PARAMS -- A Visual Basic Example

Introduction:

'PARAMS' is a Visual Basic code example that demonstrates a method for passing parameters between Visual Basic forms when loading and unloading forms. It also shows a method for setting application wide variables. Neither of these methods use Global variables. The example code also includes some dialog management routines and an updated version of my IsFile() function.

This code example is Freeware. There is no charge or royalties for using it in your programs. Please see the Disclaimer & Legal Stuff section of this document for further information concerning distribution.

Installation and Program Requirements:

'PARAMS' is written in Visual Basic 3.0. The PARAMS.BAS module will run with either Visual Basic standard or pro editions and does not depend upon any external routines or controls.

The demo program requires Visual Basic professional edition. No custom controls are required.

The best way to prepare the PARAMS.BAS module for use in your programs is to copy into your VB directory or place it in a directory where you normally store your common VB routines.

Concept used in 'PARAMS':

The concept used in 'PARAMS' was derived from the book *Code Complete* by Steve McConnell. (Microsoft Press 1993, IBSN 1-55615-484-4). The portions of the book that relate to this example are section 10.6 'Global Variables' and the related sub-heading, 'Using Access Routines Instead of Global Data', section 6.2 'Information Hiding', in particular, the sub-heading entitled 'Module data mistaken for global data', section 11.8 'Arrays', and section 11.7 'Named Constants'.

How To Use 'PARAMS':

The heart of 'PARAMS' is the PARAMS.BAS module. This module contains get, set, and clear routines for both form as well as get and set application level parameters.

Setup

The first section of the module to consider is General|Declarations section. In this section, related constants and variables are set up. The constants, P_MAX_FORM and P_MAX_APP, are set to the actual number of parameters you will be using in the program or, if you don't know this value for sure, a value large enough to handle any contingency. This is the only part of PARAMS.BAS that might need to be modified from program to program. However, if you wish, they could be moved to Global Constants that could be modified from program to program without requiring changes to PARAMS.BAS.

In the example program, this section looks like this:

```
Const P_MAX_FORM = 4
Const P_MAX_APP = 1
```

The second part of General|Declarations are the module level variables. The variables hold the values of the parameters that are passed. These parameter arrays can either be module level arrays or, if desired, they could be moved to Static arrays in a Private function in this module to further limit their scope. This option would make the most sense if you planned on placing the code in PARAMS.BAS into a module with other, unrelated, code if you were encountering the limitation on the number of files allowed in a Visual Basic project

In the example program, this section looks like this:

```
'
' Form Level Parameters
'
  Dim msFormParams(P_MAX_FORM) As String
'
' Application Level Parameters
'
  Dim msAppParams(P_MAX_APP) As String
```

To make it easier to use the parameter routines in your program, it is recommend that you use mnemonic global constants to refer to the array elements, rather than numbers. Using this method makes your program less error prone and easier to read. However, this is not a requirement of the routines.

In the demo program, the following Global Constants related to the parameter passing routines are declared.

```
'
' Application Wide Parameter Constants
'
  Global Const PRA_DATABASE = 1    ' Database name
'
' Form Level Parameter Constants
'
  Global Const PRF_TABLE = 1       ' Database Table
  Global Const PRF_CRITERIA = 2    ' Search Criteria
  Global Const PRF_INDEX = 3       ' Position Index
  Global Const PRF_RESULT = 4      ' Search Results
```

SetFormParam

```
Sub SetFormParam (sParam As String, iElement As Integer)
```

This subroutine is sent a string value and an element number to set in the parameter array. If the element number is out of range, the value won't be modified and a programmer warning will appear on the Visual Basic debug dialog box.

Here is an example from the demo program of a call to SetFormParam:

```
SetFormParam CStr(txtCriteria(Index).Text), PRF_CRITERIA
```

This particular call sets the value to be passed into the SQL WHERE clause when the search form is loaded.

GetFormParam

```
Function GetFormParam (iElement As Integer) As String
```

This function returns the value assigned to a specified element of the parameter array. Once again, if the specified element is out of range, a debug window programmer notification will be sent. Also, the value returned by the function will be an empty string.

Here is an example from the demo program of a call to GetFormParam:

```
sCriteria = GetFormParam(PRF_CRITERIA)
```

This particular call retrieves the information that was stored in the SetFormParam mentioned in the previous section.

