JUL 26 2000
Station 15

21 ENGINEERING DATA TRANSMITTAL

Page 1 of 1 KN

1. EDT 629502

HMFS0141 105592/AJ70

2. To: (Receiving Organization) SNFP Operations Support	3. From: (Originating Organization) LMSI Software Development & Int.	4. Related EDT No.: N/A
5. Proj./Prog./Dept./Div.: MAC-II	6. Design Authority/Design Agent/Cog. Engr.: T. G. Ibsen/M. J. Winkelman	7. Purchase Order No.: N/A
8. Originator Remarks: For approval and release		9. Equip./Component No.: N/A
		10. System/Bldg./Facility: 2261 Stevens
		12. Major Assm. Dwg. No.: N/A
11. Receiver Remarks: None	11A. Design Baseline Document? Yes [✓ No]	13. Permit/Permit Application No.: N/A
		14. Required Response Date: N/A

15. DATA TRANSMITTED

(A) Item No.	(B) Document/Drawing No.	(C) Sheet No.	(D) Rev. No.	(E) Title or Description of Data Transmitted	(F) Approval Designator	(G) Reason for Transmittal	(H) Originator Disposition	(I) Receiver Disposition
1	HNF-6445		0	Software Development Guidelines for Visual Basic and SQL Server	Q	1	1	1

16. KEY

Approval Designator (F)	Reason for Transmittal (G)	Disposition (H) & (I)
E, S, Q, D OR N/A (See WHC-CM-3-5, Sec. 12.7)	1. Approval 2. Release 3. Information 4. Review 5. Post-Review 6. Dist. (Receipt Acknow. Required)	1. Approved 2. Approved w/comment 3. Disapproved w/comment 4. Reviewed no/comment 5. Reviewed w/comment 6. Receipt acknowledged

17. SIGNATURE/DISTRIBUTION
(See Approval Designator for required signatures)

(G) Reason	(H) Disp.	(J) Name	(K) Signature	(L) Date	(M) MSIN	(G) Reason	(H) Disp.	(J) Name	(K) Signature	(L) Date	(M) MSIN
		Design Authority				3	6	C. D. Lucas	[illegible]	7-10-00	
		Design Agent				1	1	M. J. Winkelman	[illegible]		
1	1	Cog. Eng. T. G. Ibsen	[illegible]	6/28/00		3	6	K. R. Morris	[illegible]	7/6/00	
1	1	Cog. Mgr. K. J. Willers	[illegible]	6/29/00		3	6	L. L. Blehm	[illegible]	6/30/2000	
1	1	QA D. D. Scott	[illegible]	6/29		3		SNF Project File			X3-85
		Safety				3		Central Files			B1-07
		Env.				3		DOE Reading Room			H2-51

18. Signature of EDT Originator / Date	19. Authorized Representative for Receiving Organization / Date	20. [illegible] 6/29/2000 Design Authority/ Cognizant Manager / Date	21. DOE APPROVAL (if required) Ctrl No. ___ ◯ Approved ◯ Approved w/comments ◯ Disapproved w/comments

BD-7400-172-2 (10/97)

BD-7400-172-1

CORRESPONDENCE DISTRIBUTION COVERSHEET

Author	Addressee	Correspondence No.
T. G. Ibsen R. J. South K. R. Stafford M. J. Winkelman		HNF-6445, Rev. 0

Subject: SOFTWARE DEVELOPMENT GUIDELINES FOR VISUAL BASIC AND SQL SERVER

DISTRIBUTION

Approval	Date	Name	Location	w/att
		Correspondence Control	A3-01	X
		Central Files	B1-07	X
		DOE-RL Reading Room	H2-53	X
		L. L. Blehm	H8-60	X
		T. G. Ibsen	H8-41	X
		C. D. Lucas	X3-74	X
		K. R. Morris	X3-74	X
		D. D. Scott	H8-43	X
		R. J. South	H8-41	X
		K. R. Stafford	H8-41	X
		K. J. Willers	H8-41	X
		M. J. Winkelman	H8-41	X
		SNF Project File	X3-85	X

HNF-6445
Revision Rev 0

Software Development Guidelines for Visual Basic and SQL Server

Prepared for the U.S. Department of Energy
Assistant Secretary for Environmental Management

