Butt(n)Meister And About Us

Purpose:
Butt(n)Meister allows you to easily Create & Maintain efficient rectangular (Vertical/Horizontal) Button-Bars and Floating Tool Pallettes for your VB Applications; or applications derived from other Windows development Tools/Compilers. The samples projects are limited to VB because that is what I use most for my application UIs.

Your Button-Bars ***will not require a VBX, but just a few lines of VB Code*** (See the Example Projects included!). I hope you find Butt(n)Meister a useful addition to Your VB or Windows development Toolbox. I've also included information on some of our other finely crafted products. If you wish to register or need further information please send a CompuServe Email to Brad Perkins CIS 76670,1030 From Internet 76670.1030@compuserve.com or: write or call:

Brad Perkins
Intelligence Mfg. Company
3136 Redwood Ave
West Sacramento, CA 95691
(916) 372-6680 (24 Hr. Message)

About Butt(n)Meister

Button-Meister allows you to arrange and compile multiple bitmaps into a single bitmap. This is convenient for those of us who like floating button bars. The PICCLIP custom control included with VB uses a single bitmap composed of many button images. It Blits or copies each button image out of the source bitmap and pastes it into a target display control. It is much more memory and resource efficient to have these individual button images compiled into a single bitmap than to store each button individually. With Butt(n)Meister, you drag-drop your individual button images onto a spreadsheet arranging them as desired and then press the compile button to magically turn your arrangement into a single bitmap file. It can then be used with VBs PICCLIP custom control. Or even better yet, you can use a single picture control to display your complete button bar!!. With a few lines of magical VB code (Included!), you will have fully functional button bar ***without any VBX required***! The sample VB code essentially replaces the PICCLIP control and allows a much more elegant and efficient solution in implementing floating toolbars in VB. Butt(n)Meister can be used for making standard type toolbars also. Butt(n)Meister also contains a clipping function that allows you to trim unwanted pixels off the edges of your buttons.

I originally developed Button-Meister for our own internal VB projects. I am a firm believer in Button-Bars since I have difficulty remembering Alt-Key-Key-Key sequences. The PICCLIP VBX that comes with VB allows BitBlits from a single bitmap into picture controls for each button. This requires a picture control for every button you have and therefore is not an optimal solution. It requires extra code to align and place the controls (for the various display device contexts) and is really difficult to use with controls like VideoSofts VS Elastic (TM). To summarize my problems in using the PICCLIP custom control:

1. It requires a picture or image control for every target button.
2. You must ship the PICCLIP VBX with all your projects.
3. Manually Creating the bitmap that contains all your buttons is a real pain in the you-know-what.
4. Maintaining the main bitmap is just as big a pain.
5. Alignment & Placement requires unnecessary code.
6. The result may be difficult to use with certain other controls.

#1 has always been a gripe for me because of the effort and code required to place and align all these individual picture boxes. Since I religiously use VideoSofts wonderful VSVBX Elastic Control on most of my forms, the multiple picture-control button bar (PICCLIP) will not work at all in many situations.

Actually #2 is not much of a problem since I am also a firm believer in not re-inventing the wheel, so I make extensive use of third party custom controls in all my projects. However Using the few lines of VB code to give you a fully functional ToolBar is more efficient in my mind than using another VBX. You also have total control over the functionality of the finished product.

#3 & #4 are directly addressed by Butt(n)Meister. It stores your buttons paths in an Access database so your button project can be retrieved and modified for maintenance purposes.

#5 & #6 can be a real pain. Individual buttons must be arranged and aligned using code. Other undesirable effects occur when the parent control has properties for sizing and placing its child controls. Butt(n)Meister compiles your buttons into a single

(picture) control eliminating these problems.

Installation

Steps

0. Unzip the Zip file in a clean directory of your choice.
1. Move the VSVBX.LIC file to the Windows directory.
2. Place the .VBX and any .DLL files in the Windows/System directory
3. Place the .HLP file(s) in the Butt(n)Meister Directory
4. Place the BUTTON.INI file into the windows directory.
5. Call or leave an Email when you need help or find a bug.
6. Please Register Butt(n)Meister so I can continue to make useful products for you.

Thanks Ahead for your Support!

