Testfile

COLLABORATORS

	TITLE : Testfile	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		April 15, 2022	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1 Testfile **1**

1.1 Table Of Contents . . . 1
1.2 popupmenu.library/--background-- . . . 2
1.3 popupmenu.library/--hooks-- . . . 2
1.4 popupmenu.library/PM_AbortHook . . . 3
1.5 popupmenu.library/PM_AlterState . . . 4
1.6 popupmenu.library/PM_ExLstA . . . 4
1.7 popupmenu.library/PM_FilterIMsgA . . . 5
1.8 popupmenu.library/PM_FindItem . . . 6
1.9 popupmenu.library/PM_FreePopupMenu . . . 6
1.10 popupmenu.library/PM_GetItemAttrsA . . . 7
1.11 popupmenu.library/PM_GetVersion . . . 7
1.12 popupmenu.library/PM_InsertMenuItemA . . . 8
1.13 popupmenu.library/PM_ItemChecked . . . 9
1.14 popupmenu.library/PM_LayoutMenuA . . . 9
1.15 popupmenu.library/PM_MakeIDListA . . . 10
1.16 popupmenu.library/PM_MakeItemA . . . 11
1.17 popupmenu.library/PM_MakeMenuA . . . 14
1.18 popupmenu.library/PM_OpenPopupMenuA . . . 14
1.19 popupmenu.library/PM_ReloadPrefs . . . 16
1.20 popupmenu.library/PM_RemoveMenuItem . . . 16
1.21 popupmenu.library/PM_SetItemAttrsA . . . 17

Chapter 1

Testfile

1.1 Table Of Contents

TABLE OF CONTENTS

--background--

--hooks--

PM_AbortHook

PM_AlterState

PM_ExLstA

PM_FilterIMsgA

PM_FindItem

PM_FreePopupMenu

PM_GetItemAttrsA

PM_GetVersion

PM_InsertMenuItemA

PM_ItemChecked

PM_LayoutMenuA

PM_MakeIDListA

PM_MakeItemA

PM_MakeMenuA

PM_OpenPopupMenuA

```
PM_ReloadPrefs

PM_RemoveMenuItem

PM_SetItemAttrsA
```

1.2 popupmenu.library/--background--

```
PURPOSE
     The popupmenu.library provides developers with an easy way of adding
     user configurable popup and pulldown menus to their applications.

COPYRIGHT
     The popupmenu.library is ©2000 by Henrik Isaksson.
```

1.3 popupmenu.library/--hooks--

```
CALLBACK HOOKS
     Callback hooks are used by popupmenu.library for several purposes,
     including handling of localized strings, handling of user input (ie.
     menu selection) and building dynamic menus. All of theese hooks have
     some things in common, and they will be described here.

     For general information about callbacks hooks, see the
     utility.library documentation and utility/hooks.h.

     When a callback hook function is called by popupmenu.library, the
     register contents are as follows:

     o a pointer to the functions hook structure is in A0.

     o a pointer to the affected object is put in A2. This is usually a
       struct PopupMenu pointer. Some future functions may use other
       objects.

     o a pointer to an array of arguments is passed in A1. Theese are
       specific to each hook.

EXAMPLE

     /* First we declare the function: */
     /* Note that the keywords __asm, __saveds, register, etc. are */
     /* specific to SAS/C. */

     ULONG __saveds __asm MyPMHookFunc(
                     register __a0 struct Hook *hook,
                     register __a2 struct PopupMenu *pm,
                     register __a1 APTR *args)
     {
             /* hook = pointer to the hook structure declared below. */
```

```
            /* pm = pointer to the item that the hook function call */
            /*      is caused by. */

            /* args = array of arguments. */

            /* Example: You want to access the first argument which is */
            /*          an ULONG. */

            ULONG arg1 = *((ULONG *)args[0]);
    }

    /* Second: Filling out the Hook structure. */

    struct Hook MyPMHook;

    MyPMHook.h_Entry = (HOOKFUNC) MyPMHookFunc;

    /* Now you can pass the structure as a pointer to one of the */
    /* popupmenu functions. */

SEE ALSO
    PM_MakeItemA(), PM_OpenPopupMenuA(), utility.library/CallHookPkt(),
    <utility/hooks.h>
```

