boxes are made around MAXWIDTH and MAXHEIGHT dimensions. If unspecified, no box is drawn. |
| BOXSHADE | The shade that the box is drawn in. The default is BLACKSHADE. |
| BOXSPACE | The number of bits between the box and the label. The default is one bit. |
| BACKGROUND | The background shade on which the item appears. The default is WHITESHADE, regardless of the group's background. |
| LINKS | Can be used to link this item to other items in the Free Menu. See the section **Free Menu Item Links**. |

**Mouse Properties**

The following properties provide a way for application functions to be called under certain mouse events. These functions are called with the ITEM, the WINDOW, and the BUTTONS depressed as

arguments. These application functions do not interfere with any Free Menu system functions that take care of handling the different item types. In each case, though, the application function is called *after* the system function. The default for all of these functions is NILL. The value of each of the following properties can be the name of a function, or a lambda expression.

SELECTEDFN — The function to be called when this item is selected. Note that Edit and EditStart items cannot have a selectedfn. See Edit items, below.

DOWNFN — The function to be called when a mouse button goes down over this item, or when the mouse moves over the item with buttons depressed.

HELDFN — The function to be called repeatedly while the mouse is held down over this item.

MOVEDFN — The function to be called when the mouse moves off this item with buttons still depressed.

## System Properties

The following Free Menu Item properties are set and maintained by Free Menu. The application should probably not change these properties directly.

GROUPID — The ID of the smallest group that the item is in. For example, in a row formatted group, the item's GROUPID will be set to the ID of the row that the item is in, not the ID of the whole group.

STATE — The current state of TOGGLE, 3STATE, or STATE items. The state of an NWAY item behaves like that of a toggle item.

BITMAP — The bitmap from which the item is displayed.

REGION — The region of the item, in window coordinates. This is used for locating the display position, as well as determing the mouse sensitive region of the item.

MAXREGION — The maximum region the item may occupy, determined by the MAXWIDTH and MAXHEIGHT properties. This is used by the formatter and the display routines.

SYSDOWNFN
SYSMOVEDFN
SYSSELECTEDFN — These are the system mouse event functions, setup by Free Menu according to the type of the item. These functions are called before the users mouse event functions, and are used to implement highlighting, state changes, editing, etc.

USERDATA — Any other properties are stored on this list in property list format. This list should probably not need to be manipulated directly.

## Predefined Item Types

### Momentary

Momentary items are like command buttons. When the button is selected, its associated function is called.

### Toggle

Toggle items are simple two-state buttons. When depressed the button is highlighted, and it stays that way until pressed again. The states of a toggle button are T and NIL, initially NIL.

**3State**

3State items rotate through NIL, T, and OFF, states each time they are pressed. The default looks of the OFF state are with a diagonal line through the button, while T is highlighted, and NIL is normal. The default initial state is NIL.

The following Item Property applies to 3State items:

OFF — Specifies the looks of a 3STATE item in its OFF state. Similar to HIGHLIGHT. The default is that the label gets a diagonal slash through it.

**State**

State items are general multiple state items. The following Item Property determines how the item changes state:

CHANGESTATE — This Item Property can be changed at any time to change the effect of the item. If a MENU datatype, then this menu is popped up when the item is selected, and the user can select the new state. Otherwise, if this property is given, it is treated as a function name, which is applied to ITEM WINDOW BUTTONS. This function can do whatever it wants, and is expected to return the new state (an atom, string, or bitmap), or NIL, meaning don't change state.

The state of the item can automatically be indicated in the Free Menu, by setting up a DISPLAY link to a Display item in the menu (see **Free Menu Item Links** below). If such a link exists, the label of the DISPLAY item will be changed to the new state. Note that the possible states are not restricted at all, except that if a popup menu is used, of course the possible selections are restricted. The state can be changed to any atom, string, or bitmap, manually via FM.CHANGESTATE.

