U2. Menus

UtilIt provides several menu-related commands, most of which are designed to support its system of "labeled" menu items and built-in Font, Size, Style, and Color menus.

Labeled Items

Labeled menu items are menu items that have a "label ID" associated with them.   Information about the menu item that is associated with each label ID is stored in a private table maintained by UtilIt. Entries are added or removed from this table using the SetItm2 command.   (This is done by FaceIt, for example, when auto-loading main menus, where the labeled items are ones with "#n" in their titles.)

Any number of menu items can be associated with the same label ID, and SetItm2 can be used to manipulate all items linked to the same label ID at once.   This makes it easy, for example, to have multiple standard "Cut" items in multiple menus since each such Cut item is associated with the same label ID (#13) and can be enabled, disabled, checked, etc., by a single call to SetItm2.

Further information about the various types of label IDs supported by FaceWare modules can be found in the FaceIt Guide under its "Menu Handling" topic.

FSSC Menus

On DoInit, UtilIt initializes Font, Size, Style, and Color menus that are loaded from MENU resources 1116-1119 and have menuIDs 196-199, respectively.   These menus are non-main menus that can be attached to hierarchical menu items in other menus.   Controls in ViewIt windows that support multiple text styles will often support the FSSC menus.   This HelpCt control, for example, supports the FSSC menus in the style menu at the top of this window.

FaceIt and ViewIt automatically process (using SelFSC) menu selections from FSSC menus and post the appropriate messages to control drivers.   The drivers then use FixFSC to update the state of the FSSC menus.   This means that main programmers using FaceIt or ViewIt will rarely need to call either SelFSC or FixFSC.

PopMen 123 a,b,c,d,uString,uMenuID,uMenuItem

Executes the toolbox call PopUpMenuSelect where a = menuID, b = vertical position, c = horizontal position of the top, left corner of the menu (in local coordinates), and d is the item to be initially selected (ignored by palette menus).   uString, uMenuID, and uMenuItem return with the selected item text, menuID, and item number.

NOTE: It's unlikely that you'll ever need to use PopMen with ViewIt windows unless attempting to pop up a special menu (such as a picture palette) in response to a click in an enabled item.

SetItm2 161   a,b,c,d,uResult

The SetItm2 command can be used to manipulate menus and menu items in several different ways.   It also supports "labeled" menu items (discussed above), and is used within all modules that manipulate labeled menu items.   (SetItm2 is the "mother" of all commands - no other command has as much functionality as this one!)

  a = menuID, resource ID, or MenuHandle of target menu*
       or use a = 0 to designate that b is a label ID number
  b = menu operation or extent of operation
     -4 = initialize menu**
     -3 = delete (if necessary) and dispose of menu**
     -2 = delete menu from menu list
     -1 = initialize (if necessary) and insert into list**
      0 = apply to all menu items in menu***,****
    > 0 = apply to this menu item number
       or a label ID number (if a = 0)
  c = menuID before which to insert new menu (if b = -1)†
       or menu item operation (if b > -1):
     1 = set checked state (d = 0 = uncheck, ≠ 0 = check)††
     2 = set enabled state (d = 0 = disable, ≠ 0 = enable)††
     3 = set item text (d = 0 or 1 = uString, 2 = uName, other = Pascal string address)††
     4 = set item mark (d = ASCII value of mark character)
     5 = set item ICON (d = ICON ID = 257 to 511)

6 = set item reduced ICON (d = ICON ID = 257 to 511)
     7 = set item SICN (d = SICN ID = 257 to 511)
     8 = set item style (d = 0 to 255)
     9 = set item command character (d = ASCII value)††
    10 = link to hierarchical menu (d = menuID = 1 to 255)
    11 = set item label ID (d = ± label ID)††,†††
  d = parameter for menu item operation
    (use d = 0 to undo operations 4 through 11)
All calls to SetItm2 return the MenuHandle of the target menu in uResult (or 0 if the menu was not found).
Also note that if one or more menus in the main menu bar are deleted/inserted or disabled/reenabled,
then you should update the bar with a call to "DrawMenuBar" (UtilIt can't do this since it doesn't know
when you're finished making changes that affect the appearance of the bar).
    To understand the power of SetItm2, consider the case of a labeled menu item (#121) that your
program needed to support in several different menus.   First, SetItm2 is called by FaceIt or ViewIt when
initializing your menus, so your labeled items get registered properly with UtilIt.   You can then change all
instances of a label with a single call to SetItm2.   For example, to disable all instances of items with label
ID 121, use,

FaceIt(nil,SetItm2,0,121,2,0);
    Pascal

FaceIt(0L,SetItm2,0L,121L,2L,0L);
/* C */

call FaceIt(0,SetItm2,0,121,2,0)
 !Fortran
where the menus affected would include all main menus, hierarchical or non-main menus, and menu
controls in ViewIt modal and modeless windows.