1.4 popupmenu.library/PM_AbortHook

```
NAME
    PM_AbortHook -- Find out if user wants to abort menu operation. (V9)

SYNOPSIS
    abort = PM_AbortHook(handle);
    D0                   A0

    BOOL PM_AbortHook(APTR);

FUNCTION
    This function is used to find out if the user has moved the mouse
    pointer outside of the region which opens the menu. It should be
    called periodically (as often as possible).
    PM_AbortHook() may redraw the menu when it is called, if the delay
    since last call has passed a certain treshold (default is 80 ms).

INPUTS
    handle - pointer to private data passed to the hook function through
             a pointer in A1.

RESULT
    Returns TRUE if you should stop doing whatever you are doing, and
    return.
    If FALSE is returned, just keep on like nothing happened.

EXAMPLE

    struct PopupMenu * __saveds __asm SubConstructFunc(
            register __a0 struct Hook *hook,
```

```
            register __a2 struct PopupMenu *selected,
            register __a1 APTR *handle)
    {
            BOOL abort=FALSE;

            do {

                    /* Do something to the menu */

                    abort = PM_AbortHook(*handle); /* Abort? */

            } while(!abort);
    }

SEE ALSO
    PM_MakeItemA(), <libraries/pm.h>
```

1.5 popupmenu.library/PM_AlterState

```
NAME
    PM_AlterState -- Simulate a (de-)selection of a checked item. (V5)

SYNOPSIS
    PM_AlterState(base, ids, action);
                  A1    A2   D1

    void PM_AlterState(struct PopupMenu *pm, struct PM_IDLst *ids,
                    UWORD action);

FUNCTION
    This function will change the state of checkable/mx items in the
    list 'ids', depending on their kind (exclude/include/reflect/inverse)
    and the value of 'action'.

INPUTS
    base   - pointer to a PopupMenu structure.
    ids    - linked PM_IDLst list.
    action - either PMACT_DESELECT (2) or PMACT_SELECT (3).
```

1.6 popupmenu.library/PM_ExLstA

```
NAME
    PM_ExLstA -- Build a list of ID numbers for mutual exclusion. (V6)

SYNOPSIS
    list = PM_ExLstA(id);
    D0               A1

    list = PM_ExLst(id1, ...);

    struct PM_IDLst *PM_ExLstA(ULONG *id);
```

```
     struct PM_IDLst *PM_ExLst(ULONG id1, ...);

FUNCTION
     This function is used to build lists of ID numbers, that are needed
     for the PM_Exclude tag.
     The array of ID's is ended by NULL.

INPUTS
     id - array of ID numbers

RESULT
     Returns a list of ID's, or NULL if it ran out of memory.
     You don't really need to care about what it returns, just pass it
     on to PM_MakeItemA().

SEE ALSO
     PM_MakeIDListA(), PM_MakeItemA(), <libraries/pm.h>
```

1.7 popupmenu.library/PM_FilterIMsgA

```
NAME
     PM_FilterIMsgA -- Handle keyboard shortcuts. (V6)

SYNOPSIS
     userdata = PM_FilterIMsgA(window, menu, imsg, tags);
     D0                        A1      A2    A3    A5

     userdata = PM_FilterIMsg(window, menu, imsg, tags, ...);

     APTR PM_FilterIMsgA(struct Window *, struct PopupMenu *,
             struct IntuiMessage *, struct TagItem *);

     APTR PM_FilterIMsg(struct Window *, struct PopupMenu *,
             struct IntuiMessage *, ULONG, ...);

FUNCTION
     This function handles keyboard shortcuts.
     It compares 'imsg' against IDCMP_VANILLAKEY, if one is found, that
     item's UserData is returned, or the MenuHandler hook is called.
     Remember to set IDCMP_VANILLAKEY for the window(s) you use for user
     input.

INPUTS
     window = pointer to the parent window.
     menu   = pointer to a popup menu.
     imsg   = IntuiMessage to be filtered.

TAGS
     Accepts the same tags as PM_OpenPopupMenuA, in addition to those
     listed below:

     PM_AutoPullDown - (BOOL) Set this to TRUE if you want a pulldown
                       menu opened automatically, when the user presses
                       RMB.
                       You will have to use WFLG_RMBTRAP and
```

```
                        IDCMP_MOUSEBUTTONS to get it working.

   RESULT
        The UserData of the item that was selected, or NULL.
        If MultiSelect is enabled, this result should not be used, since it
        would not be reliable when the user selects several items.
        (The user can ofcourse only select more than one item if the tag
        PM_AutoPullDown is used)

   SEE ALSO
        PM_OpenPopupMenuA()
```

