The following Help topics are available:

	Form Thing		Nudge Thing
	MsgBox Thing		Tab Thing
	CMDialog Thing		Dent Thing
	Type Thing		BlockOut Thing
	Call Thing		Compress Thing

For help on Help, press F1

ToolThings Help

Form Thing

MsgBox Thing

CMDialog Thing

Type Thing

Call Thing

Nudge Thing

Tab Thing

Dent Thing

BlockOut Thing

Compress Thing

When you first create a form for a window or dialog, a number of fundamental design questions should come to mind. Will the user want to resize the window? Does it need a maximize button? Minimize button? An icon? If the window is not sizable, should it have a border? Double or single? Should it have a title bar? What are the foreground and background colors? Do you need to change the mouse pointer?

Form Thing helps you decide these questions by letting you quickly develop, test, and modify a prototype of your form before you add it to your project. Once you've made the basic design decisions, just tell Form Thing to create a new form to your specification. Then you can shift your attention to other matters, such as adding controls and writing code.

For more information about Form Thing, press the More button.
For information about other Things, press the Contents button.

General

Colors/Fills

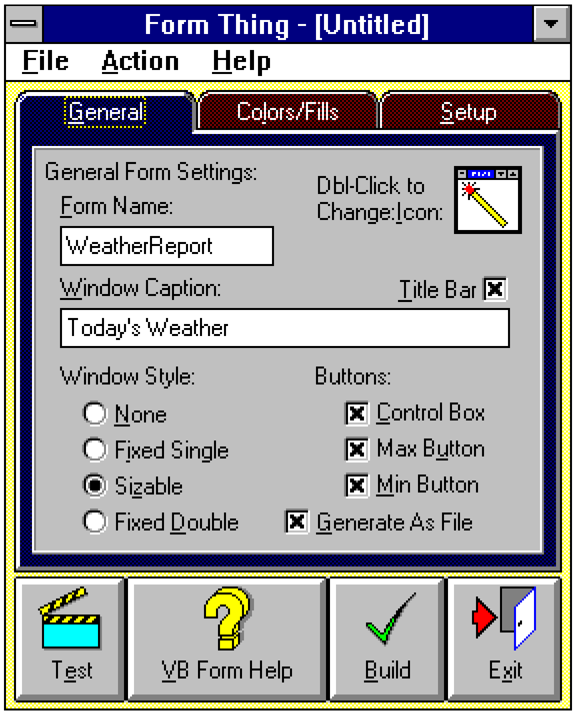
Setup

File

Action

Whenever you are in another topic, you can return to the overview by clicking this icon.

Click where you need help



General Form Settings:

Form Name: WeatherReport

Dbl-Click to Change:Icon:

Window Caption: Today's Weather

Title Bar [✓]

Window Style:

- ○ None
- ○ Fixed Single
- ● Sizable
- ○ Fixed Double

Buttons:

- [✓] Control Box
- [✓] Max Button
- [✓] Min Button

[✓] Generate As File

For more information about Form Thing, press the More button.
For information about other Things, press the Contents button.

Type a name for the form. The name can have as many as 40 characters but may not include spaces or punctuation. For more information, see the entry for the Name property in the Microsoft Visual Basic *Language Reference*.

(Optional) If your window will have a title bar, type the text you want to appear on the title bar.

If the Window Style you select permits a title bar but you do not want one, clear the Title Bar checkbox.

If your window includes a Minimize option, select an icon for the form by double-clicking the image labeled "Dbl-click to Change Icon." A File Open dialog appears and you can then browse for an icon file.

The image on Form Thing's General tab will change to reflect your choice. When Form Thing generates a new form and saves the FRM file, the icon will be embedded in the form's FRX file.

Note. This option is available only when the Generate As File checkbox is checked.

The window will have no border. A window with no border has no title bar and thus no Control box or Maximize and Minimize buttons.

A window with a Fixed Single border can include a Control box title bar, Maximize button, and Minimize button. The user will be able to minimize or maximize the window but will not be able to adjust the size by dragging the window border with the mouse.

(Default) The user will be able to resize the window using any of the optional elements listed in the Buttons group and also by dragging the window border with the mouse.

Forms with a Fixed Double border can have a title bar and Control box but cannot include Maximize or Minimize buttons. However, Maximize and Minimize items can be included on the Control menu.

With Window Style **None**, none of these options are available.

With Window Styles **Fixed Single** and **Sizable**, all are available.

With Window Style **Fixed Double**, Maximize and Minimize buttons are not available, but the options can be included on the Control box menu if a Control box is present.

Clear the checkbox marked "Generate As File" if you do not want a physical file created when the window is built.

See a prototype of your window.

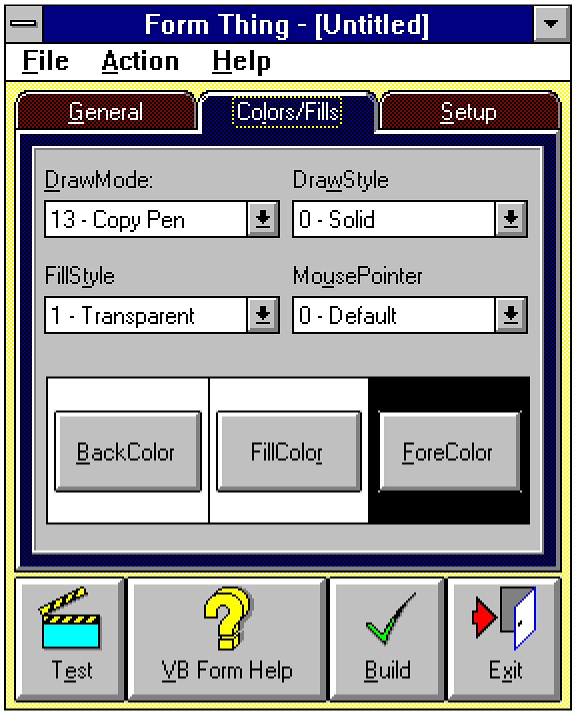
Click to display the Microsoft Visual Basic Help topic on forms.

Build the form. Form Thing adds the new form to your project.

If you have selected Generate As File, the Save Form As dialog appears. You can then name the file.

Quit.

Click where you need help


Form Thing - [Untitled]

File Action Help

General Colors/Fills Setup

DrawMode: 13 - Copy Pen

DrawStyle 0 - Solid

FillStyle 1 - Transparent

MousePointer 0 - Default

For more information about Form Thing, press the More button.
For information about other Things, press the Contents button.

Determines the appearance of output from graphics methods, or the appearance of a shape or line control.

Determines the line style for output from graphics methods.

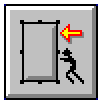
Determines the pattern used to fill shape controls as well as enclosed shapes created with graphics methods.

The mouse pointer you select will display when you place the mouse cursor over the prototype window.

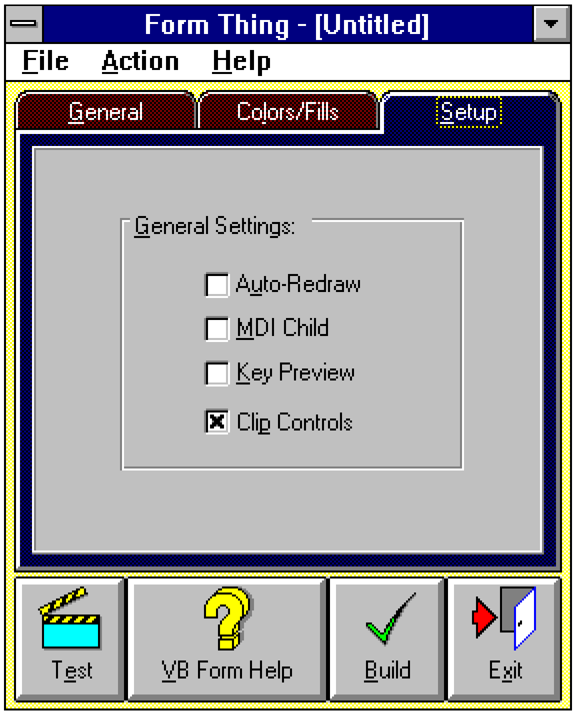
Click to select a background color for your form. Your selection is displayed in the area around the button and on the prototype form.

Click to select a fill color for your form. Your selection is displayed in the area around the button and on the prototype form.

Click to select a foreground color for your form. Your selection is displayed in the area around the button and on the prototype form.