Detailed Instructions

If you are installing Butt(n)Meister from the Shareware 'Zip' file, Expand the Zip file into the desired directory created for it. You must already have VBRUN30.DLL in your ..\Windows\System directory. You must also move all .VBX and .DLL files to the ..\Windows\System directory. If some of these VBXs and/or DLLs already exist in your ..\Windows\System directory you may opt to NOT move these files because you may have newer versions. Move the BUTTON.INI to the windows subdirectory. It contains the path to the BUTTON.MDB (Database) file. Move the BUTTON.MDB file to any directory you choose. If there are .LIC files these usually need to be placed in the Windows directory. To create an Icon for Butt(n)Meister, open the Windows File Manager and drag BUTTON.EXE into the desired program group. If you register Butt(n)Meister you will receive a standard Windows Setup Diskette. Run the SETUP.EXE file on the distribution disk and follow the on-screen instructions. Read any and all README files included with Butt(n)Meister.

When Butt(n)Meister is first started it attempts to find the BUTTON.MDB file from the path information in the BUTTON.INI file. If it is not found a dialog box will appear where you can select the correct path to it. Once the path is set you wont be asked for it again (unless it becomes invalid).

Versions 1.1 & 1.2 Fix minor bugs and Adds requested features.

IMPORTANT!!! If you already have Butt(n)Meister V1 you will need to replace the BUTTON.MDB file because format changes have been made in V1.1 and V1.2 Import your existing Button definitions using MS Access or VB.

Added features

1. BUTTON.MDB can be moved to any directory. Place the BUTTON.INI file into the windows directory. When Butt(n)Meister is first started it will ask for the path to the database file. Once established, Butt(n)Meister will not ask for the path again until the entry in the BUTTON.INI file becomes invalid.

2. Balloons can be disabled from the preferences form by unchecking the checkbox

Registration

If you wish to register Butt(n)Meister you will receive the Latest and Greatest version and a Free Fader custom control that can be used like a volume control. You will also receive regular news about what great productivity products we will deliver next. The registration fee is $35.00. Send a Check Made out to:

Brad Perkins

Intelligence Mfg. Company
3136 Redwood Ave
West Sacramento, CA 95691
(916) 372-6680

Or You can register directly on CompuServe: GO SWREG and Register Product number #4092
I will Email The latest version of Butt(n)Meister and your free Fader Custom Control.

Other Products-in-work

We wish to inform you of works-in-progress because we will be able to evaluate the level of your interest in these upcoming products to determine where we should be concentrating our resources. So please send us your feedback and comments regarding these future releases. The ones that generate the most interest will receive our immediate attention to deliver first and in the shortest time possible. Following is a brief description of these works. If you are interested in any number of these products and want more info, please call us to receive a more detailed description.

Image Scanner Support (TWAIN Custom Control)
Optical Character Recognition (OCR Custom Control)
Document/Forms Processing (Parser DLL for VB)
English Language Queries
An Expert System Custom Control

Scanner Support

1. A TWAIN VB Custom control that will drive TWAIN compliant Scanner devices. We are using this control to acquire images of MSDS (Material Safety Data Sheets) to feed our OCR custom control (see Below) from various TWAIN compliant scanners.

Optical Character Recognition (OCR)

An OCR VB custom control that converts a scanned or faxed image file into a text document that can be edited like any other ASCII text file. This control will literally allow you to create your own advanced document processing applications. This control was developed using Xeroxs TextBridge ICR Server engine API. It supplies access to the dozens of properties including languages, thumbnail images, popup image fragments for text corrections, etc., etc., etc... We are using this control with our advanced document Parsing custom control described in this document. This control will require a licensed copy of the Xerox TextBridge OCR product. At $68.00 from some mail order houses, TextBridge is an incredible value. I may also become a TextBridge reseller and get a better value for my customers. FYI, We have found TextBridge to be Slightly slower than OmniPage-Pro, but significantly more accurate on our MSDS documents.

