Cortland Menu Manager

Appendix A
Menu Calls

INITIALIZATION AND TERMINATION ROUTINES

BootMmgr Call #1 (not completed)

Called when SetTSPtr is called.

InitMenus Call #2

input: zeropg:WORD - zero page Menu Manager can use, must be on page boundary.
output: None.

Initializes system menu bar with no menus, and makes it the current menu bar.
Calls Desktop in the Window Manager to reserve space for the bar.
Menu Manager opens a grafPort.
Calls DrawMenuBar to draw an empty system menu bar.

TermMenus Call #3

input: None.
output: None.

Closes the Menu Manager's port and frees any allocated menus.

InitPalette Call #47

input: None.
output: None.

Call when you've changed the color palattes. This will reinitize the palettes needed for the color Apple logo in the system menu bar.

MmgrReset Call #5 (not completed)

input: None.
output: None.

Does nothing.

NewMenu Call #45

input: TextColor:WORD - color index for menu text.
Background:WORD - color index for menu background.
MenuString:LONG - pointer to a menu string (see MENU STRING).

output: MenuList:LONG - pointer to menu list, zero if error.

NewMenu allocates space for a menu list and its item. You pass a Menu String which is a text string that describes menu tiles, items, and special flags. See MENU STRING for the format needed. The MenuList returned can then be inserted in the default system menu bar via an **InsertMenu** call.

Call **DisposeMenu** to deallocate the menu list when finished.

DisposeMenu Call #46

input: MenuList:LONG - previously allocated via NewMenu.

output: None.

Frees the memory allocated by **NewMenu**. The menu list will no longer be usable.

FixMenuBar Call #19

input: None.

output: MenuHeight:WORD - height of the menu bar.

This routine will compute standard sizes for your menu bar and menus.

FixMenuBar will search all the menu title fonts and use the tallest one to compute the height of the menu bar, add it to Bar.top, and store it in Bar.bottom. It will set the TitleWidth width for every menu TitleWidth that is given as zero. Finally it will call **CalcMenuSize** for each menu in the menu bar.

CalcMenuSize Call #28

input: newWidth:WORD - number of pixels wide the menu should be, or zero.
newHeight:WORD - number of pixels high the menu should be, or zero.
MenuNum:WORD - menu ID number.

output: None.

This call lets you set menu dimensions, or have the Menu Manager do it. The Menu Manager will calculate the width for you if newWidth is zero and the height if newHeight is zero.

To compute the width the Menu Manager will find the widthest item in the menu plus room for a mark and command key. A default width will be used if the menu does not contain any item text.

To compute the height the Menu Manager will add up the font height of each item plus four, or use the value found in the font index of ItemFlag if bit 14 of ItemFlag is set.

This routine is called for each menu by the Menu Manager when FixMenuBar is called.

USER INTERACTION ROUTINES

MenuSelect Call #43

input: EventRec:LONG - pointer to event record which contains point of button down.
TheBar:LONG - pointer to menu bar, zero for system menu bar.

output: None ('when' field of event record contains return IDs).

Called when the a button goes down on a menu bar (see **FindWindow** if using the Window Manager). The routine will take care of drawing highlighted titles, pulling down menus, and user interaction. This is handled automatically for the system menu bar when using **TaskMaster** in Window Manager.

If a selection is made the low order WORD of the 'when' element in the event record will contain the ID number of the item selected, and the high order WORD will contain the menu's ID number. If there is a selection, the menu's title will be left highlighted. See **HiliteMenu** to redraw the title as normal.

If not selection is made by the user the low order WORD of the 'when' element in the event record will be zero.

TheBar becomes the current menu bar.

MenuKey Call #9

input: EventRec:LONG - pointer to event record which contains the character to check.
TheBar:LONG - pointer to menu bar, zero for system menu bar.

output: None ('when' field of event record contains return IDs).

Maps the given character to the associated menu and item for that character. When you get a key-down event with the Command key held down--or auto-key event, if the command being invoked is repeatable--call **MenuKey** with the character that was typed. **MenuKey** highlights the appropriate menu title if the key matches, and returns Selection.

If a selection is made the low order WORD of the 'when' element in the event record will contain the ID number of the item selected, and the high order WORD will contain the menu's ID number. If there is a selection, the menu's title will be left highlighted. See **HiliteMenu** to redraw the title as normal.

If not selection is made by the user the low order WORD of the 'when' element in the event record will be zero.

There generally there should never be more than one item in the menu list with the same keyboard equivalent, but if there is, **MenuKey** returns the first one it encounters.