The following Item Properties are relevent to State items when building a Free Menu:

MENUITEMS — If specified, should be a list of item to go in a popup menu for this item. Free Menu will build the menu and save it as the CHANGESTATE property of the item.

MENUFONT — The font of the items in the popup menu.

MENUTITLE — The title of the popup menu. The default title is the label of the State item.

**Nway**

NWay items provide a way to collect any number of items together, in any format within the Free Menu. Only one item from each Collection can be selected at a time, and that item is highlighted to indicate so.

The following Item Properties are particular to NWay items:

COLLECTION — An identifier that specifies which NWay Collection this item belongs to.

NWAYPROPS — A property list of information to be associated with this collection. This property is only noticed in the Free Menu Description on the first item in a Collection.

NWay Collections are formed by creating a number of NWay items with the same COLLECTION property. Each NWay item acts individually as a Toggle item, and can have its own mouse event functions.

Each NWay Collection itself has properties, its state for instance. After the Free Menu is created, these Collection properties can be

accessed by the macro FM.NWAYPROPS. Note that NWay Collections are different from Free Menu Groups.

There are three NWay Collection properties that Free Menu looks at:

DESELECT — If given, specifies that the Collection can be deselected, yielding a state in which no item in the Collection is selected. When this property is set, the Collection can be deselected by pressing the Right mouse button on any item in the Collection.

STATE — The current state of the Collection, which is the actual item selected.

INITSTATE — Specifies the initial state of the Collection. The value of this property is an item Link Description (see the section **Free Menu Item Links**.)

**Edit**

Edit items are textual items that can be edited. The label for an Edit item cannot be a bitmap. When the item is selected an edit caret appears at that cursor position within the item, allowing inserting and deleting characters at that point. If selected with the Right mouse button, the item is cleared before editing starts. While editing, the Left mouse button moves the caret to a new position within the item. The Right mouse button deletes from the caret to the cursor. Control-W deletes the previous word.

Editing is stopped when another item is selected, when the user clicks in another tty window, or by the Free Menu function FM.ENDEDIT, which is called when the Free Menu is reset, or the window is closed. Additionally, the Free Menu editor will time out after about a minute, returning automatically. Because of the many ways in which editing can terminate, Edit items are not allowed to have a Selectedfn, as it is not clear when this function should be called.

Each Edit item should have an ID specified, which is used when getting the state of the Free Menu, since the string being edited is defined as the state of the item, and thus cannot distinguish edit items. The following Item Properties are particular to Edit items:

MAXWIDTH — Specifies the maximum string width of the item, in bits, after which input will be ignored. If MAXWIDTH is not specified, the items becomes ‘‘infinitely wide‘‘ and input is never restricted.

INFINITEWIDTH — This property is set automatically when MAXWIDTH is not specified. This tells Free Menu that the item has no right end, so that the item becomes mouse sensitive from its left edge to the right edge of the window, within the vertical space of the item.

LIMITCHARS — The input characters allowed can be restricted in two ways: If this item property is a list, it is treated as a list of legal characters; any character not in the list will be ignored. If it is an atom, it is treated as the name of a test predicate, which is applied to ITEM WINDOW CHARACTER when each character is typed. This predicate should return T if the character is legal, NIL otherwise. The LIMITCHARS function can also call FM.ENDEDIT to force the editor to terminate, or FM.SKIPNEXT, to cause the editor to jump to the next edit item in the menu.

ECHOCHAR — This item property can be set to any character. This character will be echoed in the window, regardless of what character is typed. However the item’s label contains the actual string typed. This is useful for operations like password prompting. If ECHOCHAR is used, the font of the item must be fixed pitch.

Unrestricted Edit items should not have other items to their right in the menu, as they will be edited over. If the item is boxed, input is restricted to what will fit in the box. Typing off the edge of the window will cause the window to scroll appropriately. Control characters can be edited, including CR and LF, and they are echoed as a black box. While editing, the Skip/Next key ends editing the current item, and starts editing the next Edit item in the Free Menu.