Click where you need help



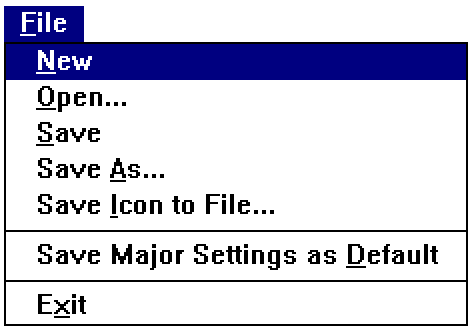
General Settings:

- [ ] Auto-Redraw
- [ ] MDI Child
- [ ] Key Preview
- [x] Clip Controls

For more information about Form Thing, press the More button.
For information about other Things, press the Contents button.

Check the boxes for properties you want to enable. For more information, see Microsoft Visual Basic Help.

Click where you need help



For more information about Form Thing, press the More button.
For information about other Things, press the Contents button.

Choose File, New to clear the current form template from memory.

Choose File, Open to open a saved form template. Form Thing displays an Open Template dialog so that you can select a template to open. All property settings and the icon associated with the form are loaded into Form Thing.

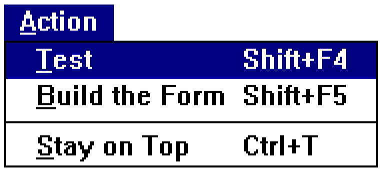
Choose File, Save to save the current prototype as a template for later reuse. Form Thing displays a Save Template dialog so that you can name the template.

Choose File, Save As to save the current template under a new name.

Choose File, Save Icon to File to save an icon stored with the current template out to an ICO file.

Choose File Save Major Settings as Default to make the current settings the default for Form Thing.

Click where you need help



For more information about Form Thing, press the More button.
For information about other Things, press the Contents button.

Stay on Top is a toggle command. Choosing Stay on Top toggles the option on or off. When it is on, a check mark appears in front of the words "Stay on Top."

If the option is on, the ToolThings window stays on top of other windows even when you switch to other applications. This can be useful when you are working simultaneously in a ToolThings application and a Visual Basic window.

To turn the option off, choose Stay on Top again. The check mark disappears and the window will no longer stay on top.

This tool puts an interface on Visual Basic's MsgBox function, making it easy for you to build Visual Basic message boxes on the fly. All options are represented visually, so construction of a message box is a simple matter of pointing at an option and clicking to select it.

A test mode displays a prototype of the message box, so you can tinker with settings and quickly see the results. When everything is how you want it, MsgBox Thing generates the code and copies it to the Windows clipboard.

You can tailor MsgBox Thing's coding style to match your own. For example, some programmers like to use string literals for message text, others prefer variables or expressions. MsgBox Thing can produce code in either style. It can adapt to your programming style in a variety of other ways as well.

Like other prototyping tools in the ToolThings toolset, MsgBox Thing lets you save templates for message boxes, making it possible to create house styles for common application messages.

For more information about MsgBox Thing, press the More button.
For information about other Things, press the Contents button.

Text

Buttons

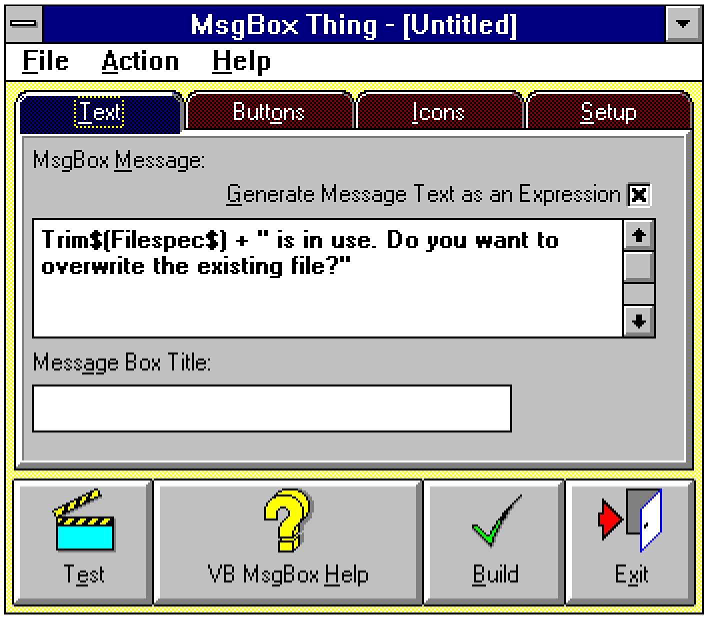
Icons

Setup

File

Action

Click where you need help



Generate Message Text as an Expression ☒

```
Trim$(Filespec$) + " is in use. Do you want to
overwrite the existing file?"
```

For more information about MsgBox Thing, press the More button.
For information about other Things, press the Contents button.

Check this box if you intend to type a valid Visual Basic expression in the text box below.

When Generate Message Text as an Expression is checked, whatever you type in the text box below will be passed unchanged as an argument to the Visual Basic function MsgBox().

When the box is cleared, the message text will be enclosed within quotation marks before it is passed.

Type either the literal text of the message or a valid expression.

If you type literal text, do not enclose it in quotation marks. Clear the box marked "Generate Message Text as an Expression."

If you type an expression, check the box marked "Generate Message Text as an Expression." If your expression includes a literal string, as the example here does, it must be enclosed within quotation marks.

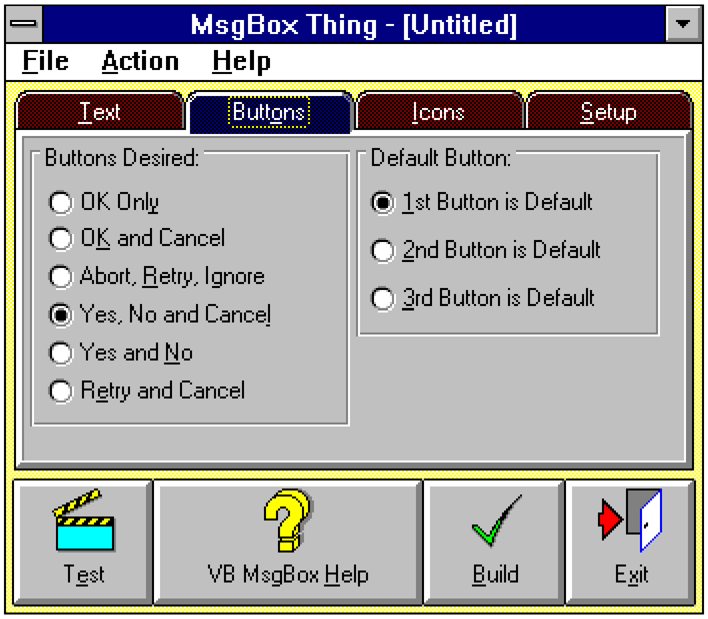
(Optional) Type the text you want to appear in the caption on the title bar. If you omit a title, the MsgBox function uses "Microsoft Visual Basic" as the default title for applications running in the Visual Basic development environment and the application name for executable files created by Visual Basic.

See a prototype of your message box.

Click this button to call up the Microsoft Visual Basic Help topic on message boxes.

Generate code for the message box. MsgBox Thing will place the code on the clipboard and ask if you want it pasted at the current insertion point in your VB code window.

Click where you need help



Buttons Desired:

- ○ OK Only
- ○ OK and Cancel
- ○ Abort, Retry, Ignore
- ● Yes, No and Cancel
- ○ Yes and No
- ○ Retry and Cancel

Default Button:

- ● 1st Button is Default
- ○ 2nd Button is Default
- ○ 3rd Button is Default

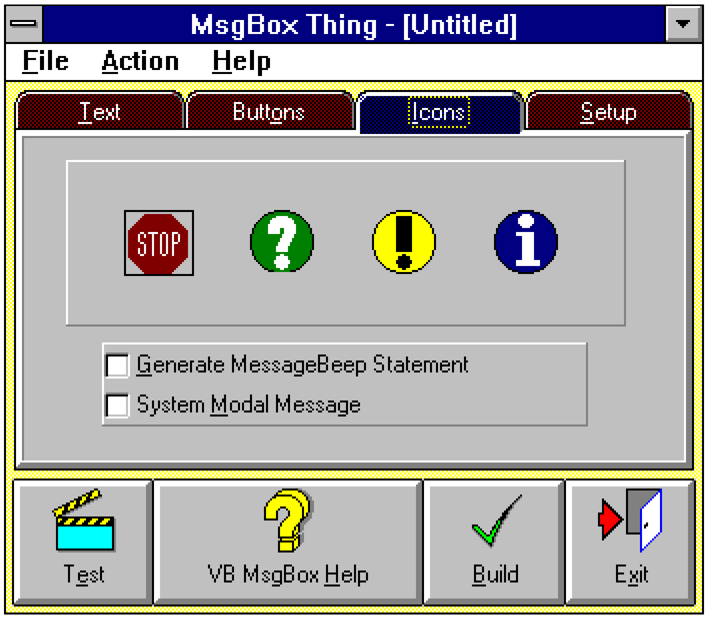
For more information about MsgBox Thing, press the More button.
For information about other Things, press the Contents button.

