Cortland Menu Manager

Appendix A
Menu Calls

INITIALIZATION AND TERMINATION ROUTINES

MenuBootInit *(not completed)*

input: None.
output: None.

Called when SetTSPtr is called.

MenuStartup

input: userID:WORD - user ID that the Menu Manager can use, mainly to allocate memory.
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.

MenuShutDown

input: None.
output: None.

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

MenuVersion

input: None.

output: version:WORD - Menu

Manager's version number.

MenuReset *(not completed)*

input: None.
output: None.

Does nothing.

NewMenuBar

inputs: theWindow:LONG - pointer to window's port, owner of menu bar, zero for system.

output: BarHandle:LONG - handle of menu bar.

NewMenuBar will create a default menu bar with no menus. **MenuStartup** calls **NewMenuBar** to create a default system menu bar. If you are only going to use one system menu bar, **NewMenuBar** will not have to be called. The default size of the menu bar is, upper left corner matches the port, and the width is the width is the width of the screen. The height of the bar is 13. The menu bar is visible and has default colors of black text on a white background.

NewMenu

input: MenuString:LONG - pointer to a menu/item line list.

output: MenuHandle:LONG - handle of menu, zero if error.

NewMenu allocates space for a menu and its items. You pass a pointer to a menu/item line list which is a text string that describes the menu title and its items. See MENU LINES AND ITEM LINES for the format needed. The MenuHandle returned can then be inserted in a menu bar via an **InsertMenu** call.

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

DisposeMenu

input: MenuHandle:LONG - previously allocated via NewMenu.

output: None.

Frees the memory used by MenuHandle. The menu will no longer be usable.

Warning: The menu is not taken out of the menu list, call **DeleteMenu** to do that. To delete a menu from the menu list and free it's memory you could do something like this:

```
pha             Space for returned handle.
pha
pea  MenuID     ID of menu to delete.
_GetMHandle     Get the handle of the menu.
                Leave menu handle on stack.

pea  MenuID     ID of menu to delete from list.
_DeleteMenu     Delete menu from list.

                Handle still on stack.
_DisposeMenu    Deallocate menu record.
```

FixMenuBar

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

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

input: TaskRec:LONG - pointer to Task record which contains point of button down.
BarHandle:LONG - handle of menu bar, zero for system menu bar.

output: None ('TaskData' field of Task 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 'TaskData' element in the Task 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 no selection is made by the user the low order WORD of the 'TaskData' element in the Task record will be zero.

BarHandle becomes the current menu bar.

The structure of TaskRec is:

what	WORD	Event record portion, unchanged from GetNextEvent.
message	LONG	
when	LONG	
where	LONG	
modifiers	WORD	
TaskData	LONG	Extended portion for TaskMaster.

MenuKey

input: TaskRec:LONG - pointer to Task record which contains the character to check.
BarHandle:LONG - handle of menu bar, zero for system menu bar.

output: None ('TaskData' field of Task 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 a pointer to a Task record that contains the character and the state of the modifier keys (the format is the same as an Event record, see Event Manager). The match will only occur if it is indicated in the Task record that the command key was down. **MenuKey** highlights the appropriate menu title if the key matches, and returns Selection.

The items are searched starting with the first menu in the menu list and all the items in the menu starting with the first. Then the second menu, and so on. The given key is compare with every item's primary keyboard equivalent of every item. If no match is found, the cycle is repeated, this time comparing to each item's alternate keyboard equivalent.

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.

MenuKey will not convert lower case characters to upper case. If you want to match on either upper or lower case, set the primary character to the upper case character and the alternate to the lower case character.

If a selection is made the low order WORD of the 'TaskData' element in the Task 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 no selection is made by the user the low order WORD of the 'TaskData' element in the Task record will be zero.

BarHandle becomes the current menu bar.

See **MenuSelect** for a description of TaskRec.

MenuRefresh

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

input: None.

output: None.

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

HiliteMenu

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 normal color if Hilite is FALSE, or hilite color if TRUE. **HiliteMenu** 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

input: None.

output: None.

This will redraw the entire current menu bar using the bar's hilite color and then again using its normal color.

MENU AND ITEM SHUFFLING

InsertMenu

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

output: None.