1.8 popupmenu.library/PM_FindItem

```
   NAME
        PM_FindItem -- Find an item in a popupmenu list. (V3)

   SYNOPSIS
        item = PM_FindItem(menu, id);
        D0                 A1    D1

        struct PopupMenu *PM_FindItem(struct PopupMenu *, ULONG);

   FUNCTION
        Find the pointer to an item using the ID number.

   INPUTS
        menu = pointer to a popup menu list.
        id   = ID number (PM_ID).

   RESULT
        Returns a pointer to the found item, or NULL if unsuccessful.

   SEE ALSO
```

1.9 popupmenu.library/PM_FreePopupMenu

```
   NAME
        PM_FreePopupMenu -- Free a menu list created by PM_MakeMenuA()
 PM_MakeMenuA}.

   SYNOPSIS
        PM_FreePopupMenu(popupmenu);
                         a1

        void PM_FreePopupMenu(struct PopupMenu *);

   FUNCTION
        This function is used to free the list of menu items created by
        PM_MakeItemA(), and PM_MakeMenuA().

   INPUTS
```

```
     popupmenu - pointer to a popup menu to free.

SEE ALSO
     PM_MakeItemA(), PM_MakeMenuA()
```

1.10 popupmenu.library/PM_GetItemAttrsA

```
NAME
     PM_GetItemAttrsA -- Get attribute values for an object. (V3)

SYNOPSIS
     result = PM_GetItemAttrsA(item, tags);
     D0                        A2    A1

     ULONG PM_GetItemAttrsA(struct PopupMenu *, struct TagItem *);

     result = PM_GetItemAttrs(item, tag1, ...);

     ULONG PM_GetItemAttrs(struct PopupMenu *, ULONG, ...);

FUNCTION
     Used to get attributes from an item.
     item can be directly taken from PM_FindItem() as the input is
     checked against NULL pointers.

EXAMPLE

     struct PopupMenu *menu;
     struct Image *image;
     BOOL checked;

     ....
     /* Initialize the menu */
     ....

     PM_GetItemAttrsA(PM_FindItem(menu, itemid),
             PM_SelectImage, &image,
             PM_Checked,     &checked,
             TAG_DONE);

INPUTS
     item = pointer to a popup menu item.
     tags = array of TagItem structures with attribute/value pairs.

RESULT
     Returns the number of successfully copied attributes.

SEE ALSO
     PM_FindItem()
```

1.11 popupmenu.library/PM_GetVersion

```
NAME
     PM_GetVersion -- Get the library version string (V9)

SYNOPSIS
     version = PM_GetVersion();
     D0

     STRPTR PM_GetVersion(void);

FUNCTION
     This function is used by the preferences program to find out which
     settings that are supported by the library.

RESULT
     A version string containing version number & date.

EXAMPLE

     puts(PM_GetVersion());

     Outputs the string:
     "$VER: popupmenu.library 9.0 (01.05.00)"
```