Document/Forms Processing

A general purpose Document Parsing Engine VB Custom Control. This product is already being used to Convert paper (See our OCR custom Control Above) MSDSs (Material Safety Data Sheets) into data elements for insertion into a local Access Database for transfer to an Oracle database residing on a mainframe. This control uses advanced Parsing technologies to find the 16 MSDS Sections (Tables) and then the Descrete Data elements (fields) within those Sections. It uses a dictionary metaphor whereas many free-form synonyms may be defined for the myriad sections and fields. Maintenance of the dictionary is accomplished using the included VB application that talks to an access database. All VB source code for the Dictionary administrator is included and may be modified and integrated with your own products. Anyone looking to convert Paper forms and even non-standard documents like MSDSs into data will love this VB Control/ Dictionary combination. We will also include a sample VB application showing how to use this control with various documents and VB editor controls.

English Language Database Queries

EQL is our English Language Query VB Custom control. It Literally translates an English Language Query into SQL that can be applied to Any ODBC database. This control will produce SQL for Microsofts new ODBC specification that allows a single SQL dialect across all supported back-ends. It uses a Dictionary of User defined synonyms for tables, fields, and database values. This control will include a VB Application to Create and maintain the Administration Templates required for your database applications. The Administrator contains information about your database Tables, Fields, Synonyms, Joins, Constraints etc. EQL Will also include a general purpose Query tool for testing your templates and demonstrating integration details (VB Source Included).

Adding Expertise to VB Apps

IBIS-Plus is our high performance Expert System Server VB Custom Control that allows you to add Artificial Intelligence to your applications. It contains a Forward & Backward chaining inference engine that uses English-Like production rules. Learning IBIS-Plus Syntax can be mastered in a matter of a few hours (sometimes less than an hour). Additionally IBIS-Plus has a very robust, integrated Prolog Language interpreter built in! Its like having 3 programming languages in the same environment. A procedural language like VB combined with the Declarative nature of IBIS-Plus will open a whole new universe of possibilities for your VB Projects! IBIS-Plus includes the VB Custom Control and a complete Development Environment written in VB with all the *Source Code! Several sample applications are included with all the VB *Source Code. Registered users of IBIS-Plus can distribute their applications Royalty-Free!

* VB Source Code Only and does not include the Source for IBIS-Plus Custom Control and Server Engine. Does not include licenses for third party custom controls used in Development environment and Sample applications.

Butt(n)Meister Problems

Not all is roses with Butt(n)Meister, for one thing its sort of a resource hog. Why you ask? Because it loads every bitmap or Icon in the current directory into the top-line list box. So there is a danger of exhausting resource memory. I have on only a few occasions run into this problem so it shouldnt be a show stopper. I have strategically placed code to check for a low resource condition. When triggered you will be given the opportunity to save your work before a fatality. To help alleviate this problem, place the bitmaps that you intend to use in your button bar in a separate directory thereby avoiding billions of bitmaps being loaded into the Top-Line list box. Other tips for this problem include frequently saving the current button database to prevent catastrophic loss.

Butt(n)Meister stores the path of each Button for the bitmap in the local database. If any buttons are deleted or moved from windows or DOS, this path information will no longer be valid and the next time Butt(n)Meister attempts to load the bitmap definition it will generate an error. Now that Ive told you, this should also not be a show stopper. Just keep the directories and files consistent and persistent as long as you need them. Right now there is no mechanism built into the program to resolve invalid paths except to manually change the path info from Access. Of course I could add extensive database management functions if such were requested by you.

Butt(n)Meister Futures

I think most would agree that the next major upgrade to Butt(n)Meister should include a code generator that will create the VB code to drive your particular button bar whether it is a vertical, horizontal or rectangular (square?) ToolBar. This I could easily add this feature but will require a certain number of registrations and interest from you. If I don't hear from you I must assume my product is not useful. Other features could be added depending on the demand.

Plans for version 2.0 include a comprehensive Button Factory to create very fancy buttons of different shapes and sizes. Other features will include:

-Multiple bitmaps with transparency options
-Fancy Angled text on created buttons
-Colors galore
-Border styles
-A reporting facility for documenting bitmap collections and projects

Butt(n)Meister User Guide

The Following topics are ordered by general operation sequence; meaning the order in which most button bars will be produced. This assumes that all of your separate button bitmap files already exist in *.BMP and/or *.ICO format. There are a few rules that are enforced when creating your Button Bars:

General Concepts

The Sequence of events in Butt(n)Meister are as follows:

1. Create Some buttons using your favorite Bitmap/Icon Editor and place them in a single directory. Of Course you can use the Buttons Included with VB or from other sources too.