Project Hanford Management Contractor for the
U.S. Department of Energy under Contract DE-AC06-96RL13200

Fluor Hanford
P.O. Box 1000
Richland, Washington

HNF-6445
Revision Rev 0
EDT 629502

Software Development Guidelines for Visual Basic and SQL Server

Document Type: EPRO Division: SNF

T. G. Ibsen
Lockheed Martin Services, Inc.

R. J. South
K. R. Stafford
M. J. Winkelman
Lockheed Martin Services, Inc.

Date Published
June 2000

Prepared for the U.S. Department of Energy
Assistant Secretary for Environmental Management

Project Hanford Management Contractor for the
U.S. Department of Energy under Contract DE-AC06-96RL13200
Fluor Hanford
P.O. Box 1000
Richland, Washington

JUL 26 2000
DATE:
STA: 15
HANFORD RELEASE
ID:
21

Karen Noland 7/26/00
Release Approval Date Release Stamp

Total Pages: 27 KN

SOFTWARE DEVELOPMENT GUIDELINES for VISUAL BASIC AND SQL SERVER

Last Update
June 2000

This page intentionally left blank.

CONTENTS

TERMS

ADO	Active Data Objects
DAO	Data Access Objects
MSSQL	Microsoft SQL Server™
RDO	Remote Data Objects
SQL	Structured query language
UDT	User-defined type
VB	Visual Basic™ development environment

DEFINITIONS

Major Version	Major versions change the scope and functionality of the system being developed. Default major version in Visual Basic™ is 1.x.x.
Minor Version	Minor versions are changes that add enhancements to existing functionality, or correct extensive problems in the system. Default minor version in Visual Basic™ is 0.x.1.x. A new software release to the customer will require a minor version change.
Revisions	Revisions are corrections of bugs found during testing, development and use. Default revision in Visual Basic™ is 0.x.x.1.
Stored Procedure	Stored procedures are sets of logical statements (a program) that is stored within a database. These may be executed from within the database by specific events or on a specific schedule, or on demand from another source, such as a Visual Basic™ program.
Tracking Number	A number assigned to a Change Control for tracking a specified problem or enhancement.
Trigger	A trigger is a stored procedure that is executed from an event within the database.
UDT	User-defined type contains one or more element of any data type specified by the developer.

1.0 INTRODUCTION

Development Guidelines are programming directions that focus not on the logic of the program but on its physical structure and appearance. These directions make the code easier to read, understand, and maintain. These guidelines are put in place to create a consistent set of conventions to follow that will standardize the development process. With these guidelines in place the readability and understanding others have when reviewing the code is greatly enhanced. Use these guidelines as a general rule when writing any set of logical statements.

1.1 PURPOSE

Development Guidelines are put into place in an effort to standardize the structure and style of the development process. They are NOT intended to limit or channel the developer's own creativity and flexibility.

1.2 SCOPE

These guidelines will cover general development syntax, organization and documentation. The general information covers the high level areas of development, no matter what the environment. This guide will detail specific Visual Basic[1] guidelines, following the same standard naming conventions set by Microsoft[2], with some minor additions. The guideline will finish with conventions specific to a Database or Microsoft's SQL Server[3] specific environment.

2.0 GENERAL GUIDELINES

This section covers the generic guidelines that apply to most logical programming methods.

2.1 FUNCTIONAL OVERVIEW COMMENT (HEADER)

All procedures, functions, subroutines, and instruction sets will be preceded with a header summarizing code being written. The information detailed below is the basic structure required. Other information may be included in the header that is not covered in this document. These added items are left to the discretion of each programmer, but the following core pieces of information must be included.

1 Visual Basic is a trademark of Microsoft Corporation, Redmond, Washington.
2 Microsoft is a registered trademark of Microsoft Corporation, Redmond, Washington.
3 SQL Server is a trademark of Microsoft Corporation, Redmond, Washington.

2.1.1 Functional Overview Comment (Header)

Each heading will list out any assumptions being made in the procedure. This list will show any information that is not obvious to the programmer when reviewing this section of the code. These assumptions are most likely objects such as an open file, database variable, control, global constant, etc. that are set outside of this procedure.