1.12 popupmenu.library/PM_InsertMenuItemA

```
NAME
     PM_InsertMenuItemA -- Insert menu items in a menu after creation.

SYNOPSIS
     success = PM_InsertMenuItemA(menu, tags);
                                  a0    a1

     success = PM_InsertMenuItem(menu, tag1, ...);

     LONG PM_InsertMenuItemA(struct PopupMenu *, struct TagItem *);

     LONG PM_InsertMenuItem(struct PopupMenu *, ULONG tag1, ...);

FUNCTION
     This function inserts one or more items in a menu, after it has
     been created with PM_MakeMenuA().

INPUTS
     menu - pointer to the menu that will recieve the new items.
     tags - see below.

RETURNS
     The function returns the number of succesfully inserted items.
     (0 if none was inserted)

TAGS
     PM_Insert_Item     (struct PopupMenu *)
                        This tag inserts the item pointed to by ti_Data
                        at the position formerly given by one of the
```

```
                        tags below. (see example)

     PM_InsertAfter     (struct PopupMenu *)
                        Insert the item(s) after this item.

     PM_InsertAfterID   (ULONG)
                        Insert the item(s) after an item with this ID.

     PM_InsertSub_First (struct PopupMenu *)
     PM_InsertSub_Last  (struct PopupMenu *)
                        Insert the item(s) at the top/bottom of the
                        submenu pointed to by ti_Data.

EXAMPLE

     PM_InsertMenuItem(mypopupmenu,
             PM_InsertAfterID,   100,
             PM_Insert_Item,     PM_MakeItem(
                                       PM_Title,  "New item",
                                       TAG_DONE),

             PM_InsertSub_First, a_submenu,
             PM_Insert_Item,     another_new_item,
             TAG_DONE);

SEE ALSO
     PM_RemoveMenuItem(), PM_MakeMenuA(), <libraries/pm.h>
```

1.13 popupmenu.library/PM_ItemChecked

```
NAME
     PM_ItemChecked -- Find out if an item is checked. (V3)

SYNOPSIS
     item = PM_ItemChecked(menu, id);
     D0                    A1    D1

     BOOL PM_ItemChecked(struct PopupMenu *, ULONG);

FUNCTION
     Fast way to find out if an item is checked using the item ID.

INPUTS
     menu = pointer to a popup menu list.
     id   = ID number (PM_ID).

RESULT
     TRUE (-1L) if the item is checked, FALSE (0L) if not checked,
     PMERR (-5L) if the ID was not found in the list.
```

1.14 popupmenu.library/PM_LayoutMenuA

```
NAME
     PM_LayoutMenuA -- Prepare the menu for opening. (V9)

SYNOPSIS
     success = PM_LayoutMenuA(window, menu, tags);
     D0                       A0      A1    A2

     success = PM_LayoutMenu(window, menu, tag1, ...);

     BOOL PM_LayoutMenuA(struct Window *, struct PopupMenu *,
                    struct TagItem *);

     BOOL PM_LayoutMenu(struct Window *, struct PopupMenu *, ULONG, ...);

FUNCTION
     Loads and remaps images and lays out the menu. Use with large menus
     that will only be used on one screen. This will speed up the menu
     significantly.

INPUTS
     window = window on the screen you want to open the menu on later. The
              window must remain open until the menu is freed.
              You must not attempt to open the menu on another screen than
              the one the window is on. It may work sometimes, but don't
              count on it.
     menu   = pointer to a popup menu list (PM_MakeMenuA).

TAGS
     No tags have been defined yet.

RESULT
     TRUE (-1L) if succesful, FALSE (0L) otherwise. Even if the function
     would fail, you can still safely call PM_OpenPopupMenu().
```

1.15 popupmenu.library/PM_MakeIDListA

```
NAME
     PM_MakeIDListA -- Create a list of ID's for exclusion/inclusion.

SYNOPSIS
     list = PM_MakeIDListA(taglist);
     d0                    a1

     struct PM_IDLst *PM_MakeIDListA(struct TagItem *tags);

     list = PM_MakeIDList(tag1, ...);

     struct PM_IDLst *PM_MakeIDList(ULONG, ...);

FUNCTION
     This function is used to create a list of ID's that is used to
     tell wich items an item should include, exclude, reflect or
     inverse reflect.
```

```
INPUTS
     taglist - pointer to a taglist.

TAGS
     PM_ExcludeID   (ULONG) ID of a item that should be unselected when
                    when this item is selected.

     PM_IncludeID   (ULONG) ID of a item that should be selected when
                    when this item is selected.

     PM_ReflectID   (ULONG) ID of a item that should copy the state
                    of this item, when it gets selected/unselected.

     PM_InverseID   (ULONG) ID of a item that should copy the inverse
                    state of this item, when it gets selected/
                    unselected.
                    Useful if you want to make sure only one of two
                    items is selected at a time.

RETURNS
     Returns a pointer to a list of id's if successful.

SEE ALSO
     PM_MakeItemA(), PM_ExLstA(), <libraries/pm.h>
```