The reason bit 8 in the key should be set is to make it easy for your application to determine if the 'Command' key was held down when the key was press. When the event manager returns a key down message you can clear the low byte of event modifier flag and then OR it to the event message (where the key is). This way **MenuKey** will check the flag for you.

Any lower case key is changed to upper case before being used. **MenuKey** will also make some other changes to accommodate the way the key might be packed into the item's record.

TheBar becomes the current menu bar.

CheckFallDown Call #10

input: EventRec:LONG - pointer to event record.
TheBar:LONG - pointer to menu bar, zero for system menu bar.

output: None ('when' field of event record contains return IDs).

This routine should be called if you are using a fall down menu bar. Call **CheckFallDown** everytime you get a null event from the Event Manager. It will check tif the current cursor position is inside the menu bar's fall down area. If it is, Menu Manager will pull down the menu and track the cursor. When the cursor enters an enabled item it will be highlighted, and when the cursor leaves the item it will unhighlighted. The user may make a selection by pressing the mouse button and releasing over the desired item. Or the user may exit the menu by moving the cursor off the menu.

If a selection is made the low order WORD of the 'when' element in the event record will contain the ID number of the item selected, and the high order WORD will contain the menu's ID number. If there is a selection, the menu's title will be left highlighted. See **HiliteMenu** to redraw the title as normal.

If not selection is made by the user the low order WORD of the 'when' element in the event record will be zero.

TheBar becomes the current menu bar.

MenuRefresh Call #11

input: RedrawRoutine:LONG - address of routine in your application.

output: None.

Note: This is called only when using the Menu Manager without the Window Manager.

RedrawRoutine is called when the Menu Manager can not restore the screen under a menu. First the Menu Manager will try to allocate a buffer large enough to save the screen part before it draws the menu. If the buffer is allocated the screen will be restored from it and then deallocate the memory buffer. If the buffer can not be allocated the Menu Manager will try to call the Window Manager (via the call the Window Manager made to MenuRefresh during initialization) to refresh the screen when the menu goes away. If no buffer can be allocated and the Window Manager isn't installed, the Menu Manager will call RedrawRoutine to refresh the screen under the menu.

The RedrawRoutine should look something like this:

```
Refresh     START
;
rect_addr   equ     6       Offset down stack to RECT pointer.
;
                    :
                    :
            Needed operations to redraw the
            screen inside the given RECT.
                    :
                    :
;
;
; Remove the given pointer from the stack:
;
                    lda     0,s             Move the return
    address down
                    sta     4,s              the stack.
                    lda     2,s
                    sta     6,s
;
                    pla                     Move the stack back to
    the return
                    pla                      address.
;
                    rtl                     Return to the Menu
    Manager.
```

DRAWING

DrawMenuBar Call #42

input: None.

output: None.

Draws the current menu bar, along with any menu titles on the bar.

HiliteMenu Call #44

input: Hilite:WORD - FALSE to draw normal, TRUE to highlight the title.
MenuNum:WORD - menu's ID.

output: None.

MenuNum is the the menu's ID. Its title is drawn using the menu bar's BarColor if Hilite is FALSE, or InvertColor if TRUE. Should be called with Hilite FALSE, and the menu ID of the selected menu, after the application has finished acting on a menu selection.

FlashMenuBar Call #12

input: None.

output: None.

This will redraw the entire current menu bar using InvertColor and then again using BarColor.

MENU AND ITEM SHUFFLING

InsertMenu Call #13

input: AddMenu:LONG - address of menu to insert.
InsertAfter:WORD - menu ID, zero to insert at front.

output: None.

Inserts AddMenu into the MenuList after InsertAfter, or at the front of the list if InsertAfter is zero. **DrawMenuBar** should be called to redraw the new menu bar.

DeleteMenu Call #14

input: MenuNum:WORD - menu ID of menu to delete.

output: None.

MenuNum is take out of MenuList. **DrawMenuBar** should be called to redraw the new menu bar.

InsertItem Call #15

input: AddItem:LONG - address of item to insert.
InsertAfter:WORD - item ID, zero to add to front.
MenuNum:WORD - menu's ID.

output: None.

Inserts an item into the ItemList after InsertAfter, or at the front of the list if InsertAfter is zero. Call **CalcMenuSize** to resize the menu if needed afterward.

DeleteItem Call #16

input: ItemNum:WORD - item ID of item to delete.
MenuNum:WORD - menu's ID.

output: None.

