



CITY *of* BRISTOL  
COLLEGE

College Green Centre  
Faculty of Computing and Digital Communication  
Document Standards:  
**Visual Basic 6 House Style**

|                         |                          |
|-------------------------|--------------------------|
| Document No: DS05/VB6HS | Dated: 24 April 2005     |
| Issue: 01               | Author(s): Alan McTavish |

# Contents

|   |   |
|---|---|
| 1. Purpose .....                                    | 3 |
| 2. Scope.....                                       | 3 |
| 3. The Coding Practice.....                         | 3 |
| 3.1. Naming Conventions .....                       | 3 |
| 3.2. Variable Naming .....                          | 3 |
| 3.3. Hungarian Type Prefixes:.....                  | 3 |
| 3.4. Control Naming .....                           | 4 |
| 3.5. Constant Naming .....                          | 4 |
| 3.6. Procedure Naming.....                          | 4 |
| 3.7. Module Naming .....                            | 5 |
| 4. Use of Variables, Procedures and Constants ..... | 5 |
| 4.1. Variables .....                                | 5 |
| 4.2. Procedures .....                               | 5 |
| 4.3. Constants .....                                | 5 |
| 5. Code Layout.....                                 | 6 |
| 5.1. Commenting Code.....                           | 6 |
| 5.2. Formatting Code .....                          | 6 |
| 5.3. Other Coding Rules.....                        | 7 |
| 6. Sample Templates.....                            | 7 |
| 6.1. Module Template:.....                          | 7 |
| 6.2. Procedure Template:.....                       | 7 |

## 1. Purpose

This working document defines a coding practice for the Visual Basic programming language (version 5 or 6) to;

- 1.1 Ensure that Visual Basic code is written to a consistent style and structure
- 1.2 Support the production of reliable, readable and maintainable code.

## 2. Scope

- 2.1. All students studying at the City of Bristol College are required to work to these standards unless specifically told otherwise by a lecturer.
- 2.2. These standards are being published to help Students to produce work that is acceptable to all lecturers in the college.
- 2.3. Where the student decides to deviate from the standards, this must be for a good reason that must be documented within the code as a comment.
- 2.4. Where code is imported, for instance, from the Visual Basic Windows API Viewer or from MSDN, it is NOT a requirement to change the code to reflect these standards. Although comments should be included stating that the code *is* imported and from where it was imported.
- 2.5. These standards only apply to new work; it is not a requirement that they be applied to existing code.

## 3. The Coding Practice

### 3.1. Naming Conventions

Providing a standard for naming variables, constants, functions, controls and modules makes your projects and code easier to follow by lecturers and other students following the same standard.

### 3.2. Variable Naming

Variable names shall describe their use, shall be in mixed case with an initial upper case character for each word, and shall have 'Hungarian' prefixes that describe their type and scope. Abbreviations shall be avoided, but where they are used they are to be used consistently throughout the entire application.

Hungarian prefixes not only identify the type of a variable but also have the advantage of avoiding confusion between variables and functions, methods or properties.

### 3.3. Hungarian Type Prefixes:

| Data Type               | Prefix | Example           |
|-------------------------|--------|-------------------|
| Boolean                 | b      | blsValid          |
| Byte                    | byt    | bytAge            |
| Currency                | cur    | curPrice          |
| Date (and Time)         | dt     | dtStart           |
| Double                  | dbl    | dblScale          |
| Integer                 | int    | intCounter        |
| Long                    | lng    | lngHighCounter    |
| Single (floating point) | sng    | sngSize           |
| String                  | str    | strDepartmentName |
| Variant                 | var    | varUnknown        |
| Object                  | obj    | objCustomer       |

Arrays shall have no special prefix, but shall have a plural name, or a name that ends with the word 'Array', for instance, sNames or ICustomersArray.

### 3.4. Control Naming

Visual Basic control names will have a three character Hungarian prefix as set out below.

| Controls              |             |
|-----------------------|-------------|
| Check Boxes           | chk         |
| Combo Box             | cbo         |
| Command Button        | cmd         |
|                       | Or .... btn |
| Common Dialog Control | dlg         |
| Data                  | dat         |
| Form                  | frm         |
| Frame                 | fra         |
| Grid                  | grd         |
| Image                 | img         |
| Label                 | lbl         |
| List Box              | lst         |
| Menu                  | mnu         |
| Option Button         | opt         |
| Picture               | pic         |
| Shape                 | shp         |
| Text Box              | txt         |
| Timer                 | tmr         |

Examples:

Check box: chkChoices

Command Button: cmdShowList

Label: lblAuthor

Text box: txtFirstName

Boolean variable: blnDisplayed

String variable: strAddress

Image: imgLogo

Integer: intYrGraduated

Option button: optBuick

Timer: tmrMove

### 3.5. Constant Naming

Constants will have names that describe their use, and they shall be in upper case with underscores separating each word. They should utilise a Hungarian type prefix, in the same manner as prefixes for variables.

### 3.6. Procedure Naming

Procedure shall have mixed case descriptive names where the initial character of each word is in upper case. The names shall only exceptionally contain

abbreviations, and where abbreviations are used they shall be used consistently throughout the application. An example would be;

```
Public Sub GetData()
```

### **3.7. Module Naming**

Modules, classes and forms shall have descriptive names. They shall also have a descriptive prefix in lower case; frm for forms, mod for standard modules, and cls for class modules. This in particular helps distinguish the name of form modules from variables referencing a form instance.