1.16 popupmenu.library/PM_MakeItemA

```
NAME
     PM_MakeItem -- Create a new menu item.

SYNOPSIS
     menu = PM_MakeItemA(taglist);
     d0                  a1

     struct PopupMenu *PM_MakeItemA(struct TagItem *tags);

     menu = PM_MakeItem(tag1, ...);

     struct PopupMenu *PM_MakeItem(ULONG, ...);

FUNCTION
     This function is used to create a new menu item to be passed to
     PM_MakeMenuA(), for linking.

INPUTS
     taglist - pointer to a taglist listing your menu items.

TAGS
     PM_Title       (STRPTR) Pointer to the menu text you want.

     PM_UserData    (ULONG) Anything of your choice, can be used to
                    identify the item when it is selected. The value
                    stored here will be returned from PM_OpenPopupMenuA()
                    when the user selects this item.
```

```
PM_ID           (ULONG) An ID number, only needed if you want to
                be able to read or change the attributes of this
                item later. (for example, to find out if an item
                is checked)

PM_Sub          (struct PopupMenu *) A pointer to a menu list
                returned from PM_MakeMenuA(). The item
                will automatically get an arrow to the right
                showing that it has a sub menu.

PM_Flags        (ULONG) Used internally. Do not use this tag!

PM_NoSelect     (BOOL) Make the item unselectable.
                If this attribute is set to FALSE for items with
                submenus, they will become selectable. Note that
                this is not currently (V8) reversable.

PM_FillPen      (BOOL) Draw the item title in FILLPEN.

PM_Checkit      (BOOL) Leave some space for a checkmark.

PM_Checked      (BOOL) Put a checkmark to the left of the item.

PM_Italic       (BOOL) Draw the text in italic.

PM_Bold         (BOOL) Make the text bold.

PM_Underlined   (BOOL) Underline the text.

PM_WideTitleBar (BOOL)
PM_TitleBar     (BOOL) Draw a horizontal separator instead of text.

PM_ShadowPen    (BOOL) Draw the text in SHADOWPEN color.

PM_ShinePen     (BOOL) Draw the text in SHINEPEN color.

PM_Exclude      (struct PM_IDLst *) List of items to be selected
                or unselected when this item gets selected.
                The list should be created with PM_MakeIDListA().

PM_Disabled     (BOOL) Makes the item unselectable, and draws a
                disable pattern over the item.

PM_ImageSelected (struct Image *)
PM_ImageUnselected (struct Image *)
                Specifies an image to be rendered under the item
                title.

PM_IconSelected (struct Image *)
PM_IconUnselected (struct Image *)
                Specifies an image to be rendered to the left of the
                item title.

PM_AutoStore    (BOOL *) A pointer to a BOOL that will reflect the
                state of the checkmark. The best way to find out
                if an item is checked or not.
```

```
PM_TextPen      (ULONG) A pen number for the text. You are
                responsible for allocating/deallocating a pen
                yourself.

PM_Shadowed     (BOOL) Give the the text a shadow using SHADOWPEN.

PM_CommKey      (STRPTR) Keyboard shortcut for this item.
                Only the first character will be used. This is a
                string pointer just to make it easier to use locale
                strings for the shortcuts.

PM_ColourBox    (ULONG) Draws a filled rectangle in using the
                specified pen.
                The box will be drawn at the end of the line in
                PM_CheckIt items, and at the beginning in other items.

PM_SubConstruct (struct Hook *)
                This hook will be called before the submenu pointed to
                by PM_Sub is opened. Using this hook you can create the
                menu just before it is opened, and show directory
                contents etc in the menu. Remember to always free the
                pm->Sub pointer before replacing it with the new one.
                The hook function will recieve a pointer to the hook
                structure in A0 and a pointer to the selected (parent)
                menuitem in A2.

PM_SubDestruct (struct Hook *)
                This hook is called after the submenu has been closed,
                and is typically used when you need to free user data.
                The hook function will recieve a pointer to the hook
                structure in A0 and a pointer to the parent menuitem
                in A2. (NOT the submenu!)

PM_Hidden       (BOOL)
                Setting this tag to TRUE will prevent the item from
                being drawn.

PM_TitleID      (ULONG)
                Locale string ID. Will be passed to the GetString
                hook. (see PM_OpenPopupMenuA()).

PM_Object       (Object *)
                A BOOPSI object should be used to render this item.

PM_Members      (PopupMenu *)
                Turns this item into a group, and adds the objects
                pointed to by ti_Data. The list of objects are
                created just like a (sub)menu with PM_MakeMenuA().
                The default layout mode is PML_Horizontal.

PM_LayoutMode   (ULONG)
                Layout method (applies only to group items).
                Available layout methods are:

                PML_Horizontal - Items are laid out horizontally.
                PML_Vertical   - Items are laid out vertically.
```

```
RETURNS
     Returns a pointer to an item if successful.

SEE ALSO
     PM_MakeMenuA(), PM_MakeIDListA(), PM_ExLstA(), PM_OpenPopupMenuA()
```

