Visual Basic Variable Naming and Coding Guidelines

Naming Variables

Data Types

Use the following three letter prefixes to indicate a variables *data type*:

Data Type	Prefix	Example
Boolean	bln	blnFound
Currency	cur	curRevenue
Date (time)	dat	datStart
Double	dbl	dblTolerance
Error	err	errOrderNum
Integer	int	intQuantity
Long	lng	lngDistance
Object	obj	objCurrent
Single	sng	sngAverage
string	str	strFName
User-defined type	udt	udtEmployee
Variant	vnt	vntCheckSum

Control Types

2 Part # Part Head

Use the following three letter prefixes to indicate a controls *type*:

Control Type **Prefix** **Example**
Animation button ani aniMailBox
Checkbox chk chkReadOnly
Combo box, Drop down list box cbo cboEnglish
Common dialog control dlg dlgFileOpen
Communications com comFax
Control (Used within procedures when the specific type is unknown) ctr ctrCurrent
Data control dat datBiblio
Directory list box dir dirSource
Drive list box drv drvTarget
File list box fil filSource
Form frm frmEntry
Frame fra fraLanguage
Gauge gau gauStatus
Graph gra graRevenue
Grid grd grdPrices
Horizontal scroll bar hsb hsbVolume

3 Part # Part Head

Image
    img
    imgIcon
Key state
    key
    keyCaps
Label
    lbl
    lblHelpMessage
Line
    lin
    linVertical
List box
    lst
    lstPolicyCodes
MAPI message
    mpm
    mpmSentMessage
MAPI session
    mps
    mpsSession
MCI
    mci
    mciVideo
MDI child form
    mdi
    mdiNote
Menu
    mnu
    mnuFileOpen
OLE control
    ole
    oleWorksheet
Outline control
    out
    outOrgChart
Pen Bedit
    bed
    bedFirstName
Pen Hedit
    hed
    hedSignature
Pen Ink
    ink
    inkMap
Picture
    pic
    picVGA
Picture clip
    clp
    clpToolbar
Report control
    rpt
    rptQtr1Earnings

Shape controls
shp
shpCircle
Spin control
spn
spnPages
Text Box
txt
txtLastName
Timer
tmr
tmrAlarm
Vertical scroll bar
vsb
vsbRate

Database Objects

Use the following three letter prefixes to indicate database objects:

Data Object
Prefix
Example

Database
db
dbAccounts
Dynaset object
ds
dsSalesByRegion
Field Object
fd
fdAddress
Index object
ix
ixAge
QueryDef object
qd
qdSalesByRegion
Query*
Qry (suffix)
SalesByRegionQry
Snapshot object
ss
ssForecast
Table object
tb
tbCustomer
TableDef object
td
tdCustomers

* Using a suffix for queries allows each query to be sorted with its associated table in Access dialogs (Add Table, List Tables Snapshot).

When writing online Help examples for data access topics, you may want to use the above prefixes as variable names. Here are some examples:

```
Dim DB As Database, DS As Dynaset
Dim TB As Table, strSQLStmt As String
Const DB_READONLY = 4      ' Set constant.
Set DB = OpenDatabase("BIBLIO.MDB")     ' Open database.
' Set text for the SQL statement.
strSQLStmt = "SELECT * FROM Publishers WHERE State = 'NY'"
' Create the new Dynaset.
Set DS = DB.CreateDynaset(strSQLStmt, DB_READONLY)
```

Menu Naming Conventions

Applications frequently use an abundance of menu controls; thus necessitating a different set of naming conventions for these controls. Menu control prefixes should be extended beyond the initial *mnu* label by adding an additional prefix for each level of nesting, with the final menu caption at the end of the name string. For example:

Menu Caption Sequence
Menu Handler Name

Help Contents
mnuHelpContents
File Open
mnuFileOpen
Format Character
mnuFormatCharacter
File Send Fax
mnuFileSendFax
File Send Email
mnuFileSendEmail

When this convention is used, all members of a particular menu group are listed next to each other in the object drop-down list boxes (in the code window and property window). In addition, the menu control names clearly document the menu items to which they are attached.

Coding Guidelines

Setting Environment Options

Use Option Explicit (Require Variable Declaration)

Declaring all variables saves programming time by reducing the number of bugs caused by typos (for example, aUserNameTmp vs. sUserNameTmp vs. sUserNameTemp). In the Environment Options dialog, set Require Variable Declaration to Yes. The Option Explicit statement requires you to declare all the variables in your Visual Basic program.