**Number**

Number items are Edit items that are restricted to numerals. The state of the item is coerced to the the number itself, not a string of numerals.

There is one Number specific Item Property:

NUMBERTYPE — If FLOATP (or FLOAT), then decimals are accepted. Otherwise only whole numbers can be edited.

**EditStart**

EditStart items serve the purpose of starting editing on another item when they are selected. The associated Edit item is linked to the EditStart item by an EDIT link (see **Free Menu Item Links** below). If the EditStart item is selected with the Right mouse button, the Edit item is cleared before editing is started. Similar to Edit items, EditStart items cannot have a Selectedfn, as it is not clear when the associated editing will terminate.

**Display**

Display items serve two purposes. First, they simply provide a way of putting dummy text in a Free Menu, which does nothing when selected. The item's label can be changed, though. Secondly, Display items can be used as the base for new item types. The application can create new item types by specifying DOWNFN, HELDFN, MOVEDFN, and SELECTEDFN for a Display item, making it behave as desired.

## Free Menu Item Highlighting

Each Free Menu Item can specify how it wants to be highlighted. First of all, if the item doesn't specify a HIGHLIGHT property, there are two default highlights. If the item is not boxed, the label is simply inverted, as in normal menus. If the item is boxed, it is highlighted in the shade of the box.

Alternatively, the value of the HIGHLIGHT property can be a SHADE, which will be painted on top of the item when a mouse event occurs on it. Or the HIGHLIGHT property can be an alternate label, which can be an atom, string, or bitmap. If the highlight label is a different size than the item label, the formatter will leave enough space for the larger of the two.

In all of these cases, the looks of the highlighted item are determined when the Free Menu is built, and a bitmap of the item with these looks is created. This bitmap is stored on the item's HIGHLIGHT property, and simply displayed when a mouse event occurs. The value of the highlight property in the Item Description is copied to the userdata list, in case it is needed later for a label change.

## Free Menu Item Links

Links between items are useful for grouping items in abstract ways. In particular, links are used for associating Editstart items with their item to edit, and State items with their state display. The Free Menu Item property LINKS is a property list, where the value of each Link Name property is a pointer to another item.

In the Item Description, the value of the LINK property should be a property list as above. The value of each Link Name property is a Link Description.

A Link Descriptions can be one of the following forms:

`<ID>` Simply an ID of an item in the Free Menu. This is okay if items can be distinguished by ID alone.

`(<GROUPID> <ID>)` A list whose first element is a GROUPID, and whose second element is the ID of an item in that group. This way items with similar purposes, and thus similar ID's, can be distinguished across groups.

`(GROUP <ID>)` A list whose first element is the keyword GROUP, and whose second element is an item ID. This form describes an item with ID, in the same group that this item is in. This way you don't need to know the GROUPID, just which group you're in.

Then after the entire menu is built, the links are setup, turning the Link Descriptions into actual pointers to Free Menu Items. There is no reason why circular Item Links cannot be created, although such a link would probably not be very useful. If circular links are created, the Free Menu will not be garbage collected after it is not longer being used. The application is responsible for breaking any such links that it creates.

## Free Menu Window Properties

FM.PROMPTWINDOW Specifies the window that Free Menu should use for displaying the item's messages. If not specified, PROMPTWINDOW is used.

FM.BACKGROUND The background shade of the entire Free Menu. This property can be set automatically by specifying a BACKGROUND argument to the function FREEMENU. The window border must be 4 or greater when a Free Menu background is used, due to the way the Window System handles window borders.

FM.DONTRESHAPE Normally Free Menu will attempt to use empty space in a window by pushing items around to fill the space. When a Free Menu window is reshaped, the items are repositioned in the new shape. This can be disabled by setting the FM.DONTRESHAPE window property.