1.17 popupmenu.library/PM_MakeMenuA

```
NAME
     PM_MakeMenu -- Create a new menu list.

SYNOPSIS
     menu = PM_MakeMenuA(taglist);
     d0                  a1

     struct PopupMenu *PM_MakeMenuA(struct TagItem *tags);

     menu = PM_MakeMenu(tag1, ...);

     struct PopupMenu *PM_MakeMenu(ULONG, ...);

FUNCTION
     This function is used to link menu items returned by
     PM_MakeItemA().

INPUTS
     taglist - pointer to a taglist listing your menu items.

TAGS
     PM_Item - pointer to a menuitem returned from PM_MakeItemA().

RETURNS
     Returns a pointer to a list of items if successful.

SEE ALSO
     PM_MakeItemA()
```

1.18 popupmenu.library/PM_OpenPopupMenuA

```
NAME
     PM_OpenPopupMenuA -- Open a popup menu.

SYNOPSIS
     userdata = PM_OpenPopupMenuA(prevwnd, taglist);
     d0                           a1       a2

     ULONG PM_OpenPopupMenuA(struct Window *prevwnd, struct TagItem *tags);

     userdata = PM_OpenPopupMenu(prevwnd, tag1, ...);
```

```
        ULONG PM_OpenPopupMenu(struct Window *, ULONG, ...);

    FUNCTION
        This function is used to open a popup menu based on an item list
        created with PM_MakeMenuA()

    INPUTS
        prevwnd - pointer to parent window, used to find out screen, font
                  and other drawing attributes.
        taglist - pointer to a taglist of menu options.

    TAGS
        PM_Menu             (struct PopupMenu *) Pointer to a menu list
                            created by PM_MakeMenuA().

        PM_RecessSelected   OBSOLETE!
        PM_WideSelectBar    OBSOLETE!
        PM_Compact          OBSOLETE!
        PM_SubMenuTimer     OBSOLETE!
        PM_OldLook          OBSOLETE!
        PM_SameHeight       OBSOLETE!
        PM_CheckMark        OBSOLETE!
        PM_ExcludeMark      OBSOLETE!
        PM_SubMenuMark      OBSOLETE!
        PM_SmartRefresh     OBSOLETE!

        PM_Left             (ULONG) Horizontal position of the menu, relative
                            to the menus left edge. (V3)

        PM_Top              (ULONG) Vertical position of the menu, relative
                            to the menus top edge. (V3)

        PM_Code             OBSOLETE!

        PM_PullDown         (BOOL) Turn the menu into a pulldown menu. (V5.1)

        PM_MenuHandler      (struct Hook *) Menu handler function (hook). (V6.0
)

        PM_Right            (ULONG) Horizontal position of the menu, relative
                            to the menus right edge. (V7.3)

        PM_Bottom           (ULONG) Vertical position of the menu, relative
                            to the menus bottom edge. (V7.3)

        PM_CenterScreen     (BOOL) Center the menu on the screen. (V7.3)

        PM_UseLMB           (BOOL) Reverses the function of the mouse buttons.
                            Left button will be used to select items, and right
                            button to select multiple items. This tag is only
                            required when multiselect is used, and the menus is
                            opened with LMB. (V7.3)

        PM_LocaleHook       (struct Hook *)
                            'GetString' hook used to get localized strings.
                            The hook function will recieve a pointer to the
                            menu item (struct PopupMenu *) in A2. A pointer
```

```
                        to the Hook structure will be in A0, and an APTR
                        pointer ("pointer to pointer") will be in A1.
                        At the first location in this array is the string
                        ID stored.
                        A0. (V9)

     PM_ForceFont       (struct TextFont *)
                        Render the menus using this font only. (V9)

     PM_HintBox         (BOOL)
                        Make the menu dissappear when the mouse is moved.
                        Intended for displaying "ToolTips". (Like the
                        Help Bubbles in MUI, only they don't look like
                        bubbles, and you have to handle the opening
                        yourself. (V9)


RETURNS
     Returns the value of UserData of the selected item, if no item
     was selected, NULL is returned.

SEE ALSO
     PM_MakeMenuA()
```