Select push buttons for your message box by clicking one of the radio buttons in the Buttons Desired group.

Choose which button you want to have focus when the message box displays.

For example, if the buttons in the message box are "Yes, No and Cancel" and you want the default button to be "No," click "2nd Button is Default."

Click where you need help



☐ Generate MessageBeep Statement

☐ System Modal Message

For more information about MsgBox Thing, press the More button.
For information about other Things, press the Contents button.

Click the picture of the icon you want to be displayed in the message box.

Check this box if you want to alert the user with a sound when the message box displays.

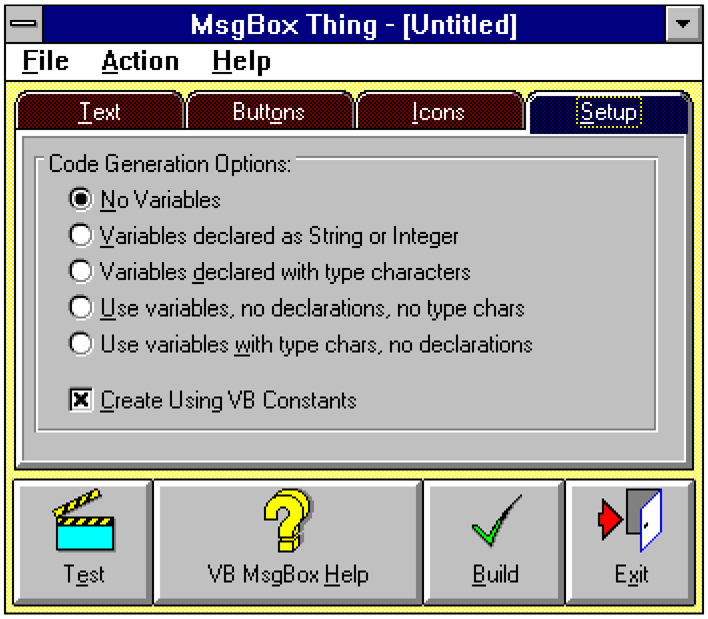
When you tell MsgBox Thing to generate the code for your message box, it will include a call to the Windows API function MessageBeep().

Note. MsgBox Thing does not automatically declare the function MessageBeep(). If you have not declared it, you must do so before running the project. From the Action menu, choose Generate MessageBeep Declaration. Code to declare the function will be placed on the clipboard. You can then paste the code in a declarations section of your project.

Check this box if you want to force your users to act on the message before switching to another application.

A system modal message suspends all activity on the system until the user responds. This is generally not considered good manners, so use the option wisely and infrequently.

Click where you need help



MsgBox Thing - [Untitled]

File Action Help

Text | Buttons | Icons | Setup

Code Generation Options:

- (●) No Variables
- ( ) Variables declared as String or Integer
- ( ) Variables declared with type characters
- ( ) Use variables, no declarations, no type chars
- ( ) Use variables with type chars, no declarations

[✓] Create Using VB Constants

For more information about MsgBox Thing, press the More button.
For information about other Things, press the Contents button.

Generated code does not use variables.

```
Select Case MsgBox("Do you want to save your changes?",
↳ MB_YESNOCANCEL +MB_DEFBUTTON1 + MB_ICONQUESTION +
      MB_APPLMODAL, "Application Name")
    Case IDCANCEL 'Cancel Selected
    Case IDYES 'Yes Selected
    Case IDNO 'No Selected
End Select
```

Generated code uses variables declared As String or As Integer.

```
Dim Msg As String, MsgTitle As String, Response As Integer
Msg = "Do you want to save your changes?"
MsgTitle = "Application Name"
Response = MsgBox(Msg, MB_YESNOCANCEL +
 MB_DEFBUTTON1 + MB_ICONQUESTION +
 MB_APPLMODAL, MsgTitle)
Select Case Response
    Case IDCANCEL 'Cancel Selected
    Case IDYES 'Yes Selected
    Case IDNO 'No Selected
End Select
```

Generated code uses variables declared with type characters.

```
Dim Msg$, MsgTitle$, Response%
Msg$ = "Do you want to save your changes?"
MsgTitle$ = "Application Name"
Response% = MsgBox(Msg$, MB_YESNOCANCEL +
      MB_DEFBUTTON1 + MB_ICONQUESTION +
      MB_APPLMODAL, MsgTitle$)
Select Case Response%
    Case IDCANCEL 'Cancel Selected
    Case IDYES 'Yes Selected
    Case IDNO 'No Selected
End Select
```

Generated code uses variables without type characters. Variables are not declared.

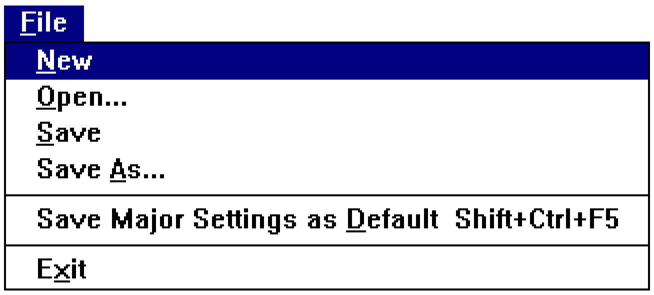
```
Msg = "Do you want to save your changes?"
MsgTitle = "Application Name"
Response = MsgBox(Msg, MB_YESNOCANCEL +
       MB_DEFBUTTON1 + MB_ICONQUESTION +
       MB_APPLMODAL, MsgTitle)
Select Case Response
    Case IDCANCEL 'Cancel Selected
    Case IDYES 'Yes Selected
    Case IDNO 'No Selected
End Select
```

Generated code uses variables with type characters. Variables are not declared.

```
Msg$ = "Do you want to save your changes?"
MsgTitle$ = "Application Name"
Response% = MsgBox(Msg$, MB_YESNOCANCEL +
      MB_DEFBUTTON1 + MB_ICONQUESTION +
      MB_APPLMODAL, MsgTitle$)
Select Case Response%
    Case IDCANCEL 'Cancel Selected
    Case IDYES 'Yes Selected
    Case IDNO 'No Selected
End Select
```

When this box is checked, generated code will use constants from CONSTANT.TXT. You must have declared these constants in your project.

Click where you need help



For more information about MsgBox Thing, press the More button.
For information about other Things, press the Contents button.

Choose File, New to clear the current form template from memory.

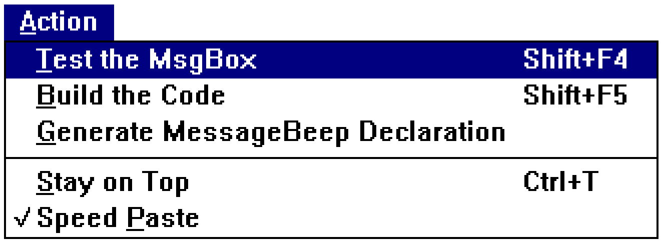
Choose File, Open to open a saved message box template. MsgBox Thing displays an Open Template dialog so that you can select a template to open. All settings saved with the template are loaded into MsgBoxThing.

Choose File, Save to save the current prototype as a template for later reuse. MsgBox Thing displays a Save Template dialog so that you can name the template.

Choose File, Save As to save the current template under a new name.

Choose File Save Major Settings as Default to make the current settings the default for MsgBox Thing.

Click where you need help



Action	
Test the MsgBox	Shift+F4
Build the Code	Shift+F5
Generate MessageBeep Declaration	
Stay on Top	Ctrl+T
✓ Speed Paste	

For more information about MsgBox Thing, press the More button.
For information about other Things, press the Contents button.

Speed Paste is a toggle command. Choosing Speed Paste toggles the option on or off. When it is on, a check mark appears in front of the words "Speed Paste."