2.1.2 Passed Information

Each header will list any passed information and the format of that information. Such as "sTableName – name of a database table" or "sLastName – Last Name of an employee". By addressing each passed piece of information, the developer reviewing your code, knows the intent of that data structure.

2.1.3 Purpose

The heading will contain a brief description of the basic functionality being performed. An example would be as follows:

> "Purpose: This function generates a recordset of data based on the fields passed."

This information is important when trying to determine what is being done in the routine. Developers can generate many different solutions for the same result and this brief description provides the person reviewing the code a basic idea of what is being accomplished. In the purpose statement describe WHAT is being done, do not detail HOW the process is accomplished,

2.1.4 Created By and Date

Following the purpose, there should be the name of the developer who created the procedure and the date. (e.g., "Created by: John Doe April 7, 2000").

2.1.5 Revisions

After the "Created By" information in the header, there should be a listing of revisions made to this procedure. The revision line should include the date, version, developers name and description of the revision. The most current revision should go at the top of the list. Such as:

06/04/98 1.0.1	Jane Doe - Corrected overflow on iCount variable, by changing it to a long integer lngCount.
06/03/98 1.0.0	John Doe - Originally Written

When tracking numbers are being used for problems or enhancements, follow the revision number with the tracking number as follows:

06/04/98 1.0.1(1234)	Jane Doe - Corrected overflow on iCount variable, by changing it to a long integer lngCount

2.1.6 Example

```
'************************************************************************
'*
'*Assumptions:
'*      File number being passed already references an open file.
'*
'* Passed Information:
'*      dDate, Date of transaction
'*      iFileNum, File Handle Number for data output
'*
'*Purpose:
'*      This function uses the variable passed, converts it to a special YYYYMMDD
'*      date, and writes the result to an ASCII file.
'*
'*Created by:
'*      John Doe       06/04/98

'*Revisions:
'*      06/04/98  1.0.0       John Doe      Originally written
'*
'************************************************************************
```

2.2 COMMENTS

Comments must be written throughout the application. These comments are designed to give general direction and explanation of what is being accomplished in that particular section of the code. An example would be the underlined text below:

' <u>Display the hourglass mouse pointer</u>.
Screen.MousePointer = vbHourglass

There is no hard and fast rule on adding comments, but generally the more explanation the better. Try to write the comments in general terms to provide the individual, who is reading your code, an "introduction" to the next set of instructions. In addition to introducing your code, any further description should be placed before the line of code or at the end of the line; for example, the following underlined text:

```
'Open Employee File
Open sFileName for output as iFileNum
Or
Open sFileName for output as iFileNum  'Open Employee File
```

Keep in mind that the purpose of comments is to make the code more understandable, and more readable. When the code readability begins to degrade because of excessive comments, the comments become significantly less useful. Where possible, summarize.

2.2.1 Revision Remarks

When remarking a revision, note the change in the header, but also put the revision number in a remark preceding the correction, such as the underlined text below:

```
'1.0.1 Corrected Overflow with iCount by changing data type to long
Dim lngCount as Long
```

In some cases a tracking number from a change control system is assigned to the revision being made. In this case the tracking number should follow the revision number as follows:

```
'1.0.1(1234) Corrected Overflow with iCount by changing data type to long
Dim lngCount as Long
```

2.3 FORMATTING YOUR CODE

The format of your code should be nested together to best describe the logical flow of your program.

- Standard, tab-based, and nested blocks should be indented four spaces which is the default tab setting for the Microsoft Visual Basic Editor. (However, this document uses the default MS Word tab settings – so may be off from the same code in VB)