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

DeleteMenu

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

output: None.

MenuNum is take out of current menu bar. **DrawMenuBar** should be called to redraw the new menu bar after **DeleteMenu.** The menu is not deallocated, call **DisposeMenu** to do that.

InsertItem

input: AddItem:LONG - address of item line to insert.
InsertAfter:WORD - item ID, zero to add to front, $FFFF to append to end of menu.
MenuNum:WORD - menu ID number to add item to, zero for first menu.

output: None.

Inserts an item into the ItemList after InsertAfter. If InsertAfter is zero, the item will be inserted at the front of MenuNum. If InsertAfter is $FFFF, the item will be appended at the end of MenuNum. If MenuNum is zero, the menu will be considered the first menu. Call **CalcMenuSize** to resize the menu if needed afterward. See MENU LINES AND ITEM LINES for the definition of an item line.

DeleteItem

input: ItemNum:WORD - item ID of item to delete.

output: None.

ItemNum is taken out of ItemList of its menu in the current menu bar. Call CalcMenuSize to resize the menu if needed afterward.

MENU BAR ACCESS

SetSysBar

input: BarHandle:LONG - handle of new system menu bar.

output: None.

Handle of new system menu bar is given. The system menu bar becomes the current menu bar.

GetSysBar

input: None.

output: BarHandle:LONG - handle of the system menu bar.

Returns the handle of the system menu bar.

SetMenuBar

input: BarHandle:LONG - handle of current menu bar.

output: None.

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

GetMenuBar

input: None.

output: BarHandle:LONG - handle of current menu bar.

Returns the handle of the current menu bar.

CountMItems

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.

SetBarColors

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

output: None.

Normal Color: bits 0-3 = text color when not selected.
bits 4-7 = background color when not selected.
bits 8-15 = zero.
Negative to not change normal color.

Hilite Color: bits 0-3 = text color when selected.
bits 4-7 = background color when selected.
bits 8-15 = zero.
Negative to not change hilite color.

Outline Color: bits 0-3 = zero.
bits 4-7 = color of menu bar outline, menu outline, underlines, and dividing lines
bits 8-15 = zero.
Negative to not change outline color.

Color of current menu bar is set to given values that are not negative. Call DrawMenuBar to draw menu bar in new colors.

GetBarColors

input: None.

output: Colors:LONG - colors of menu bar.

Returned menu bar colors are returned in one LONG of which:

bits 31-24 = zero.
bits 23-18 = color of menu bar outline, menu outline, underlines, and dividing lines
bits 19-16 = zero.
bits 15-12 = background color when selected.
bits 11-8 = text color when selected.
bits 7-4 = background color when not selected.
bits 3-0 = text color when not selected.

SetTitleStart

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. Xstart should be at least 1. Zero will over write the left side outline of the menu bar. 127 is the maximum value allowed.

GetTitleStart

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

GetMHandle

input: MenuNum:WORD - menu ID.

output: MenuHandle:LONG - handle of menu, zero if error.

Handle of menu with an ID number that matches menuNum is returned, or zero if the menu is not found.

SetTitleWidth

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

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

input: NewValue:WORD - new bit value to set or clear.
MenuNum:WORD - menu ID.

output: None.

Possible NewValues:

EnableMenu	$FF7F	Menu will not be dimmed and will be selectable.
DisableMenu	$0080	Menu will be dimmed and not selectable.
UnhiliteMenu	$FFBF	Menu will appear in its highlighted state.
HiliteMenu	$0040	Menu will appear in its normal (unhighlighted) state.
ColorReplace	$FFDF	The menu's title and background will be redrawn to hilite.
XORhilite	$0020	The menu's title area will be XORed to hilite.
StandardMenu	$FFE7	The menu will be considered a standard menu.
CustomMenu	$0010	The menu will be considered a custom menu.

If you change a flag that affects the appearance of a menu title you should also call **DrawMenuBar** after **SetMenuFlag** to redraw the titles in their new state.

GetMenuFlag

input: MenuNum:WORD - menu ID.

output: MenuState:WORD - desired bits from MenuFlag.

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

SetMenuTitle

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

input: MenuNum:WORD - menu ID.

output: TheTitle:LONG - pointer to TitleName.