Turning Speed Paste on can save time. Each time you click Build, generated code is automatically pasted at the current insertion point in the VB code window. ToolThings does not pause to ask if that's what you want to do.

Note. If the code window is not open, or if it is open to a declarations section, generated code is copied to the clipboard. No attempt is made to paste the code into your project.

To turn the option off, choose Speed Paste again. The check mark disappears and the next time you Build you will be prompted for permission to paste code at the insertion point.

Code to declare the Windows API function MessageBeep() will be placed on the clipboard. You can then paste the code in a declarations section of your project.

CMDialog Thing supplies the interactive design feature missing from Visual Basic's common dialog control. It lets you set dialog flags and properties in design mode and immediately see the result without writing a single line of code. When you have a dialog the way you want it, CMDialog Thing generates the code and copies it to the clipboard. All you need to do is insert the contents of the clipboard at the appropriate spot in your event procedure.

For more information about CMDialog Thing, press the More button.
For information about other Things, press the Contents button.

File

Color

Print

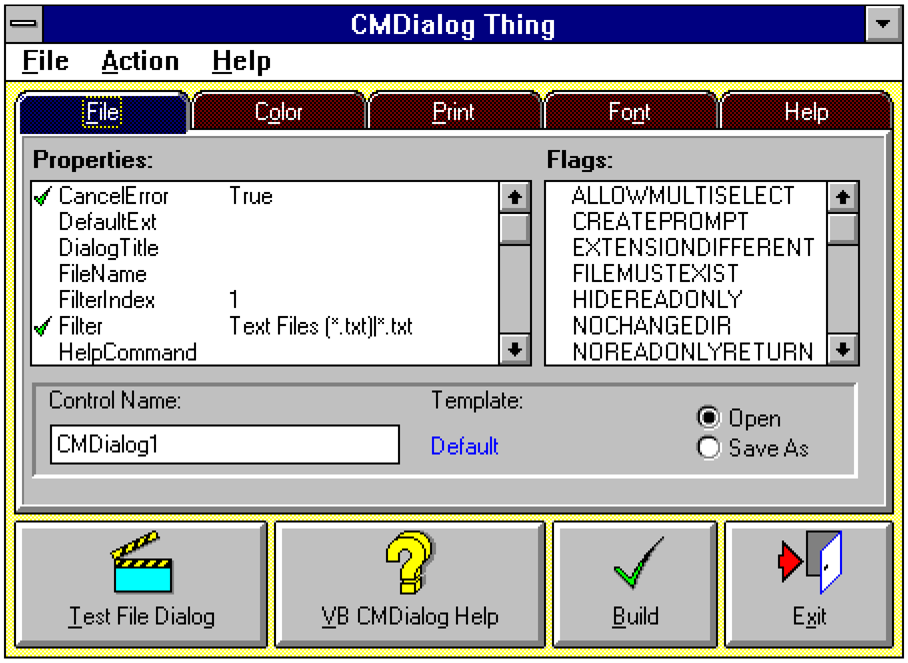
Font

Help

File

Action

Click where you need help



Properties:

✓	CancelError	True
	DefaultExt	
	DialogTitle	
	FileName	
	FilterIndex	1
✓	Filter	Text Files (*.txt)|*.txt
	HelpCommand	

Flags:

ALLOWMULTISELECT
CREATEPROMPT
EXTENSIONDIFFERENT
FILEMUSTEXIST
HIDEREADONLY
NOCHANGEDIR
NOREADONLYRETURN

Control Name: CMDialog1

Template: Default

[✓ Open] Save As

For more information about CMDialog Thing, press the More button.
For information about other Things, press the Contents button.

Double-click the name of the property you want to set. A property-setting window appears and you can then choose a setting.

A check mark in front of the name of a property indicates that the property is enabled. To enable a property, double-click the property name. When the property-setting window appears, check the Enable box.

Note. You can set a value without enabling a property.

Double-click the name of a flag to set it on. A check mark appears in front of the name, indicating that the flag is on. Double-clicking the name again sets the flag off.

See a prototype of your common dialog.

Click to see the Microsoft Visual Basic Help topic on the CMDialog control.

Build the common dialog. CMDialog Thing generates the code, places it on the clipboard, and asks if you want the code pasted at the current insertion point in your VB code window.

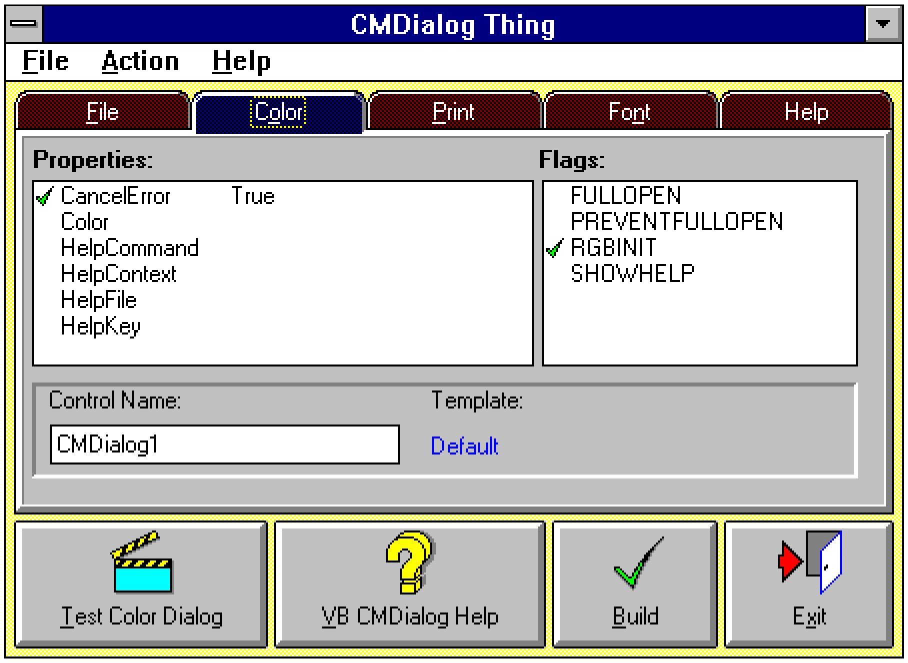
Type the name of the control.

Note. You must have dropped a copy of the common dialog control on your form.

Name of the currently loaded common dialog template.

To create an Open dialog, click the Open button. To create a Save As dialog, click the Save As button.

Click where you need help



Properties:

✓	CancelError	True
	Color	
	HelpCommand	
	HelpContext	
	HelpFile	
	HelpKey	

Flags:

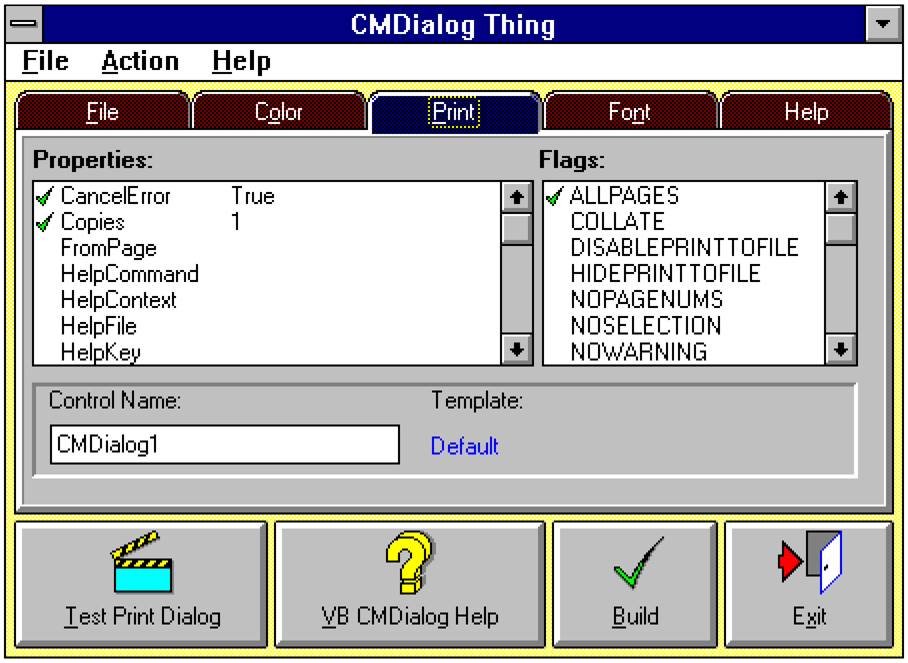
FULLOPEN
PREVENTFULLOPEN
✓ RGBINIT
SHOWHELP

Control Name: CMDialog1

Template: Default

For more information about CMDialog Thing, press the More button.
For information about other Things, press the Contents button.