SetItm2 Notes:
* Do not pass a resource ID or menuID when referring to menu controls within ViewIt windows since each such
control is given its own MenuHandle which is a copy of the linked MENU resource (use GetCtl to get this
MenuHandle in cHiData).   Also note that in cases where SetItm2 needs to generate a resource ID from a menuID, it
assumes the relationship: resource ID = menuID + 900 (= FaceIt's auto-load scheme for main program menus).
** When initializing menus, UtilIt scans the menus for labeled items and processes the items.   When disposing of
menus, UtilIt removes the corresponding entries from its labeled item tables.   You can also use b = -4 with menus
that are already initialized to get UtilIt to process their labeled items, or pass an empty MenuHandle with b = -3 to
force UtilIt to remove entries for a menu that has already been disposed of.
*** If b, c, and d are zero, and the menu is a "picture palette" menu, then the entire palette is redrawn.   This may be
useful in cases where a large number of palette items need redrawing.   In such a case, toolbox calls could be used
to quickly fix the status of the menu items, and then SetItm2 called once to redraw the entire palette.
**** If b = 0 (all items) and c = 2 (enable/disable) then the menu itself is enabled or disabled, not the individual items
within the menu.
† Use c = -1 to insert the menu as a non-main menu (or use a "+" or "-" as the first character of the menu's title).
†† These operations also work with "picture palette" menus.
††† When adding new label entries to its private tables, UtilIt does not check whether the entries already exist
(except in the case where a label ID number is used to change all instances of the label ID).   Do not assign more
than one label to a menu item.

GetItm   162   a,b,c

    Parameters a and b define a menu item in the same way as they are used with SetItm2.   Information
about this item is returned as,
  uString = menu item text
  uMenuID = menuID number
  uMenuItem = menu item number
  uResult = label number (±1 to ±7499, 0 if unlabeled)
  uI1 = mark character (ASCII value)
  uI2 = ICON or SICN resID (257-511)

uStyle = menu item style
　uMenuHdl = MenuHandle
where an empty string or zero is returned if the item is not found, and the label number in uResult will be negative if it was entered as such in the MENU resource.
　A typical use for this command might be to get the menuID and menu item number of an item specified by a label ID number (a = 0, b = label ID).　In this case you can also use parameter c to specify the nth instance of the labeled item that you are looking for.

FixFSC　163　a,b,c,d

　Notifies UtilIt that its built-in Font, Size, Style, and/or Color (FSSC) menus need updating according to parameters a, b, c, and d.　Pass -1 for parameters to be ignored, and -2 to uncheck all items in the corresponding menu (which will also occur if the parameter does not correspond to any item in the menu). This command is most often used from within control drivers to update the state of the FSSC menus to reflect the current selection's text style and color.
　a = font number OR address of Pascal string containing font name (outlining of sizes in Size menu is also updated)
　b = size (12 pt. if b = 0)
　c = style = sum of following constants:　0 = Plain,
　　1 = Bold, 2 = Italic, 4 = Underline, 8 = Outline,
　　16 = Shadow, 32 = Condensed, 64 = Extended
　d = address of RGB color OR old-style color constant:
　　33 = black, 30 = white, 205 = red, 341 = green,
　　409 = blue, 273 = cyan, 137 = magenta, 69 = yellow

SelFSC　164　a,b

　Processes an FSSC menu selection by comparing the chosen item with the current state of the FSSC menus (set by calling FixFSC), where a is the item's menu ID and b is the item number.　If the selection would change the state of the FSSC menus, then uResult is returned with a non-zero value, and the new font, size, style, or color is placed in the current port's txFont, txSize, txFace, or fgColor fields (or rgbFgColor for color windows).
　In the case of a style change, txFace is set to contain just the newly selected style (Plain OR Bold OR Italic...) so that the selected style can be turned on or off without affecting the other styles in the current text selection.
　This command is rarely used by either main programmers or module developers since FaceIt and ViewIt automatically preprocess most menu selections.