2. Run Butt(n)Meister, Find the Directory using the main Window Directory/File lists. Butt(n)Meister will automatically load the BitMaps into the Palette Menu. Select your preferred File Type.

3. Drag Buttons From the Palette Menu to the Spreadsheet and arrange them as desired.

4. Save the arrangement into the database (optional) by typing a description of your project into the combobox editor.

5. Compile the Arrangement into a single Bitmap File.

6. Modify one of the sample Projects so it works with your new Toolbar.

7. Copy the code into your product along with the two Picture controls.

8. Use VB Code to use your new Toolbar (Case Statements... etc.).

Butt(n)Meister Rules

All bitmap images must be of the exact same X/Y physical dimensions. In other words, buttons within your button-bar must all be the same size. Butt(n)Meister will raise an error message stating this fact when you attempt to insert an odd sized button. The legal button size is established when you drag the first button onto the spreadsheet. This can only be reset by clearing the spreadsheet and starting over.

Butt(n)Meister Stores all Button-Bar definitions by <Path>"FileName". This can be a drawback; especially when project bitmap files or directories are moved, deleted or renamed. Any of these actions can cause the database entry(s) to become invalid when attempting to load a previous Button-Bar project. Butt(n)Meister doesn't yet have a facility for handling this potential problem.

Select The File Type

Select the type of Image file (.ICO or .BMP) using the radio buttons.

You may mix .ICO and .BMP images on the spreadsheet. The final compiled result is always saved as a .BMP file.

Find The Drive/Directory

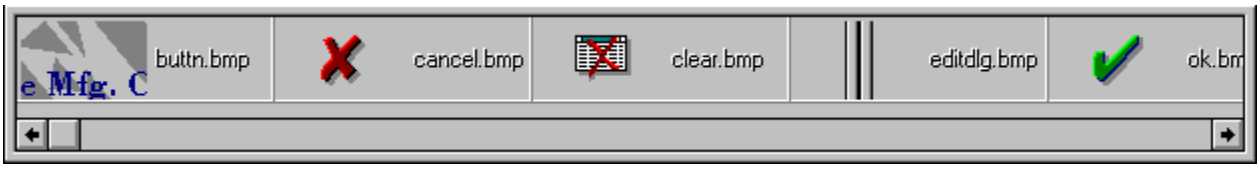
Select the Drive...

... & Directory where your button bitmaps and/or Icons reside.

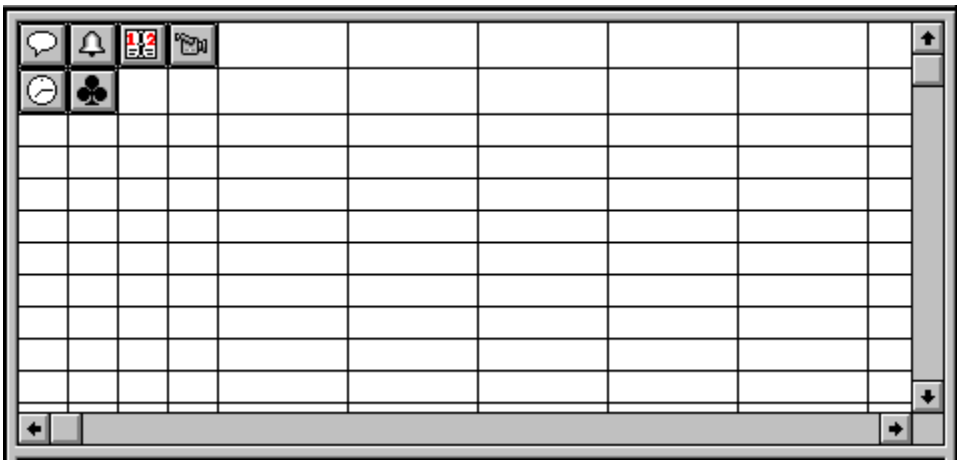
Butt(n)Meister will load ALL bitmap (.BMP) and Icon (.ICO) files from the selected directory into the topline button pallette.