- The header comment should be indented one space from the preceding asterisk. The comment body under the header should be indented one tab. Comments on a particular block of code should be indented the same number of tabs as the code being commented. For example:

```
'*************************************************
'* Assumptions:
	'*	None
'*
'* Purpose:
'*	Returns the first occurrence of a specified user in the UserList array.
'
'* Passed Information:
'*	saUserList():  List of users to be searched.
'*	sTargetUser:  Username to search for
'*
'*Returns:
'*	The index of the first occurrence of the strTargetUser in the straUserList array.
'*	 If target user is not found, return -1.
'*
'* Created by:
'*	John Doe	06/04/98
'*
'* Revisions:
	'*06/04/98	1.0.1	John Doe Originally Written

'*************************************************
Function iFindUser(saUserList() As String, sTargetUser As String)As Integer
Dim iI as Integer		' Loop counter.
Dim bFoundas Boolean		' Target found flag.

iFindUser = -1
iI = 0
'Start While Loop
While iI <= Ubound(saUserList) and Not blnFound
	'Nested If Statement
	If saUserList(intI) = sTargetUser Then
		bFound = True
		iFindUser = iI
	End If
Wend
End Function
```

2.3.1 Grouping Constants

Variables and defined constants should be grouped by function rather than split into isolated areas or special files. Visual Basic™ generic constants should be grouped in a single module to separate them from application-specific declarations.

2.3.2 & and + Operators

Use the & operator when connecting strings and the + operator when working with numerical values. Using the + operator 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
```

2.3.3 Creating Strings for MsgBox, InputBox, and SQL Queries

When creating a long string, use the underscore line-continuation character to create multiple lines of code so that you can read or debug the string easily. This technique is particularly useful when displaying a message box (MsgBox) or input box (InputBox) or when creating an SQL string. For example:

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

Dim sQRY As String
sQRY = "SELECT *" _
        & " FROM Titles" _
        & " WHERE [Year Published] > 1988"
TitlesQry.SQL = sQRY
```

2.4 NAMING

When naming any variable, constant, stored procedure, table, or field, the programmer must use common sense and select a name that applies to the object being written. No slang or colorful naming unique to that programmer will be accepted. The naming should try to reflect the scope of information being written. For example:

```
'Declare Variables
Dim iRecCount as Integer               'Record Count
Dim recEmpRecordset as Recordset       'Employee Recordset
Dim  bExitFlag as Boolean              'Exit Flag
```

2.5 ERROR HANDLING

All trappable errors need to be handled within an error handling section of code. The use of any "On Error Resume Next" method of error handling is only acceptable in simple procedures, where extensive error handling would not be helpful in solving the problem.

Where applicable make the label for the error handling method by using the follow suffix "_EH:", such as:

```
Sub EmployeeName
 On Error goto EmployeeName_EH:
        'Program
Exit Sub
EmployeeName_EH:
'Handle Error
End Sub
```

3.0 VERSION NUMBERS

Version numbers used for released software will follow the format, X.X.X (Major. Minor.Revision). Version 1.x.x will be the initial release of the software. During development, we will follow the above format, but add additional numbers, X.X.X(x) (Major.Minor.Revision(Change Control Number)). The change control number will be used only for re-marking corrections or enhancements tracked in a change control system. Numerous bugs may be corrected to create a 1.2.5 revision, and to locate the exact fix, the tracking number is used.

Visual Basic (6.0) will automatically increment the revision number if that option is set at the project level. A new revision number is assigned automatically with every successful compile of the program. When a series of revisions is approved for release to the customer, the minor version (or major version) will be manually incremented, and the revision manually reset to zero.

4.0 VISUAL BASIC™

This section covers the specific guidelines used with Visual Basic™.

4.1 THE STANDARD

Microsoft® has set the basic standards for writing with their Visual Basic™ development environment. These guidelines will be followed on this project, with some slight modifications.

4.1.1 Visual Basic™ Coding Conventions

All objects will be named using the proper prefix. This naming will make it easy to identify each object in the code. The list of prefixes below covers most of the major objects in Visual Basic™.

Suggested Prefixes for Controls

Control Type	**Prefix**	**Example**
3D Panel	pnl	PnlGroup
Animated button	ani	AniMailBox
Check box	chk	ChkReadOnly
Combo box, drop-down list box	cbo	CboEnglish
Command button	cmd	CmdExit
Common dialog	dlg	DlgFileOpen
Communications	com	ComFax
Control (used within procedures when the specific type is unknown)	ctr	CtrCurrent
Data control	dat	DatBiblio
Data-bound combo box	dbcbo	DbcboLanguage
Data-bound grid	dbgrd	DbgrdQueryResult
Data-bound list box	dblst	DblstJobType
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