ItemNum is taken out of ItemList. Call **CalcMenuSize** to resize the menu if needed afterward.

MENU BAR ACCESS

SetSysBar Call #18

input: NewBar:LONG - pointer to new system menu bar.

output: None.

Address of new system menu bar is given. If you want the system menu bar to be the default, pass zero for NewBar. The system menu bar becomes the current menu bar.

GetSysBar Call #17

input: None.

output: SysMenu:LONG - pointer to the system menu bar.

Returns a pointer to the system menu bar.

SetMenuBar Call #57

input: TheBar:LONG - pointer to new system menu bar.

output: None.

Address of menu bar to make current is given. If you want the system menu bar to be the current menu bar, pass zero for TheBar.

GetMenuBar Call #58

input: None.

output: CurBar:LONG - pointer to the system menu bar.

Returns a pointer to the current menu bar.

CountMItems Call #20

input: MenuNum:WORD - menu's ID.

output: NumOfItems:WORD - number of items in menu.

Returns the number of items, including any dividing lines, in the menu.

SetFallArea Call #21

input: FallHeight:WORD - number of pixels in fall down area.

output: None.

FallHeight is the number of pixels down from the top of the menu bar the fall down area should be.

GetFallArea Call #22

input: None.

output: FallHeight:WORD - number of pixels in fall down area.

Returns the height of the fall down area.

SetBarColors Call #23

input: NewBarColor:WORD - normal bar color.
NewInvertColor:WORD - selected bar color.
NewOutColor:WORD - Outline color in bits 7-4.

output: None.

NewBarColor: bits 0-3 = text color when normal.
bits 4-7 = background color when normal.

NewInvertColor: bits 0-3 = text color when selected.
bits 4-7 = background color when selected.

NewOutColor: bits 4-7 = color of menu bar outline, menu outline, underlines, and dividing lines

GetBarColors Call #24

input: None.

output: Colors:LONG - colors of menu bar.

Returned menu bar colors are returned in one LONG of which bits 7-0 is BarColor, 15-8 is InvertColor, and 23-16 is Outline. See CREATING A MENU IN YOUR PROGRAM to see what these colors affect.

SetTitleStart Call #25

input: XStart:WORD - menu bar title starting position.

output: None.

XStart is the number of pixels from the left side of the menu bar that the titles should start. For square cornered menu bars this value should be 1. Zero will over write the left side outline of the menu bar. 127 is the maximum value.

GetTitleStart Call #26

input: None.

output: XStart:WORD - menu bar title starting position.

XStart is the number of pixels from the left side of the menu bar that the titles start from.

MENU RECORD ACCESS ROUTINES

GetMenuPtr Call #27

input: LookFor:WORD - search flag.
MenuNum:WORD - menu ID.

output: MenuPtr:LONG - pointer to menu asked for, zero of error.

Returns a pointer to a MENU record.

If LookFor = 0: A pointer to the menu with the given ID will be returned.
If LookFor = 1: A pointer to MenuNum.MenuList is returned (head of list).
If LookFor = 2: A pointer to the last menu in the menu bar is returned.
If LookFor = neg: A pointer to the menu in front of the given menu ID is returned.

SetTitleWidth Call #29

input: NewWidth:WORD - new width of title.
MenuNum:WORD - menu ID.

output: None.

Sets the width of a title. This is the area where the user can select a menu and the area that is inverted when the title is highlighted.

GetTitleWidth Call #30

input: MenuNum:WORD - menu ID.

output: TheWidth:WORD - width of title, zero if error.

Returns the width of a title. This is the area where the user can select a menu and the area that is inverted when the title is highlighted.

SetMenuFlag Call #31

input: NewState:WORD - new bit value to set.
FlagMask:WORD - mask to mask-off affected bits.
MenuNum:WORD - menu ID.

output: None.

Standard flag values are:

Flag to Set	FlagMask	NewState
Enable menu.	$FF7F	$0000
Disable menu.	$FF7F	$0080
Normal menu title.	$FFBF	$0000
Invert menu title.	$FFBF	$0040
Redraw to highlight.	$FFDF	$0000
XOR to highlight.	$FFDF	$0020
Text menu.	$FFE7	$0000
Color menu.	$FFE7	$0008
Application defined menu.	$FFE7	$0010

GetMenuFlag Call #32

input: MenuNum:WORD - menu ID.

output: MenuState:WORD - desired bits from MenuFlag.

Returns MenuNum.MenuFlag (see MENU RECORDS for definition).

SetMenuTitle Call #33

input: NewStrg:LONG - Address of string to replace ItemName.
MenuNum:WORD - menu ID.

output: None.