Click where you need help



Properties:

✓	CancelError	True
✓	Copies	1
	FromPage	
	HelpCommand	
	HelpContext	
	HelpFile	
	HelpKey	

Flags:

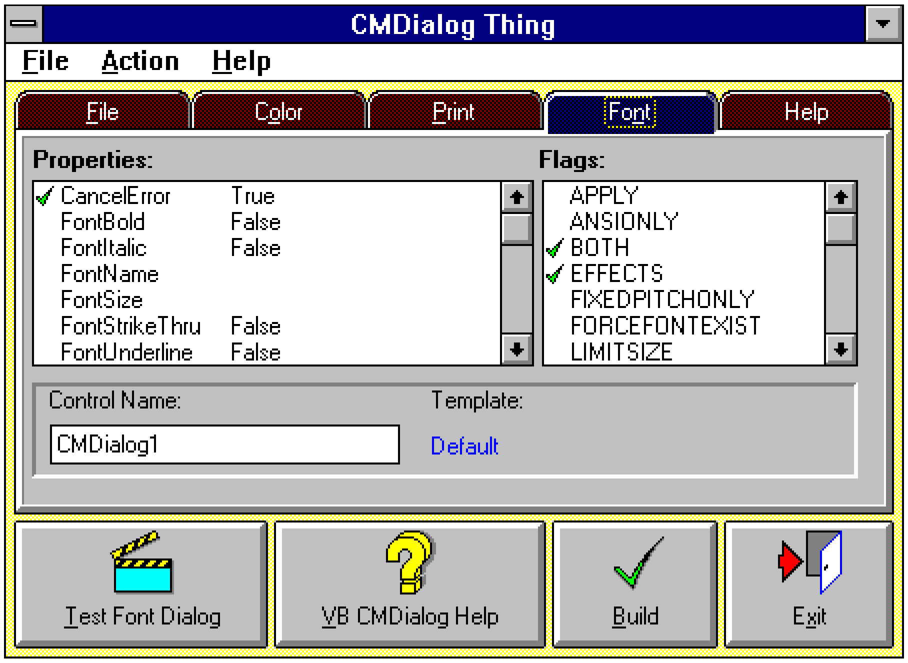
✓	ALLPAGES
	COLLATE
	DISABLEPRINTTOFILE
	HIDEPRINTTOFILE
	NOPAGENUMS
	NOSELECTION
	NOWARNING

Control Name: CMDialog1

Template: Default

For more information about CMDialog Thing, press the More button.
For information about other Things, press the Contents button.

Click where you need help


CMDialog Thing

File Action Help

File | Color | Print | Font | Help

Properties:

✓	CancelError	True
	FontBold	False
	FontItalic	False
	FontName	
	FontSize	
	FontStrikeThru	False
	FontUnderline	False

Flags:

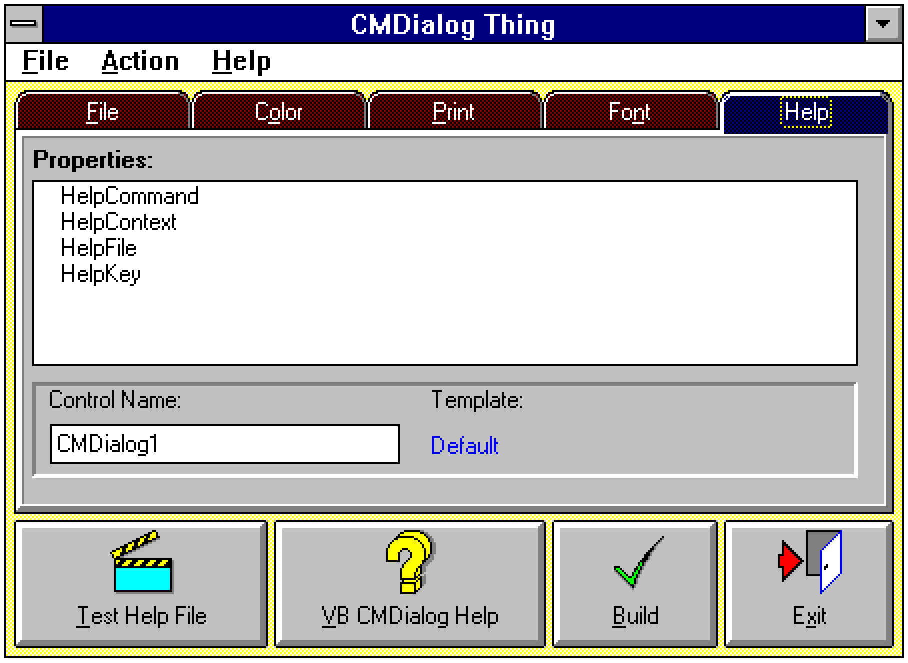
APPLY
ANSIONLY
✓ BOTH
✓ EFFECTS
FIXEDPITCHONLY
FORCEFONTEXIST
LIMITSIZE

Control Name: CMDialog1

Template: Default

For more information about CMDialog Thing, press the More button.
For information about other Things, press the Contents button.

Click where you need help



Properties:

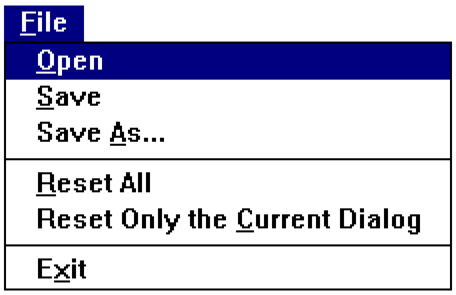
HelpCommand
HelpContext
HelpFile
HelpKey

Control Name: CMDialog1

Template: Default

For more information about CMDialog Thing, press the More button.
For information about other Things, press the Contents button.

Click where you need help



For more information about CMDialog Thing, press the More button.
For information about other Things, press the Contents button.

Choose File, Open to open a saved common dialog template. CMDialog Thing displays an Open Template dialog so that you can select a template to open.

Note. The Open Template dialog lists only templates for the type of common dialog associated with the current tab. For example, if you are at the Font tab, you will see only templates for font dialogs. At the Print tab, you will see only templates for print dialogs. And so on.

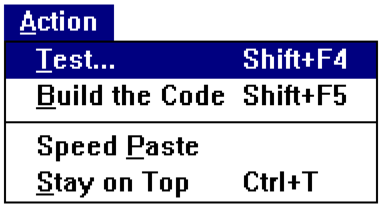
Choose File, Save to save the prototype at the current tab as a template for later reuse. CMDialog Thing displays a Save Template dialog so that you can name the template.

Choose File, Save As to save the current template under a new name.

Choose File, Reset All to reset all tabs to the factory defaults.

Choose File, Reset Only the Current Dialog to reset the current dialog to the factory defaults.

Click where you need help



For more information about CMDialog Thing, press the More button.
For information about other Things, press the Contents button.

Type Thing is the tool to use when you are working with user-defined data types. It can keep a library of type structures and insert a declaration in your code. It can scan your current project, store any type structures that it finds, and then help you make assignments to elements of an instance of a declared structure.

You can also use Type Thing to declare constants and global variables, or to make assignments using constants or global variables.

Type Thing's Auto-Str$(), Auto-Trim$(), and Auto-Val() options simplify the process of making assignments. Once the necessary options have been selected, Type Thing generates code and pastes it to the clipboard or, optionally, directly into the current routine. A test button lets you view the code before it is copied.

For more information about Type Thing, press the More button.
For information about other Things, press the Contents button.

