LCD Menu Module

Description

Module for displaying a hierarchical menu system on a character LCD. The module handles scrolling menu items and navigation up and down menu levels. The module triggers an event in the main program when the user selects a menu item which requires an action other than opening another menu. All menu data is held as Const Arrays in a separate module. Supports LCDs from 1 to many lines.

① Menu Title
Can be turned on or off with module option. If the menu title isn't shown then all the LCD lines are used to show the menu items.

② Menu Items
Menu items may open another menu or carry out an action. In the example shown, Stop and Change Direction are 'action' items whilst Motor Settings opens a new menu.

③ Up Scroll Indicator
Shows that there are menu items above those shown. Can be turned on or off with module option.

④ Menu Pointer
Shows currently selected menu item. Character used for pointer can be set with module option.

⑤ Down Scroll Indicator
Shows that there are menu items below those shown. Can be turned on or off with module option.

Module Options

Menu_LCDLines	Integer. Number of lines on LCD. Default value: 4.
Menu_ShowMenuTitle	Boolean. True = Show menu title on first line of LCD. Default: True.
Menu_ShowScrollIndicators	Boolean. True = Show up and down scroll indicators. Default: True. Note: If Menu_ShowScrollIndicators = True then the first 2 custom character slots of the LCD are used to store the up and down indicator symbols.
Menu_ScrollWrapping	Boolean. True = Menu scrolling wraps around from top to bottom and bottom to top. Default value: False.
Menu_PointerCharacter	Byte. Character used as pointer. Default value: 126 (→).

Public Subs

Initialise(*pItemActionEvent* As TEvent)	Should be called at start of main program to initialise module. *pItemActionEvent* = name of event handler in main program which is called when user selects an 'action' menu item.
ShowMenu(*pMenuNumber* As Byte = 0)	Displays the specified menu number. If no menu is specified then the root menu (number 0) is shown.
MoveUp() and **MoveDown**()	Move pointer up or down, scrolling the menu if necessary.
Back()	Move up one level in the menu hierarchy, showing the parent of the current menu.
SelectItem()	Selects a menu item. If the item opens a new menu then the new menu will be displayed. If the item is an 'action' item then the event handler in the main program is called to carry out the action.

Public Variables

SelectedMenuItem	Byte. Holds the index number of the selected menu item.

Menu Data Module

The menu structure, titles, and menu items are all stored as a number of Const Arrays in a separate module called the Menu Data Module. To let the Menu Module know where to find the Menu Data Module, the following line must be placed in the main program before any `Include` statements:

```
#define MenuDataModule = "MyMenuData.bas"
```

Replacing `MyMenuData.bas` with the filename of the Menu Data Module.

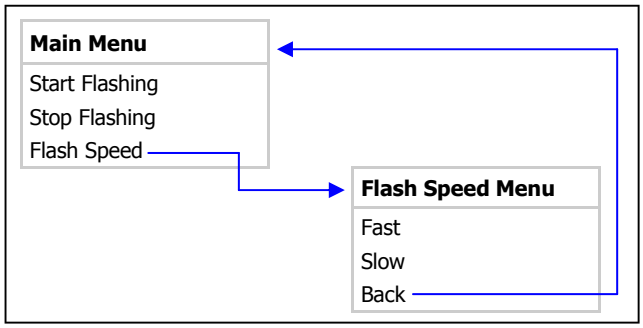
Menu Data Const Arrays

The following arrays are used in the Menu Data Module to store the menu data:

Public Const mnuMenuName	String. Stores the name of each menu.
Public Const mnuMenuItemStart	Byte. Stores the index number of the first menu item for each menu.
Public Const mnuMenuItemEnd	Byte. Stores the index number of the last menu item for each menu.
Public Const mnuParentMenuID	Byte. Stores the index number of the parent menu for each menu.
Public Const mnuItemName	String. Stores the name of each menu item.
Public Const mnuItemAction	Byte. Stores what action each menu item carries out. If the item opens another menu then this array holds the index number of the menu to open. If the item is an action item then this array holds 255.