Create The Button Arrangement

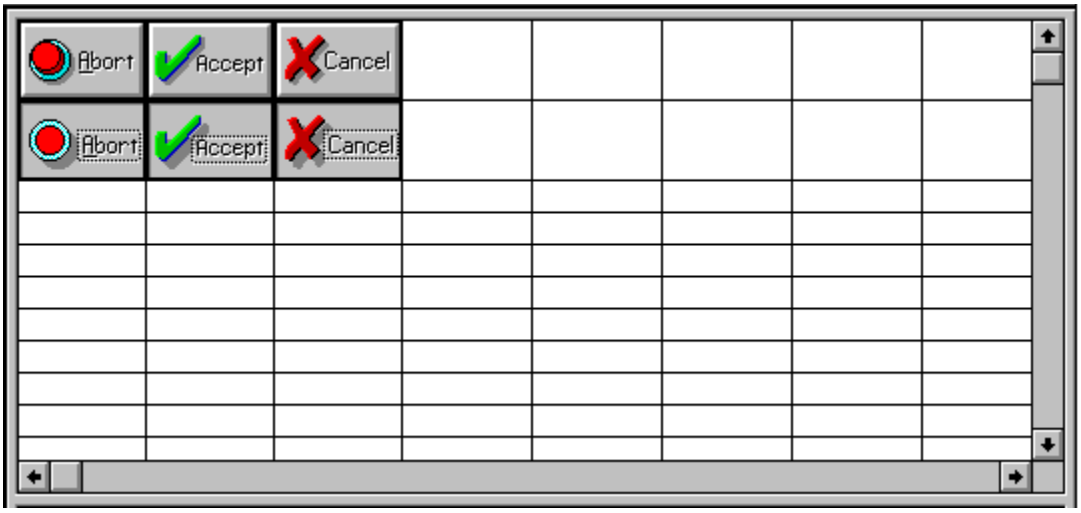
Drag/Drop the desired Up-Position bitmaps from the top-line bitmap Palette...



...to the appropriate spreadsheet location.



Place the Up-Position buttons first in the desired arrangement and order. The Up Position Buttons are the ones you will see on your finished product. Duplicate the Arrangement with the Down-Position bitmaps directly below or to the right of the Up-Position array.



The Down Position Buttons will normally be hidden in a separate ***Picture Control*** when your product is running. Your program will BitBlit the Up Position buttons into a visible ***Picture control*** from the other hidden ***Picture control***. When the left button is pressed over the bitmap; calculations are made to determine which portion of the Hidden Bitmap must be Blitd over the Up Position button.

Re-Arranging And Deleting Buttons

Buttons may also be dragged from location to location on the spreadsheet. You may delete buttons from the spreadsheet by dragging them to the ***Delete*** button on the button bar.

The Delete Button also deletes Button Bar Definitions from the database based on the name in the Combo-Box. Note: The ***Delete*** Button Does ***Not Delete Files***, So your bitmap files are safe.

Testing Your Button Bar

There are two Sample VB Projects that will help you get started using your new toolbars. First the Win API function BitBlt must be declared. Constants Representing your toolbar and Button Dimensions must also be determined by using Butt(n)Meisters Status Bar...