Control Type	**Prefix**	**Example**
Horizontal scroll bar	hsb	HsbVolume
Image	img	ImgIcon
Key status	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
MS Flex grid	msg	MsgClients
MS Tab	mst	MstFirst
OLE (ActiveX)	ole (actx)	OleWorksheet (actxWorksheet)
Outline	out	OutOrgChart
Pen BEdit	bed	BedFirstName
Pen Hedit	hed	HedSignature

Control Type	**Prefix**	**Example**
Pen ink	ink	InkMap
Picture	pic	PicVGA
Picture clip	clp	ClpToolbar
Report	rpt	rptQtr1Earnings
Shape	shp	ShpCircle
Spin	spn	SpnPages
Text box	txt	TxtLastName
Timer	tmr	TmrAlarm
UpDown	upd	UpdDirection
Vertical scroll bar	vsb	VsbRate
Slider	sld	SldScale
ImageList	ils	IlsAllIcons
TreeView	tre	TreOrganization

Control Type	Prefix	Example
Toolbar	tlb	TlbActions
TabStrip	tab	TabOptions
StatusBar	sta	StaDateTime
ListView	lvw	LvwHeadings
ProgressBar	prg	PrgLoadFile
RichTextBox	rtf	RtfReport

4.1.2 Data Access Object Naming

The following prefixes cover Data Access Objects (DAO).

Database object	Prefix	Example
Container	con	ConReports
Database	db	DbAccounts
DBEngine	dbe	DbeJet
Document	doc	DocSalesReport
Field	fld	FldAddress
Group	grp	GrpFinance
Index	idx	IdxAge
Parameter	prm	PrmJobCode
QueryDef	qry	QrySalesByRegion
Recordset	rec	RecForecast
Relation	rel	RelEmployeeDept
TableDef	tdef	TdefCustomers
User	usr	UsrNew
Workspace	wsp	WspMine

Some examples:

```
        Dim dbOSSB As Database
        Dim recEmp As Recordset, sSQLStmt As String
        Const dbReadOnly = 4                    ' Set constant.
        'Open database.
        Set dbBiblio = OpenDatabase("OSSP.MDB")
                ' Set text for the SQL statement.
                sSQLStmt = "SELECT * FROM Employees WHERE Last_Name = "Doe"
                ' Create the new Recordset object.
                Set recEmp = db.OpenRecordset(sSQLStmt, dbReadOnly)
```

4.1.3 Suggested Prefixes for Menus

When generating menus, use the following naming conventions for the specific controls. Note that each menu control prefix is extended beyond the "mnu" label by adding an additional prefix for each level. The ending name for each menu should detail the action being performed. The following table provides the following:

Menu Caption Sequence	**Menu Handler Name**
File Save	MnuFileSave
Edit Copy	MnuEditCopytoClipboard
Edit Find	MnuEditFind
Help Contents	MnuHelpContents

When this naming convention is used, all members of a particular menu group are listed next to each other in Visual Basic's™ Properties window. In addition, the menu control names clearly document the menu items to which they are attached.

4.1.4 Choosing Prefixes for Other Controls

For controls not listed above or third party controls, a standard prefix should be created that is different than any already set in place for other controls. Make the prefix two or more characters which are consistent with the specific control's functionality and a prefix that provides a clear understanding.

Note: For any controls that a new prefix is created, please inform the technical lead to have this document updated.

4.1.5 Constant and Variable Naming

In addition to objects, constants and variables also require well-formed naming conventions. This section lists the recommended conventions for constants and variables supported by Visual Basic.™ It also discusses the issues of identifying data type and scope.

Variables should always be defined with the smallest scope possible. Global (Public) variables can create enormously complex state machines and make the logic of an application extremely difficult to understand. Global variables also make the reuse and maintenance of your code ~~much~~ more difficult.

Variables in Visual Basic™ can have the following scope:

Scope	Declaration	Visible In
Procedure-level	'Private' in procedure, sub, or function	The procedure in which it is declared
Module-level	'Private' in the declarations section of a form or code module (.frm, .bas)	Every procedure in the form or code module
Global	'Public' in the declarations section of a code module (.bas)	Everywhere in the application

4.1.6 Variable Scope Prefixes

To separate a scope prefix from the normal variable prefix, a scope prefix will be followed with an underscore character.