1.19 popupmenu.library/PM_ReloadPrefs

```
NAME
     PM_ReloadPrefs -- Reload preferences. (V9)

SYNOPSIS
     PM_ReloadPrefs();

     void PM_ReloadPrefs(void);

FUNCTION
     This function is used by the preferences program to tell the library
     to reload the settings file. ("ENV:popupmenu.cfg")
```

1.20 popupmenu.library/PM_RemoveMenuItem

```
NAME
     PM_RemoveMenuItem -- Remove a menu item.

SYNOPSIS
     item = PM_RemoveMenuItem(menu, item);
     d0                       a0    a1

     struct PopupMenu *PM_RemoveMenuItem(struct PopupMenu *,
                                         struct PopupMenu *);

FUNCTION
     This function removes an item from a popup menu.
```

```
     The removed item is NOT freed, you MUST free this item with
     PM_FreePopupMenu(), unless you plan to reuse in another menu.
     You may, for example, insert this item in another menu or at
     another position in this menu, using the function
     PM_InsertMenuItemA(). (but then ofcourse, you MUST NOT free the
     item)

INPUTS
     menu - pointer to the holds the item to be removed.
     item - pointer to the item to be removed.

RETURNS
     Returns a pointer to the removed item, or NULL if the item was not
     present in the specified menu.
     You are responsible for freeing this item!

SEE ALSO
     PM_MakeMenuA(), PM_FreePopupMenu(), PM_InsertMenuItemA()
```

1.21 popupmenu.library/PM_SetItemAttrsA

```
NAME
     PM_SetItemAttrsA -- Specify attribute values for an object. (V3)

SYNOPSIS
     result = PM_SetItemAttrsA(item, tags);
     D0                        A2    A1

     ULONG PM_SetItemAttrsA(struct PopupMenu *, struct TagItem *);

     result = PM_SetItemAttrs(item, tag1, ...);

     ULONG PM_SetItemAttrs(struct PopupMenu *, ULONG, ...);

FUNCTION
     Specifies a set of attribute/value pairs with meaning as
     defined in libraries/pm.h.
     item can be directly taken from PM_FindItem as the input is
     checked against NULL pointers.

EXAMPLE

     struct PopupMenu *menu;

     ....
     /* Initialize the menu... */
     ....

     PM_SetItemAttrsA( PM_FindItem( menu, itemid ),
             PM_Checkit,     TRUE,
             PM_Checked,     TRUE,
             TAG_DONE);

INPUTS
     item = pointer to a popup menu item.
```

```
        tags = array of TagItem structures with attribute/value pairs.

    RESULT
        Returns the number of successfully changed attributes.
```