## Free Menu Interface Functions

(FREEMENU *DESCRIPTION TITLE BACKGROUND BORDER*) [Function]

Creates a Free Menu from a Free Menu Description, returning the window. This function will return quickly unless new display fonts have to be created. See the example above.

## Accessing Macros

These Accessing Macros are provided to allow the application to get and set information in the Free Menu data structures. They are implemented as macros so that the operation will compile into the actual access form, rather than figuring that out at run time.

(FM.ITEMPROP *ITEM PROP {VALUE}*) [Macro]

Similar to WINDOWPROP, this macro provides an easy access to the fields of a Free Menu Item. A handle on the item can be gotten from the Free Menu by the function FM.GETITEM, described below. *VALUE* is optional, and if not given, the current value of the *PROP* property will be returned. If *VALUE* is given, it will be used as the new value for that *PROP*, and the old value will be returned.

When a call to FM.ITEMPROP is compiled, if the *PROP* is known (quoted in the calling form), the macro figures out what field to access, and the appropriate Data Type access form is compiled. However, if the *PROP* is not known at compile time, the *function* FM.ITEMPROP, which goes through the necessary property selection at run time, is compiled.

The TYPE and USERDATA properties of a Free Menu Item are Read Only, and an error will result from trying to change the value of one of these properties.

(FM.GROUPPROP *WINDOW GROUP PROP {VALUE}*) [Macro]

Provides access to the Group Properties set up in the PROPS list for each group in the Free Menu Description. *GROUP* specifies the ID of the desired group, and *PROP* the name of the desired property. If *VALUE* is specified, it will become the new value of the property, and the old value will be returned. Otherwise, the current value is returned.

(FM.MENUPROP *WINDOW PROP {VALUE}*) [Macro]

Provides access to the group properties of the top-most group in the Free Menu, that is to say, the entire menu. This provides an easy way for the application to attach properties to the menu as a whole, as well as access the Group Properties for the entire menu.

(FM.NWAYPROP *WINDOW COLLECTION PROP {VALUE}*) [Macro]

This macro works just like FM.GROUPPROP, except it provides access to the NWay Collections.

## Accessing Functions

(FM.GETITEM *ID GROUP WINDOW*) [Function]

Get a handle on item *ID* in *GROUP* of the Free Menu in *WINDOW*. This function will search the Free Menu for an item whose ID property matches, or secondly whose LABEL property matches *ID*. If *GROUP* is NIL, then the entire Free Menu is searched. If no matching item is found, NIL is returned.

(FM.GETSTATE *WINDOW*) [Function]

Return in property list format the ID and current STATE of every NWay Collection and item in the Free Menu. If an item's or Collection's state is NIL, then it is not included in the list. This provides an easy way of getting the state of the menu all at once. If the state of only one item or Collection is needed, the application

can directly access the STATE property of that object using the Accessing Macros above. Note that this function can be called when editing is in progress, in which case it will provide the label of the item being edited at that point.

## Changing Free Menus

Many of the following functions operate on Free Menu Items, and thus take the item as an argument. The *ITEM* argument to these functions can be the Free Menu Item itself, or just a reference to the item. In the second case, FM.GETITEM will be used to find the item in the Free Menu.

The reference can be in one of the following forms:

`<ID>` Specifies the first item in the Free Menu whose ID or LABEL property matches <ID>.

`(<GROUPID> <ID>)` Specifies the item whose ID or LABEL property matches <ID> within the group specified by <GROUPID>.

(FM.CHANGELABEL *ITEM NEWLABEL WINDOW UPDATEFLG*) [Function]

This function changes an item's label after the Free Menu has been created. It works for any type of item, and state items will remain in their current state. If the window is open, the item will be redisplayed with its new appearance. *NEWLABEL* can be an atom, a string, or a bit map (except for Edit items), and will be restricted in size by the MAXWIDTH and MAXHEIGHT Item Properties. If these properties are unspecified, the item will be able to grow to any size. *UPDATEFLG* specifies whether or not the regions of the groups in the menu are recalculated to take into account the change of size of this item. The application should not change the label of an Edit item while it is being edited.