The use of variables in applications can become extensive, and to identify properly the scope of each variable, the following prefixes are used.

Scope	Prefix	Example
Public or Global	G_	G_sUserName
Module-level	M_	M_blnCalcInProgress
Local	None	dblVelocity

The use of these prefixes does not declare the variables scope; it just describes how this variable was declared in the application. This is extremely useful and prevents developers from having to "track down" the scope of these variables.

Note: The Visual Basic™ syntax checker will not catch any inconsistent use of prefixes. Such as declaring a public-level variable that begin with a "M" or a "G"

4.1.7 Proper Location to Declare Variables

In most cases, the location where a variable is declared gives that variable scope: The exception applies to public (global) variables. When declaring a public variable, this should be done always in the declaration section in a single module, grouped by function. Give that module a meaningful name that indicates its purpose, such as modPublic.bas. Global variables should be used only when there is no other convenient way to share data between forms or modules.

4.1.8 Variables

It is important that every project require "Option Explicit" which enforces the declaration of ALL variables in the project. This saves programming effort and avoids any problems caused by typos (e.g., saUserNameTmp vs. sUserNameTmp vs. sUserNameTemp). To activate either go to the Editor tab of the Options dialog, check the Require Variable Declaration option or use the "Option Explicit" statement in your project. By doing this, Visual Basic™ will require the declaration of all variables.

4.1.9 Variable Data Types

Use the following prefixes to indicate a variable's data type.

Data type	**Prefix**	**Example**
Boolean	bln	bFound
Byte	byt	bytRasterData
Collection object	col	colWidgets
Currency	cur	curRevenue
Date (Time)	d	dStart
Double	Dbl	dblTolerance
Error	Err	errOrderNum
Integer	i	intQuantity
Long	lng	lngDistance
Object	obj	objCurrent
Single	f	fAverage
String	s	sFName
User-defined type	u	udtEmployee
Variant	vnt	vntCheckSum

Also it is allowable and sometimes easier to use character specific data types. Note that not all data types have a character declaration. For those you must follow, use the following suffix naming:

Data type	Suffix	Example
Currency	@	Revenue@
Double	#	Tolerance#
Integer	%	Quantity%
Long	&	Distance&
Single	!	Average!
String	$	FName$
Variant	None	CheckSum

4.1.10 Arrays

To indicate an array, append a lowercase "a" after the normal variable prefix.

Example:

```
Dim saName() as String      'Array of names
Dim faSalary() as Single    'Array of salary values
```

4.1.11 Constants

The body of constant names should be mixed case with capitals initiating each word. For constant naming, follow the same rules as variables, but add a C.

For example:

```
MC_intUserListMax      'Max entry limit for User list
                       '(Module level constaint,integer value,local to module)
GC_strNewLine          'New Line character
                        '(string, global to application)
cstrName
```

4.1.12 Descriptive Variable and Procedure Names

The body of a variable or procedure name should use mixed case and should be as long as necessary to describe its purpose. In addition, function names should begin with a verb, such as InitNameArray or CloseDialog. For frequently used or long terms, standard abbreviations are recommended to help keep name lengths reasonable. In general, variable names greater than 32 characters can be difficult to read on VGA displays. When using abbreviations, make sure these are consistent throughout the entire application. Randomly switching between Cnt and Count within a project will lead to unnecessary confusion.

4.1.13 User-Defined Types

User-defined types (UDT) should be given a three-character prefix starting with the letter "u". This makes the UDT easy to identify and locate. For example: "uemp" could be used as the prefix for a user-defined employee type.

4.1.14 Passed Variables

By default, Visual Basic™ passes all variables by Reference (which means the original data changes when changes are made inside the procedure) Whenever a variable is to be passed to a procedure by Value (so the original data is unchanged despite what happens in the procedure) that variable must be declared to be BYVAL in the procedure being called.