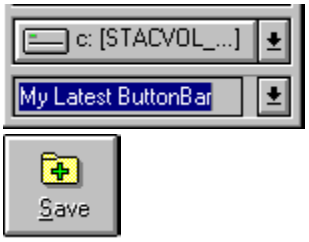
...when you have finished compiling the final toolbar. *Button Dim, X:24 Y:22* gives the Width and Height respectively of each Button in Pixels. *Bit Dim, X:4 Y:2* gives the Width And Height Respectively of the Toolbar in Buttons (Here it is 4 Buttons Wide By 2 Buttons High). Count give the total number of buttons in the bitmap. Note: You can omit buttons and leave **Holes** in the compiled result but these holes are included in the Button Count by Butt(n)Meister. Ideally Butt(n)Meister would do all the calculations and generate the VB Code to run your toolbar. If User support is great enough and there is a genuine demand, then I will develop that function for the product.

```
Sub ButtonBar_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' paste the down button image over the Up button currently displayed
    Dim Success%
    ' Calculate the position of the Down Button from the source bitmap that
must be copied here
    ColX = X \ ButtonWidth: RowY = Y \ ButtonHeight
    Success = BitBlt(ButtonBar.hDC, ColX * ButtonWidth, RowY * ButtonHeight,
ButtonWidth, ButtonHeight,                    Picture1.hDC, ButtonWidth * ColX,
ButtonHeight * (RowY + BarHeight), &HCC0020)
    ButtonBar.Picture = ButtonBar.Image
    ' Make sure it's seen
    ButtonBar.Refresh
End Sub

Sub ButtonBar_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single)
    ' Restore the Up button image
    Dim Success%
    Success = BitBlt(ButtonBar.hDC, ColX * ButtonWidth, RowY * ButtonHeight,
ButtonWidth, ButtonHeight,                    Picture1.hDC, ButtonWidth * ColX,
ButtonHeight * RowY, &HCC0020)
    ButtonBar.Picture = ButtonBar.Image
End Sub

Sub Form_Load ()
    Dim Success%
    ButtonBar.Left = 0
    ButtonBar.Top = 0
    '   Size the target button bar
    ButtonBar.Width = (ButtonWidth * Screen.TwipsPerPixelX) * BarWidth
    ButtonBar.Height = (ButtonHeight * Screen.TwipsPerPixelY) * BarHeight
    '   Then Size the form to the ButtonBar
    Me.Height = (Me.Height - Me.ScaleHeight) + ButtonBar.ScaleHeight
    Me.Width = ButtonBar.ScaleWidth + Screen.TwipsPerPixelX
    '   Move the source bitmap out of the way
    Picture1.Left = -10000
    Picture1.Top = ButtonBar.Height
```

```
    ScaleMode = PIXELS
    Picture1.ScaleMode = PIXELS
    ButtonBar.ScaleMode = PIXELS
    '   Copy the main Bitmap (Buttons) to the destination minus the Down-
Buttons
    Success = BitBlt(ButtonBar.hDC, 0, 0, ButtonBar.Width, ButtonBar.Height,
Picture1.hDC, 0, 0,&HCC0020)
    '   Initialize the Picture Property
    ButtonBar.Picture = ButtonBar.Image
End Sub
```

Saving Your ToolBar Description

Save the result to the database by typing a description into the Combo-Box editor and pressing the Save button...

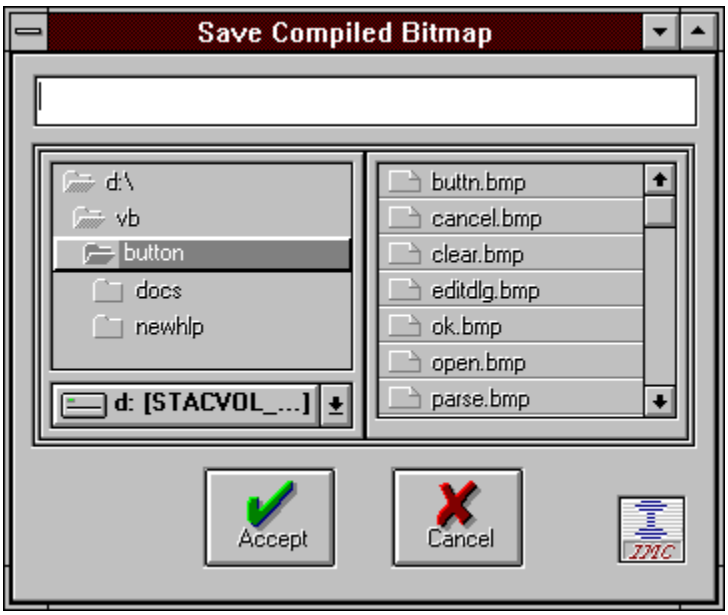


...or select an existing database description from the combo-box and then press Save.

Compile Your ToolBar

Compile the result into a single bitmap by pressing the Compile Button.

This brings up The Save Dialog Box. Type a valid DOS Filename to save the compiled result bitmap.



This is the file you will use in your project. Using the sample code as a template create the final test application and test your new button-bar! If you need help please leave a CIS (76670,1030) Email Message to me or call. Ill be glad to help.