Example:

The diagram on the right shows a simple menu system consisting of two menus and six menu items.



The **Main Menu** contains two action items (**Start Flashing** and **Stop Flashing**) and the item **Flash Speed** which opens the **Flash Speed Menu**.

The **Flash Speed Menu** contains two action items (**Fast** and **Slow**) and the item **Back** which opens the **Main Menu**.

The Const Array declarations for this menu system are:

```
Public Const mnuMenuName(2) As String = ("Main Menu", "Flash Speed Menu")
Public Const mnuMenuItemStart(2) As Byte = (0, 3)
Public Const mnuMenuItemEnd(2) As Byte = (2, 5)
Public Const mnuParentMenuID(2) As Byte = (0, 0)
Public Const mnuItemName(6) As String = ("Start Flashing", "Stop Flashing",
"Flash Speed", "Fast", "Slow", "Back")
Public Const mnuItemAction(6) As Byte = (255, 255, 1, 255, 255, 0)
```

Menu Builder

Menu Builder is a PC-based utility which allows you to design the menu structure using a simple tree-view control. Menu builder will then generate the Const Array declarations and other text for the Menu Data Module automatically. See separate note, “Menu Builder Utility”.

Main Program Event Handler

The module triggers an event in the main program whenever an 'action' menu item is selected. The event handler can find which menu item has been selected by checking the value of the variable **SelectedMenuItem** which is set to the index number of the selected menu item.

The name of the event handler should be passed to the module at the start of the main program by calling the Initialise sub:

```
Initialise(MenuEvent)
```

Replacing `MenuEvent` with the name of the event handler.

Example:

Using the same menu structure as used in the previous example, the event handler routine may look like this:

```
Event ItemSelected()
    Select Menu.SelectedMenuItem
        Case 0                      'Start Flashing
            LEDFlash = True
        Case 1                      'Stop Flashing
            LEDFlash = False
        Case 3                      'Flash Speed - Fast
            FlashDelay = 100
        Case 4                      'Flash Speed - Slow
            FlashDelay = 200
    EndSelect
End Event
```

To improve readability, the index numbers of the menu items can be aliased as meaningful names. The Menu Builder utility automatically generates the necessary alias declaration statements which should be placed in the Menu Data Module. The aliases generated by Menu Builder take the following format:

mnu*MenuName_ItemName*

The Menu Builder utility also generates Select...Case statements which can be pasted directly into your event handler. The same event handler routine as above but using the aliases and code generated by Menu Builder would look like this:

```
Event ItemSelected()
    Select Menu.SelectedMenuItem
        Case mnuMainMenu_StartFlashing
            LEDFlash = True
        Case mnuMainMenu_StopFlashing
            LEDFlash = False
        Case mnuFlashSpeedMenu_Fast
            FlashDelay = 100
        Case mnuFlashSpeedMenu_Slow
            FlashDelay = 200
    EndSelect
End Event
```

Modification to Swordfish Library Module - LCD.bas

In order for the Menu module to work, it is necessary to add a new sub to the LCD module provided with Swordfish. This new sub allows you to write Const strings to the LCD.

The following code should be placed after the “WriteAt” sub in the module LCD.bas:

```
Public Sub WriteConstStringAt(pY, pX As Byte, ByRefConst pText As String)

    SetLocationY(pY)
    SetLocationX(pX)
    MoveTo()

    EECON1 = 0
    EECON1.7 = 1

    TABLEPTR = @pText

    ASM
        TBLRD*+
    End ASM

    While TABLAT <> 0

        WriteItem(TABLAT)

        ASM
            TBLRD*+
        End ASM

    Wend

End Sub
```

The modified module should then be saved in the UserLibrary folder with the filename “LCD.bas”.