## **4. Use of Variables, Procedures and Constants**

### **4.1. Variables**

Variables shall be explicitly declared, and to this end Option Explicit is to appear at the top of all modules.

Object variables are not to be declared 'As New', rather they will be declared as the type of object and when an instance is required it will be created with the 'Set ... = New ' or by using CreateObject as appropriate.

The generic form variable that is automatically created by VB with the same name as the form modules is not to be used, rather, whenever an instance of a form is required a specific variable declared as the form type will be used.

Variables shall be declared at the narrowest scope possible. Global scope modules are to be avoided.

Local variables shall be declared at the top of a procedure, not within the body.

The Variant type shall only be used when no other more specific type is possible.

Variables shall have a focused use. They shall not be declared once and used for several different purposes.

String variables shall be concatenated with the ampersand character (i.e. & not +)

When naming boolean variables, the positive rather than the negative form shall be preferred, i.e. blsValid rather than blsNotValid.

### **4.2. Procedures**

Procedures shall have an appropriate clearly defined scope (Public, Private or Friend) and not rely on the default setting.

Procedures shall have as few exit points as possible; Exit Function/Sub and Property are to be avoided unless it improves the readability.

Procedures shall have robust exit code where object references are explicitly tidied up rather than relying on Visual Basic to clean up as they go out of scope.

Parameters shall have a specified data type.

Data will be passed to and from procedures with parameters rather than module or global scope variables.

Parameters with a small range of values shall be declared with an enumeration type.

Where parameters are not to be changed they shall be declared ByVal unless there is some performance reason to declare them ByRef.

### **4.3. Constants**

System constants and enumerations shall be used wherever possible, for example, vbMinimized.

Constants shall not be declared in multiple places, rather they shall be declared at a higher scope i.e. in a module.

## 5. Code Layout

### 5.1. Commenting Code

A comment shall specify what the purpose of a piece of code is, it shall not simply repeat the code.

All deviations from coding standards or good programming practice shall have a comment specifying the reasons why.

Comments shall precede the relevant code and shall be indented to the same level as the code that follows them.

End-of-line comments shall be avoided, the exception being in the declaration section if a comment for a variable declaration is required.

Comments shall be specified with an apostrophe followed by a space and not with the antiquated Rem statement.

Every procedure and module shall have a Template ( a comment section) that describes its purpose and parameters.

Solid comment lines (for instance '\*\*\*\*\*') are not to be used except in module or procedure Templates. The exception is during development when sections of code are waiting to be modified, improved or completed.

Document all decision and loop structures.

### 5.2. Formatting Code

The indent tab setting in the Visual Basic Options shall be left at the default setting of 4 spaces.

Indenting and white space shall be used to make the code more readable, like the paragraphs of a book.

No more than one statement shall appear on a single line.

There should be a blank line between any variable declarations and the first line of code in a function or procedure and another between the last line of code and the 'End Sub' statement.

Use the continuation character ( \_ ) to split long lines to make them more readable.

The continuation lines shall be indented.

Indent code as follows:

- Between an If statement and its End If, Else or Elseif.
- Between an Else and its End If.
- Between an Elseif and its Else or End If.
- Between a Select statement and its End Select statement.
- Between each Case statement in a Select.
- Between a Do statement and its Loop.
- Between a With statement and its End With.
- Between a For statement and its Next.
- Between an Edit or AddNew method and its Update or CancelUpdate.
- Between the start and end of a transaction.
- Within the declaration section to show subordination, for instance between a Type and its End Type.

### 5.3. Other Coding Rules

Every procedure shall contain error handling to capture and re-raise or report errors.

The most specific type of object variable shall be used in a For Each..Next loop. Always include a Case Else in Select Case structures.

Split an If statement over several lines and use an End If even where there is only one statement to execute.

Never compare a boolean for equality with True or False.

Use parenthesis to improve readability even when they are strictly not required.

Use upper case letters for GoTo labels.

Use GoTo only where it improves readability.

Do not use GoSub.

Do not use default properties, rather specify them explicitly, for example txtName.Text = sName

The reference counter used in a For..Next loop shall not be used after the loop has finished.

The antiquated While..Wend loop shall not be used.

Compile On Demand is not to be used in the Visual Basic options.

DoEvents shall be avoided unless required, specifically, where a control or form is to be refreshed the Refresh method shall be used in preference to DoEvents.

## 6. Sample Templates

### 6.1. Module Template:

```
*****
' MODULE:      frmMain
' FILENAME:    C:\Project\frmMain.frm
' AUTHOR:      A. N. Other
' CREATED:     15-Mar-2005
' COPYRIGHT:   Copyright 2005 City of Bristol College.
'              All Rights Reserved.
' DESCRIPTION:
' ***Description goes here***
'
' MODIFICATION HISTORY:
' 1.0      15-Mar-2005
'          A. N. Other
'          Initial Version
*****
```

### 6.2. Procedure Template:

```
*****
' GetUserName (FUNCTION)
'
' PARAMETERS:
' (In) - IUserID - Long -
'
' RETURN VALUE:
' String -
'
' DESCRIPTION:
' ***Description goes here***
*****
```