ClearFormParam

```
Sub ClearFormParam ()
```

This subroutine clears the entire form parameters array so that it's prepared for use with another form. Individual items may be cleared by sending them an empty string.

In the demo program, this sub is called after the main form has retrieved the data sent to it by the search form.

SetAppParam

```
Sub SetAppParam (sParam As String, iElement As Integer)
```

This routine is used to hold data that will be shared throughout the program. In practice, you will want to have separate Get/Set routines for each category of data. For example, if your program had screen position information and database information, it would be best to have separate Get/SetDatabaseParam and Get/SetScreenParam routine sets, rather than having them mixed together. (See the reference to this in section 10.6, page 231 of *Code Complete*)

In the demo program, this sub is used to store the location and name of the BIBLIO.MDB database in Sub Main().

```
SetAppParam sFileName, PRA_DATABASE
```

GetAppParam

```
Function GetAppParam (iElement As Integer) As String
```

This function is used to retrieve data from the application parameter array. In the example program, it is used to get the location and name of the BIBLIO.MDB database in several places in the program.

```
Set dbBiblio = OpenDatabase(GetAppParam(PRA_DATABASE), True, True)
```

Other Routines included in the 'PARAMS' demo program:

The 'PARAMS' demo program contains the following routines that you might find useful as well.

IsFile

```
Function IsFile (sFileName As String, Method As Integer) As Integer
```

This function is an enhancement to the previously released IsFile routine (ISFILE.ZIP). The enhancements include the addition of error messages to indicate why the file wasn't found when network or file sharing situations occur.

Also, this version of the routine has two different methods of obtaining this information. You can either use the OpenFile API call, that reports more detailed information concerning why a file couldn't be opened. Or, you can use the Dir$ method, where the DOS directory list is used to determine the existence of a file. The API method is particularly useful in checking to see if a database file can be opened. The Dir$ method is better when you only want to check file existence, not file opening capability.

SetDialogMenu

```
Sub SetDialogMenu (frm As Form)
```

This routine makes a standard VB form's control menu take on a look more similar to the one found in standard Windows dialogs. Control menus on fixed double bordered pop-up dialogs in most Windows programs only contain a Move and Close entry with no separator bars. In VB, this kind of dialog will have the Switch Too... entry along with a separator bar. This routine eliminates this from the form, so that it takes on a more standard look.

Some property setup is required at design time to setup for the function. The MaxButton and MinButton properties must be set to False for the target form. Also, the fixed double border style must be selected.

CenterForm

```
Sub CenterForm (frm As Form)
```

Using the Move method, this routine will center a form on the screen.

PlaceDialog

```
Sub PlaceDialog (frmSource As Form, frmDialog As Form, iPos As Integer)
```

This routine places a target form in relation to a source form. The routine currently has two options, placing the dialog form slightly down and to the right of the top of the source form, or placing it centered, relative to the source form. A possible addition to this routine would be to place the dialog in relation to a specified location on the source form.

Disclaimer & Legal Stuff:

'PARAMS' is FreeWare. You can use it and modify it as you wish for your own personal use or for inclusion in your own programs. You may upload it to free BBS systems or give it away as long as the entire UNMODIFIED text files and source code are distributed as a whole.

For permission and information about including and distributing 'PARAMS' in a disk/CD collection of shareware/freeware programs or on a pay-for-use BBS system, please contact J. Frank Carr at the address given in the Comments & Suggestions section of this document.

J. Frank Carr makes no warranty of any kind with regard to this software or its documentation, including, but not limited to, any implied warranties of merchantability and fitness for a particular purpose. J. Frank Carr shall not be liable for errors or for incidental or consequential damages in connection with the furnishing, performance, or use of this product. The software and the information contained in the documentation are subject to change without notice.

Comments & Suggestions:

I would like to hear from anyone who gets any use out of this program, or who encounters any problems or bugs with it, or if there are changes or enhancements you would like to see. Should I get enough feedback on this program, I'll probably make enhancements to it.

I am also available for programming projects, training, and consulting in Visual Basic. Contact me as listed below for information regarding my availability and prices.

You may contact me at:

J. Frank Carr
685 Kenneth Lane
Norcross, GA 30093 USA

Voice Mail: (404) 880-5762

CIS: 75120, 2420

America OnLine: JFCarr