The following Item Property is relevant to changing labels:

CHANGELABELUPDATE Exactly like *UPDATEFLG* except specified on the item, rather than as a function paramater.

(FM.CHANGESTATE *X NEWSTATE WINDOW*) [Function]

Programmatically changes the state of items and NWay Collections. *X* is either an item or a Collection name. For items *NEWSTATE* is a state appropriate to the type of the item. For NWay Collections, *NEWSTATE* should be the desired item in the Collection, or NIL to deselect. For Edit and Number items, this function just does a label change. If the window is open, the item will be redisplayed.

(FM.RESETSTATE *ITEM WINDOW*) [Function]

Set an item back to its initial state.

(FM.RESETMENU *WINDOW*) [Function]

Reset every item in the menu back to its initial state.

(FM.RESETSHAPE *WINDOW ALWAYSFLG*) [Function]

Reshapes the window to its full extent, leaving the lower-left corner unmoved. Unless *ALWAYSFLG* is T, the window will only be increased in size as a result of resetting the shape.

(FM.RESETGROUPS *WINDOW*) [Function]

Recalculate the extent of each group in the menu, updating group boxes and backgrounds appropriately.

(FM.HIGHLIGHTITEM *ITEM WINDOW*) [Function]

This function provides a way of programmatically forcing an item to be highlighted. This might be useful for items which have a direct effect on other items in the menu. The item will be highlighted according to its HIGHLIGHT property, as described in the section **Free Menu Item Highlighting**. Note that this highlight is temporary, and will be lost if the item is redisplayed, by scrolling for example.

## Editor functions

(FM.EDITITEM *ITEM WINDOW CLEARFLG*) [Function]

Start editing an Edit or Number item at the beginning of the item, as long as the window is open. This function will most likely be useful for starting editing of an item that is currently the null string. If *CLEARFLG* is set, the item is cleared first.

(FM.SKIPNEXT *WINDOW CLEARFLG*) [Function]

This function causes the editor to jump to the beginning of the next Edit item in the Free Menu. If *CLEARFLG* is set, then the next item will be cleared first. If there is not another Edit item in the menu, this function will simply cause editing to stop. If this function is called when editing is not in progress, editing will begin on the first Edit item in the menu. This function can be called from any process, and can also be called from inside the editor, in a LIMITCHARS function.

(FM.ENDEDIT *WINDOW WAITFLG*) [Function]

Stop any editing going on in *WINDOW*. If *WAITFLG*, then block until the editor has completely finished. This function can be called from another process, or from a LIMITCHARS function.

(FM.EDITP *WINDOW*) [Function]

If an item is in the process of being edited in the Free Menu *WINDOW*, that item is returned. Otherwise, NIL is returned.

## Miscelaneous

(FM.REDISPLAYMENU *WINDOW*) [Function]

Redisplays the entire Free Menu in its window, if the window is open.

(FM.REDISPLAYITEM *ITEM WINDOW*) [Function]

Redisplays a particular Free Menu Item in its window, if the window is open.

(FM.SHADE *X SHADE WINDOW*) [Function]

*X* can be an item, or a group ID. *SHADE* is painted on top of the item or group. Note that this is a temporary operation, and will be undone by redisplaying. For more permanent shading, the application may be able to add a REDEDISPLAYFN and SCROLLFN for the window as necessary to update the shading.

(FM.WHICHITEM *WINDOW POSorX Y*) [Function]

Gets a handle on an item from its known location within the window. If *WINDOW* is NIL, (WHICHW) is used, and if *POSorX* is NIL, the current cursor location is used.

(FM.TOPGROUPID *WINDOW*) [Function]

Return the ID of the top group of this Free Menu.