Returns a pointer to the title of a menu.

SetMenuID

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

SetItem

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

input: ItemNum:WORD - item ID.

output: ItemStrg:LONG - pointer to ItemName.

Returns a pointer to an item's text string.

EnableItem

input: ItemNum:WORD - item ID.

output: None.

Item will appear as normal and selectable.

DisableItem

input: ItemNum:WORD - item ID.

output: None.

Item will appear dimmed and will not be selectable.

CheckItem

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

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

input: ItemNum:WORD - item ID.

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

SetItemStyle

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.

Bits 0-2 of chStyle are all used to set the item's text style. For example:

chStyle = $0007	The item is printed as bold italic and underscored.
chStyle = $0005	The item is printed as bold and underscored.
chStyle = $0000	The item is printed as plain (no bold italic or underscore).

GetItemStyle

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

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

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

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

output: None.

The item is assigned the given ID number.

SetItemBlink

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

GetMenuMgrPort

input: None.

output: MenuMgr:LONG - pointer to Menu Manager's port.

Getting the Menu Manager's port might be useful if you would like to change its font.

MNewRes

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.

InitPalette

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.

Constants

Masks for MenuFlag:

M_INVIS	$04	FASLE if menu is visible *(not completed)*.
M_STANDARD	$10	FALSE if menu is a standard (not custom) menu.
M_NO_XOR	$20	TRUE if menu title is highlighted using XOR.
M_NORMAL	$40	TRUE if menu title is highlighted.
M_ENABLED	$80	FALSE if menu is disabled.

mDrawMsg	0	Draw menu command.
mChooseMsg	1	Hit test item command.
mSizeMsg	2	Compute menu size command.
mDrawTile	3	Draw menu's title command.

Posiible inputs to **SetMenuFlag:**

EnableMenu	$FF7F	Menu will not be dimmed and will be selectable.
DisableMenu	$0080	Menu will be dimmed and not selectable.
UnhiliteMenu	$FFBF	Menu will appear in its highlighted state.
HiliteMenu	$0040	Menu will appear in its normal (unhighlighted) state.
ColorReplace	$FFDF	The menu's title and background will be redrawn to hilite.
XORhilite	$0020	The menu's title area will be XORed to hilite.
StandardMenu	$FFE7	The menu will be considered a standard menu.
CustomMenu	$0010	The menu will be considered a custom menu.

Possible NewValue input to **SetItemFlag:**

UnderItem	$0040	Underline item.
NoUnderItem	$FFBF	Do not underline item.
XORHilite	$0020	Use XOR highlighting on item.
NoXORHilite	$FFDF	Use redraw highlighting on item.

Data Types

TaskRec (TaskMaster record):

what	0	Integer	Same as event record.
message	2	LongInt	Same as event record.
when	6	LongInt	Same as event record.
where	10	LongInt	Same as event record.
modifiers	14	Integer	Same as event record.
TaskData	16	LongInt	Return for ID numbers.
TaskMask	20	Integer	Unused.
TASKREC_SIZE	22		Size of TaskRec.

MENUBAR (Menu Bar Record):

CtrlNext	0	Handle	Not used.
CtrlOwner	4	Pointer	Pointer to menu bar's window.
CtrlRect	8	RECT	Enclosing rectangle.
CtrlFlag	16	Byte	Bit flags.
CtrlHilite	17	Byte	Not used.
CtrlValue	18	Integer	Not used.
CtrlProc	20	Pointer	Not used.
CtrlAction	24	Pointer	Not used.
CtrlData	28	LongInt	Reserved for CtrlProc's use.
CtrlRefCon	32	LongInt	Reserved for application's use.
CtrlColor	36	Pointer	Menu bar's color table.
MenuList	40	Handle[]	Menu bar's color table.

MENU (Menu record):

MenuID	0	Integer	Menu's ID number.
MenuWidth	2	Integer	Width of menu.
MenuHeight	4	Integer	Height of menu.
MenuProc	6	Pointer	Menu's definition procedure.
MenuFlag	10	Byte	Bit flags.
TitleWidth	11	Integer	Width of menu's title.
TitleName	13	Pointer	Menu's title.