The value in NewStrg is moved into the menu's TitleName.

GetMenuTitle Call #34

input: MenuNum:WORD - menu ID.

output: TheTitle:LONG - pointer to TitleName, same as in if not in list.

Returns a pointer to the title of a menu.

SetMenuID Call #55

input: NewID:WORD - new ID to be assigned.
MenuNum:WORD - current menu ID.

output: None.

The menu is assigned the given ID number.

ITEM RECORD ACCESS ROUTINES

GetItemPtr Call #35

input: LookFor:WORD - search flag.
ItemNum:WORD - item ID.

output: ItemPtr:LONG - pointer to the item asked for, zero if error.

Returns a pointer to an ITEM record.

If LookFor = 0: A pointer to the item with the given ID will be returned.
If LookFor = 1: A pointer to MenuPtr.ItemList is returned (head of list).
If LookFor = 2: A pointer to the last item in the item list is returned.
If LookFor = neg: A pointer to the item in front of the given item ID is returned.

SetItem Call #36

input: NewStrg:LONG - Address of string to replace ItemName.
ItemNum:WORD - item ID.

output: None.

The item's ItemName pointer is replaced with NewStrg.

GetItem Call #37

input: ItemNum:WORD - item ID.

output: ItemStrg:LONG - pointer to ItemName, or zero if not in list.

Returns a pointer to an item's text string.

EnableItem Call #48

input: ItemNum:WORD - item ID.

output: None.

Item will appear as normal and selectable.

DisableItem Call #49

input: ItemNum:WORD - item ID.

output: None.

Item will appear dimmed and will not be selectable.

CheckItem Call #50

input: Checked:WORD - TRUE to check item, FALSE to uncheck item.
ItemNum:WORD - item ID.

output: None.

Item will appear with a check mark to the left of the item's text, or nothing will appear if Checked is zero.

SetItemMark Call #51

input: Mark:WORD - character to mark item with, zero for no mark.
ItemNum:WORD - item ID.

output: None.

Item will appear with the character given to the left of the item's text, or the mark will not appear if Mark is zero.

GetItemMark Call #52

input: ItemNum:WORD - item ID.

output: Mark:WORD - character that marks item, zero = no mark.

SetItemStyle Call #53

input: ChStyle:WORD - text style to use on item's text.
ItemNum:WORD - item ID.

output: None.

Bits in ChStyle are set to enable special text drawing. Bits affected are:

$0001 - **Bold.**
$0002 - *Italic.*
$0004 - Underscore.

GetItemStyle Call #54

input: ItemNum:WORD - item ID.

output: ChStyle:WORD - text style to use on item's text.

Bits in ChStyle are set to enable special text drawing. Bits affected are:

$0001 - **Bold.**
$0002 - *Italic.*
$0004 - Underscore.

SetItemFlag Call #38

input: NewValue:WORD - new bits to set.
ItemNum:WORD - item ID.

output: None.

This call is used to set desire states of an item. Input flags are:

Function	NewValue
Underline item.	$0040
Not underline an item.	$FFBF
Use XOR highlighting.	$0020
Use redraw highlighting.	$FFDF

GetItemFlag Call #38

input: ItemNum:WORD - item ID.

output: Divide:WORD - current underline value.
XOR:WORD - current highlighting method.

Outputs are:

OldDivide	0 = no underline	1 = underline
OldXOR	0 = redraw to highlight	1 = XOR to highlight

SetItemID Call #56

input: NewID:WORD - new ID to be assigned.
ItemNum:WORD - current item ID.

output: None.

The item is assigned the given ID number.

SetItemBlink Call #40

input: Count:WORD - number of times item should blink when selected.

output: None.

This call affects all menu bars, system and window. When an enabled item is selected by the user the item blinks briefly to conform the chioce. Normally, your application shouldn't be concerned with this blinking; the user sets it with the Control Panel desk accessory. If you're writing a desk accessory like the Control Panel. though, **SetItemBlink** allows you to control the duration of the blinking. The Count parameter is the number of times menu items will blink.

MISCELLANEOUS ROUTINES

MNewRes Call #41

input: None.

output: None.

Called when the screen resolution changes. Menu Manager makes needed adjustments for the new resolution and redraws the current system menu bar.

MmgrVersion Call #4

input: None.

output: version:WORD - Menu

Manager's version number.

Initialization

Termination

User Interaction

Drawing

Shuffling

Menu Bar Access

Menu Access

Item Access

INDEX