Types

Constants

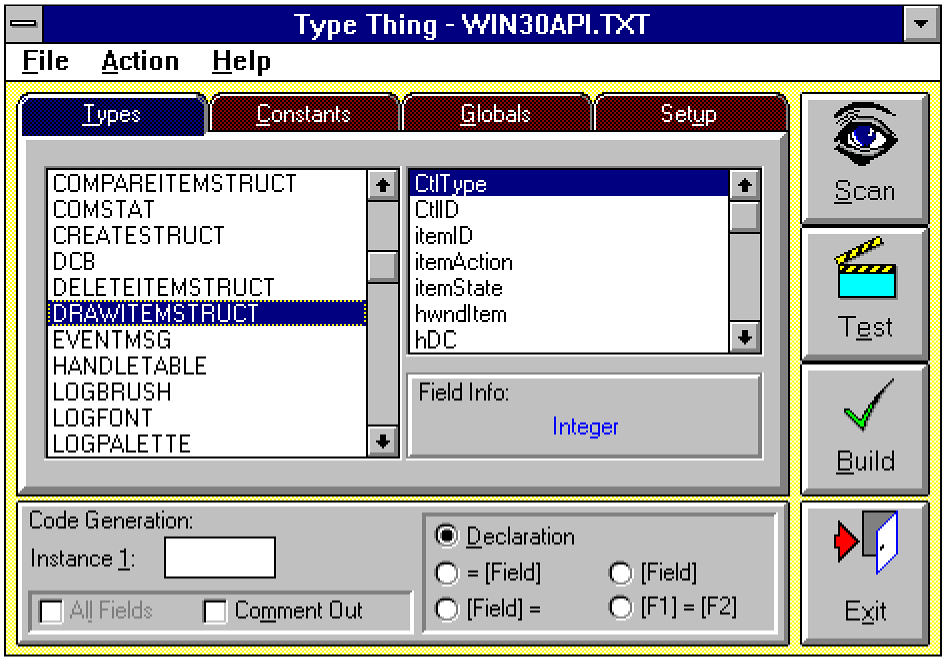
Globals

Setup

File

Action

Click where you need help



COMPAREITEMSTRUCT
COMSTAT
CREATESTRUCT
DCB
DELETEITEMSTRUCT
DRAWITEMSTRUCT
EVENTMSG
HANDLETABLE
LOGBRUSH
LOGFONT
LOGPALETTE

CtlType
CtlID
itemID
itemAction
itemState
hwndItem
hDC

Field Info: Integer

Code Generation:
Instance 1:

[ ] All Fields [ ] Comment Out

[✓ Declaration]
= [Field]
[Field]
[Field] =
[F1] = [F2]

For more information about Type Thing, press the More button.
For information about other Things, press the Contents button.

Select the type structure you want to work with by clicking on its name.

Tip. You can move quickly to an item in a long list by typing the first few characters of its name. To clear the search buffer and start over, press the Escape key.

The elements of the selected type structure are displayed here. If you want to work with a particular element, select it by clicking on its name. If you want to work with several elements, multi-select them by holding down the Ctrl key while you click.

The data type of the selected element is displayed here.

Type the name of this instance of the selected type.

Check this box if you want to make assignments for all elements of the type structure.

Check this box if you want code to be commented out automatically when it is generated.

Click this button if you want to create a declaration for the selected item.

Click this button if you want Type Thing to create code in the form

<operator> instance.field

where

operator	is a relational operator selected at Type Thing's Setup tab.
instance	is the name of entered in the Instance 1 box.
field	is a currently selected element.

For example, if the current operator is "=", the current instance of DRAWITEMSTRUCT is *rc*, and the currently selected fields are itemAction and itemState, the code generated by Type Thing will be:

```
= rc.itemAction
= rc.itemState
```

Click this button if you want Type Thing to create code in the form

> *instance.field <operator>*

where

instance	is the name of entered in the Instance 1 box.
field	is a currently selected element.
operator	is a relational operator selected at Type Thing's Setup tab.

For example, if the current instance of DRAWITEMSTRUCT is *rc*, the currently selected fields are itemAction and itemState, and the current operator is "=", the code generated by Type Thing will be:

```
rc.itemAction =
rc.itemState =
```

Click this button if you want Type Thing to create code in the form

<operator> instance.field

where

instance	is the name of entered in the Instance 1 box.
field	is a currently selected element.

For example, if the current instance of DRAWITEMSTRUCT is *rc*, and the currently selected fields are itemAction and itemState, the code generated by Type Thing will be:

```
rc.itemAction
rc.itemState
```

Click this button if you want Type Thing to create code in the form

<operator> instance1.field = <operator> instance2.field

where

operator	is a relational operator selected at Type Thing's Setup tab.
instance1	is the name of entered in the Instance 1 box.
instance2	is the name of entered in the Instance 2 box.
field	is a currently selected element.

For example, if the current operator is "=", the instances of DRAWITEMSTRUCT are *rc1* and *rc2*, and the currently selected fields are itemAction and itemState, the code generated by Type Thing will be:

```
rc1.itemAction = rc2.itemAction
rc1.itemState = rc2.itemState
```

Click this button to scan the current project.

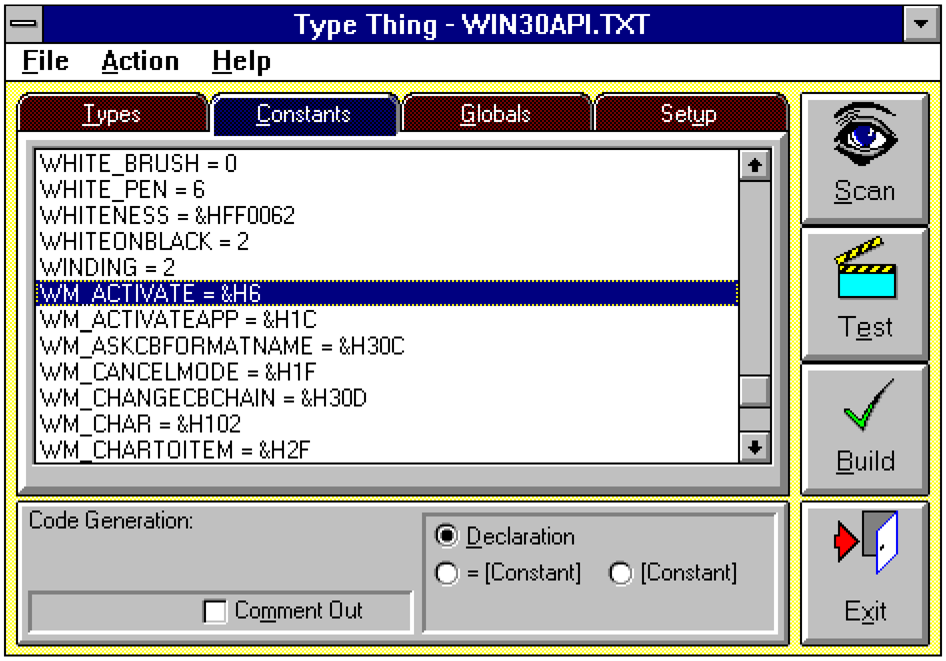
Type Thing searches the project for type structures, global variables, and constants. While the search is underway, a status bar displays its progress. When the scan is complete, the name of the project displays on the title bar and Type Thing's lists are populated with the results of the search.

Preview the generated code.

You can edit the code in the preview window and then copy it to the clipboard.

Build the code. Type Thing will copy the code to the clipboard and then ask if you want it pasted at the current insertion point in your VB code window.

Click where you need help



```
WHITE_BRUSH = 0
WHITE_PEN = 6
WHITENESS = &HFF0062
WHITEONBLACK = 2
WINDING = 2
WM_ACTIVATE = &H6
WM_ACTIVATEAPP = &H1C
WM_ASKCBFORMATNAME = &H30C
WM_CANCELMODE = &H1F
WM_CHANGECBCHAIN = &H30D
WM_CHAR = &H102
WM_CHARTOITEM = &H2F
```

Code Generation:

[✓ Declaration] = [Constant] [Constant]

☐ Comment Out

For more information about Type Thing, press the More button.
For information about other Things, press the Contents button.

Select the constant you want to work with by clicking on its name.

Tip. You can move quickly to an item in a long list by typing the first few characters of its name. To clear the search buffer and start over, press the Escape key.