Save Files as ASCII Text (Visual Basic Only)

Saving form (.FRM) and module (.BAS) files as ASCII text facilitates the use of version control systems and minimizes the damage that can be caused by disk corruption. In addition, you can:

Use your own editor
Use automated tools, such as grep
Create code generation or CASE tools for Visual Basic

Perform external analysis of your Visual Basic code

To have Visual Basic always save files as ASCII text, from the Environment Options dialog, set the Default Save As Format option to Text.

Commenting Your Code

All procedures and functions should begin with a brief comment describing the functional characteristics of the routine (what it does). This description should not describe the implementation details (how it does it) because these often change over time, resulting in unnecessary comment maintenance work, or worse yet - erroneous comments. The code itself and any necessary in-line or local comments will describe the implementation.

Parameters passed to a routine should be described when their functions are not obvious and when the routine expects the parameters to be in a specific range. Function return values and global variables that are changed by the routine (especially through reference parameters) must also be described at the beginning of each routine.

Routine header comment blocks should look like this (see the next section "Formatting Your Code" for an example):

Procedure Header Comment Blocks

Section
Comment Description

Purpose
What the routine does (not how).

Inputs
Each non-obvious parameter on a separate line with in-line comments.

Assumes
List of each non-obvious external variable, control, open file, etc.

Returns
Explanation of value returned for functions.

Effects
List of each effected external variable, control, file, etc. and the affect it has (only if this is not obvious)

Every non-trivial variable declaration should include an in-line comment describing the use of the variable being declared.

Variables, controls, and routines should be named clearly enough that in-line commenting is only needed for complex or non-intuitive implementation details.

An overview description of the application, enumerating primary data objects, routines, algorithms, dialogs, database and file system dependencies, etc. should be included at the start of the .BAS module that contains the project's Visual Basic generic constant declarations.

Note

The Project window inherently describes the list of files in a project, so this overview section only needs to provide information on the most important files and modules, or the files the Project window doesn't list, such as initialization (.INI) or database files.

Formatting Your Code

Because many programmers still use VGA displays, screen real estate must be conserved as much as possible, while still allowing code formatting to reflect logic structure and nesting.

Standard, tab-based, block nesting indentations should 4 spaces (the default).
The functional overview comment of a routine should be indented one space. The highest level statements that follow the overview comment should be indented one tab, with each nested block indented an additional tab.
For example:

```
'*********************************************************************************************
'Purpose:  Locates first occurance of a specified user in the UserList array.
'Inputs:     strUserList():  the list of users to be searched
'              strTargetUser: the name of the user to search for
'Returns:   The index of the first occurance of the rsTargetUser in the rasUserList array.
'              If the target user is not found, return -1.
'*********************************************************************************************
Function intFindUser (strUserList() As String, strTargetUser as String) As Integer
  Dim i As Integer              ' Loop counter.
  Dim blnFound As Integer               ' Target found flag.
  intFindUser = -1
  i = 0
  While i <= Ubound(strUserList) and Not blnFound
        If strUserList(i) = strTargetUser Then
              blnFound = True
              intFindUser = i
        End If
  Wend
End Function
```

Constants

Variables and non-generic constants should be grouped by function rather than by being split off into isolated areas or special files. Visual Basic generic constants such as HOURGLASS should be grouped in a single module to keep them separate from application-specific declarations.

Operators

Always use & when concatenating strings and + when working with numerical values. Using + to concatenate may cause problems when operating on two variants. For example:

```
vntVar1 = "10.01"
vntVar2 = 11
vntResult = vntVar1 + vntVar2          ' vntResult =  21.01
vntResult = vntVar1 & vntVar2          ' vntResult = 10.0111
```

Miscelaneous

Creating Strings for MsgBox, InputBox, and SQL Queries

When creating a long string, use multiple lines of code so the string is easily readable by the programmer. This technique is particularly useful when displaying a MsgBox, InputBox, or creating a SQL string.

```
Dim Msg as String
Msg = "This is a paragraph that is to be "
Msg = Msg & "in a message box. The text is "
Msg = Msg & "broken into several lines of code"
Msg = Msg & "in the source code, making it easier"
Msg = Msg & "for the programmer to read and debug."
MsgBox Msg

Dim QRY as String
QRY = "SELECT *"
QRY = QRY & " FROM Titles"
```

8 Part # Part Head

```
QRY = QRY & " WHERE [Year Published] > 1988"
TitlesQry.SQL = QRY
```