An example:

```
'Function passing by Value variable to UpdateMyAge procedure
Sub SetAge ()
'Dim local Variables
Dim intMyAge as integer
        'Call Procedure to update my age
        Call UpdateMyAge(intMyAge)
        If intMyAge<18 Then
                'Conditional Processing Code Goes Here.
        End If
End Sub
                '
Sub UpdateMyAge (By Val intMyAge)
        'Set age to referenced variable
        G_intMyAge  = intMyAge
        End Sub
```

4.2 NAMING OBJECT FILES

Some objects in Visual Basic™ require the developer to save them to a file, (e.g., forms, modules and class modules). The guidelines below outline the naming of these objects and the name of the related file being saved.

4.2.1 Modules

When naming modules in Visual Basic™, use the prefix mod_, (e.g., modMain). When saving this file to the project, use the same naming, (e.g., modMain.bas).

4.2.2 Class Modules

When naming class modules in Visual Basic™, use the prefix cls, (e.g., clsMain). When saving these files to the project, use the same naming (e.g., clsmain.bas).

4.2.3 Forms

When naming forms in Visual Basic™, use the prefix frm as noted above. When saving the form files to the project, use the same prefix, (e.g., frmMain.frm).

4.2.4 Mdi Forms

When naming midi forms in Visual Basic™, use the prefix mdi as noted above. When saving mdi form files to a project, use the same prefix, (e.g., mdiMain.frm).

5.0 GENERAL DATABASES AND MICROSOFT SQL SERVER™

This section will cover some specific guidelines related to Microsoft's SQL Server™ or basic database design.

5.1 DATABASE DESIGN

The database design should follow a basic set of rules. When naming tables, fields, views, queries, stored procedures, etc. the programmer needs to avoid the use of special characters (not including underscores '_'), and keep the naming of the objects descriptive.

5.1.1 Tables

The two sections below will cover the guidelines for naming tables. Keep the table names inline with the data being stored.

5.1.1.1 Permanent Tables. Permanent tables have no prefixes.

Naming Convention:	TableName
Example:	Users

5.1.1.2 Temporary Tables. Temporary tables use the prefix tmp.

Naming Convention:	tmpTableName
Example:	tmpSummaryStats

For SQL Server™ temporary tables, make sure the table name is preceded with a #. The SQL Server™ will automatically drop the table upon termination of the connection/session/stored procedure. To remain consistent, keep the tmp prefix as follows:

Naming Convention:	#tmpTableName
Example:	#tmpSummaryStats

5.1.2 Fields

Fields represent specific data in a table. The guidelines below outline some simple practices to follow when naming these items. When naming fields, use the same name for the same field in different tables.

5.1.2.1 Naming. When naming a field, use a descriptor that best outlines the data being stored. Use a name that relates to what is being stored in the field, (e.g., First_Name, Last_Name).

5.1.2.2 Field ID's. When naming any field that is a unique ID or an index for a table, that field name should end with a suffix of ID, (e.g., Social_SecurityID, AccountID, EmployeeID).

5.1.3 Stored Procedures

When writing stored procedures, follow the same general rules above as they apply. In addition, the file name of the stored procedure should start with a stp_, then the name of the procedure, followed by the suffix .prc when saving to a file as follows.

Naming Convention:	stp_StoredProcedureName.prc
Example:	stp_CreateTempTable.prc

5.1.3.1 Additional Header Information. Apply header information from Section 2.1 above, but also include file name, procedure type, target tables and created objects as follows.

* File Name:	stp_Procedure.spc
* Procedure type:	Update
* Target Tables:	History, Users
* Created:	None

5.1.4 Rules

Any SQL Server™ rule needs to use the prefix rl as follows.

Naming Convention:	rl_Table_Column
Example:	rl_History_Date

5.1.5 Triggers

Any SQL Server™ trigger needs to use the prefix tg. The name of the trigger, followed by a suffix detailing the type of trigger written.

Trigger Suffix's:	
dl:	Delete Trigger
up:	Update Trigger
in:	Insert Trigger
Naming Convention:	tg_Name_XX
Example:	tg_History_up

5.1.6 Views

Any SQL Server™ view needs to use the prefix vw as follows.

Naming Convention:	vw_Table1_Table2 or vw_Description
Example:	vw_History_Current

6.0 REFERENCES

Microsoft, *VB 5.0 On-line Books: Visual Basic Coding Conventions, Microsoft Corporation.*