Click this button if you want Type Thing to create code in the form

<*operator*> *constant*

where

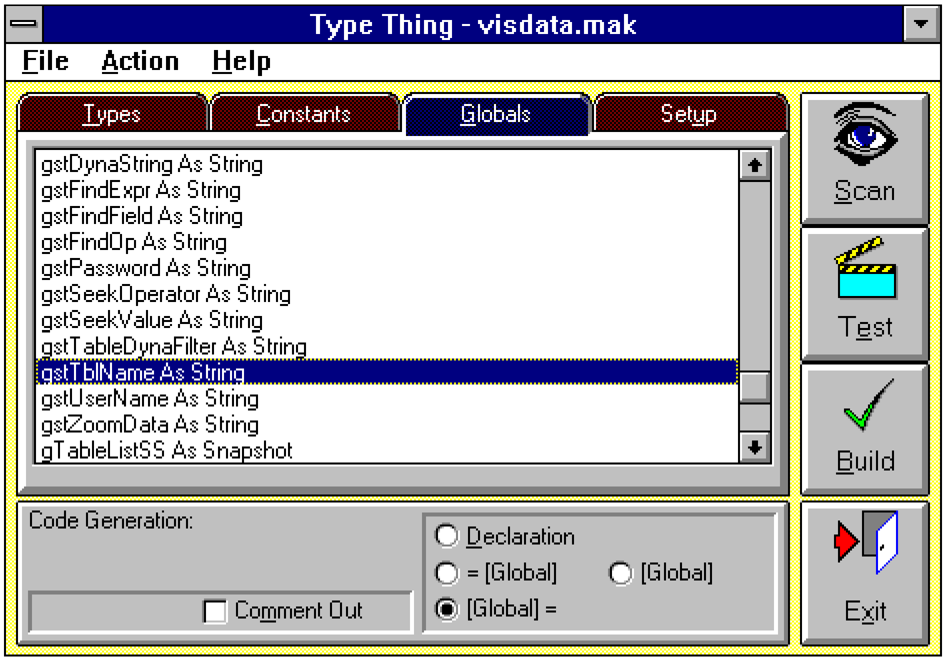
operator	is a relational operator selected at Type Thing's Setup tab.
constant	is a currently selected constant.

For example, if the current operator is "=" and the currently selected constant is WM_ACTIVATE, the code generated by Type Thing will be:

```
= WM_ACTIVATE
```

Click this button if you want Type Thing to simply list the selected constants when it generates code.

Click where you need help



gstDynaString As String
gstFindExpr As String
gstFindField As String
gstFindOp As String
gstPassword As String
gstSeekOperator As String
gstSeekValue As String
gstTableDynaFilter As String
gstTblName As String
gstUserName As String
gstZoomData As String
gTableListSS As Snapshot

Code Generation:

☐ Comment Out

○ Declaration
○ = [Global]
○ [Global]
● [Global] =

For more information about Type Thing, press the More button.
For information about other Things, press the Contents button.

Select the global variable you want to work with by clicking on its name.

Tip. You can move quickly to an item in a long list by typing the first few characters of its name. To clear the search buffer and start over, press the Escape key.

Click this button if you want Type Thing to create code in the form

<operator> global

where

operator	is a relational operator selected at Type Thing's Setup tab.
global	is a currently selected global variable.

For example, if the current operator is "=" and the currently selected variables are LastX1 and LastY1, the code generated by Type Thing will be:

```
= LastX1
= LastY1
```

Click this button if you want Type Thing to create code in the form

global <operator>

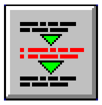
where

global	is a currently selected global variable.
operator	is a relational operator selected at Type Thing's Setup tab.

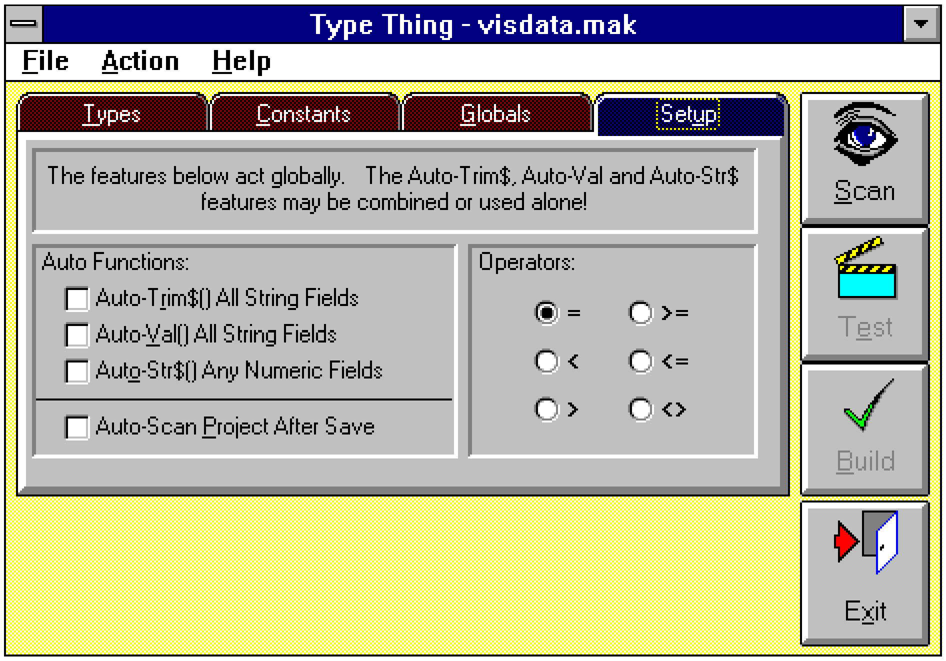
For example, if the current operator is "=" and the currently selected variables are LastX1 and LastY1, the code generated by Type Thing will be:

```
LastX1=
LastY1=
```

Click this button if you want Type Thing to simply list the selected variables when it generates code.



Click where you need help



Type Thing - visdata.mak

File Action Help

Types | Constants | Globals | Setup

The features below act globally. The Auto-Trim$, Auto-Val and Auto-Str$ features may be combined or used alone!

Auto Functions:

- [ ] Auto-Trim$() All String Fields
- [ ] Auto-Val() All String Fields
- [ ] Auto-Str$() Any Numeric Fields
- [ ] Auto-Scan Project After Save

Operators:

[✓ =] >= < <= > <>

For more information about Type Thing, press the More button.
For information about other Things, press the Contents button.

Select the relational operator you want to use in expressions built by Type Thing.

Enable this option if you want generated code to pass the value of any string field to the Trim$() function:

```
Trim$(Icon.FileName)
```

Enable this option if you want generated code to pass the value of any string field to the Val() function:

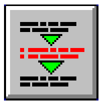
```
Val(Bitmap.bmPlanes)
```

Enable this option if you want generated code to pass the value of any numeric field to the Str$() function:

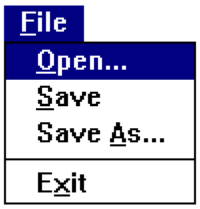
```
Str$(Icon.Changed)
```

When this option is enabled, Type Thing will detect a Save Project command in Visual Basic and, after a slight pause, automatically rescan the project.

Note. You must have scanned the project manually at least once.



Click where you need help



For more information about Type Thing, press the More button.
For information about other Things, press the Contents button.

Choose File, Open to open an external BAS, MAK, or TXT file, or a saved Type Thing Library (TBF) file. Type Thing automatically scans the external file for type structures, constants, and global variables.

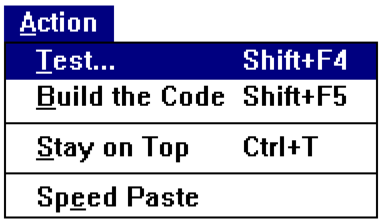
Note. Type Thing can scan any ASCII file. It cannot scan a Visual Basic file saved in binary format.

Choose File, Save to save the current type structures, constants, and global variables as a Type Thing Library (TBF) file. The next time you want to use some of these declarations, just open the library from Type Thing.

Tip. Type Thing Libary files are simple ASCII text files. You can combine files by appending one to the end of another or by using a text editor to cut and paste. However, the size of a library file is limited to 3000 items. If you combine libraries, take care not to exceed this limit.

Choose File, Save As to save the current Type Thing Library under a new name or in a new location.

Click where you need help



For more information about Type Thing, press the More button.
For information about other Things, press the Contents button.

Call Thing is a tool designed to help you build calls. It can come in handy at that point in a project when you find yourself having to check the syntax, return type, or the name of a routine before making a call.

Just tell Call Thing to scan your project and memorize all of its subroutines and functions. Then keep it minimized on your desktop. When you want to insert a call, pop up Call Thing and just point and click. Or, even better, stay right where you are, type a the first few characters of the routine's name and press Alt + C. Call Thing will complete the call for you.

Call Thing can also handle declarations. If you are using someone else's DLLs, scan their declaration file. Or recycle your own work by scanning a project that has already has the proper declarations. Whatever the source of your information, once it is in Call Thing's memory it becomes infinitely more manageable. When you decide to use an external function, let Call Thing hunt down the declaration and insert it in your code.

For more information about Call Thing, press the More button.
For information about other Things, press the Contents button.

DLLs

Subs

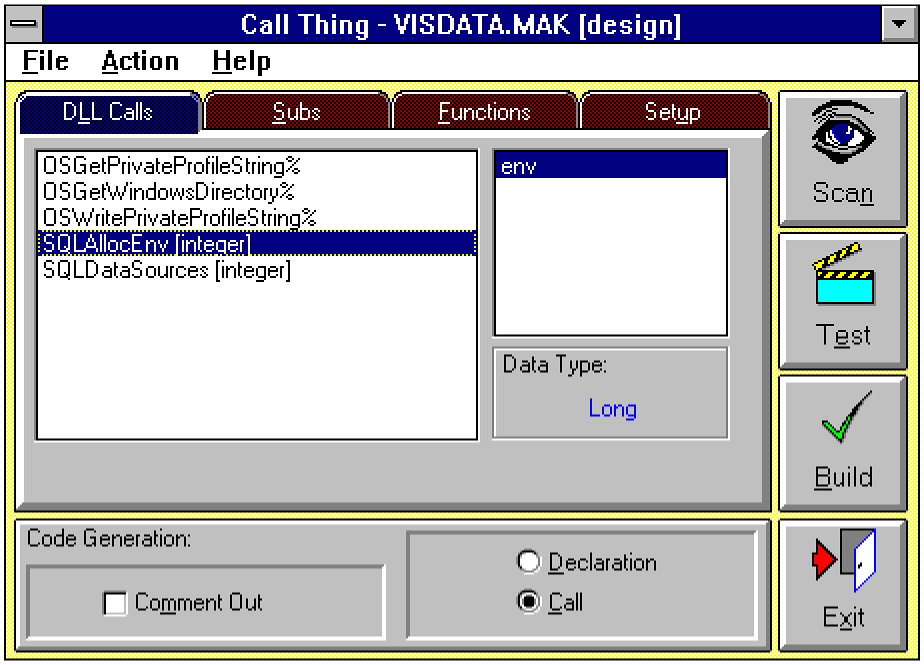
Functions

Setup

File

Action

Click where you need help



OSGetPrivateProfileString%
OSGetWindowsDirectory%
OSWritePrivateProfileString%
SQLAllocEnv [integer]
SQLDataSources [integer]

env

Data Type:
Long

[ ] Comment Out

( ) Declaration
(●) Call

For more information about Call Thing, press the More button.
For information about other Things, press the Contents button.

Select the procedure you want to work with by clicking on its name.

If the routine you have selected is a function, the data type of the return value is indicated by a type identifier. Or, if the function was declared with explicit typing (i.e., with an As statement), the return type is displayed in brackets.

Tip. You can move quickly to an item in a long list by typing the first few characters of its name. To clear the search buffer and start over, press the Escape key.

Parameters, if any, for the currently selected routine.

Data type of the currently selected parameter.

Check this box if you want code to be commented out automatically when it is generated.

Click this button if you want to create a declaration for the selected routine.

Click this button if you want to create a call to the selected routine.

Click this button to scan the current project.

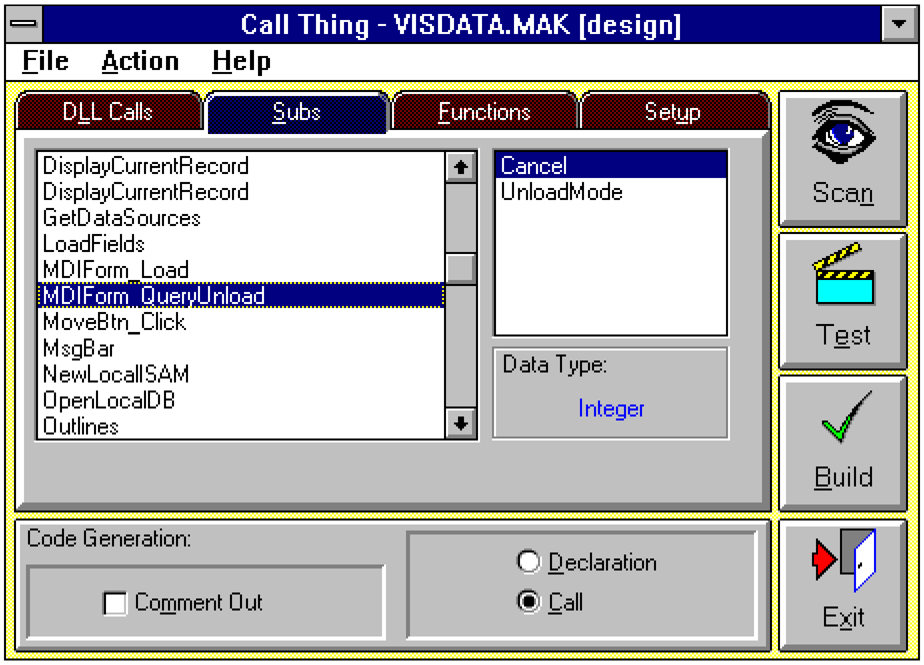
Call Thing searches the project for DLL procedures, subroutines, and functions. While the search is underway, a status bar displays its progress. When the scan is complete, the name of the project displays on the title bar and Call Thing's lists are populated with the results of the search.

Preview the generated code.

You can edit the code in the preview window and then copy it to the clipboard.

Build the code. Call Thing will copy the code to the clipboard and then ask if you want it pasted at the current insertion point in your VB code window.

Click where you need help


Call Thing - VISDATA.MAK [design]

File Action Help

DLL Calls | Subs | Functions | Setup

DisplayCurrentRecord
DisplayCurrentRecord
GetDataSources
LoadFields
MDIForm_Load
MDIForm_QueryUnload
MoveBtn_Click
MsgBar
NewLocalSAM
OpenLocalDB
Outlines

Cancel
UnloadMode

Data Type:
Integer

Code Generation:
Comment Out
Declaration
[✓ Call]

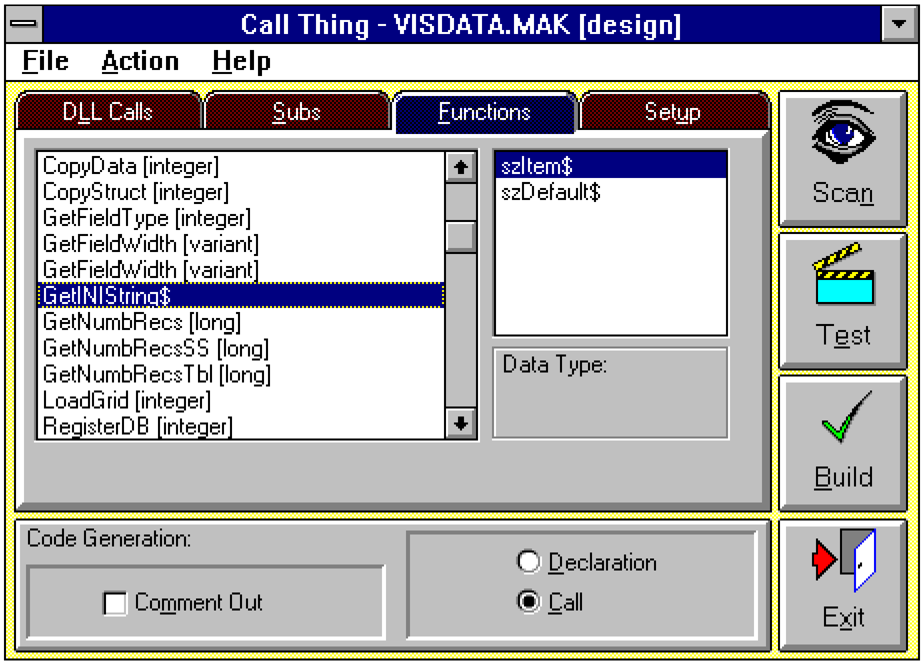
For more information about Call Thing, press the More button.
For information about other Things, press the Contents button.

Select the procedure you want to work with by clicking on its name.

Tip. You can move quickly to an item in a long list by typing the first few characters of its name. To clear the search buffer and start over, press the Escape key.

Parameters, if any, for the currently selected routine.

Click where you need help



CopyData [integer]
CopyStruct [integer]
GetFieldType [integer]
GetFieldWidth [variant]
GetFieldWidth [variant]
GetINIString$
GetNumbRecs [long]
GetNumbRecsSS [long]
GetNumbRecsTbl [long]
LoadGrid [integer]
RegisterDB [integer]

szItem$
szDefault$

Data Type:

Code Generation:

☐ Comment Out

○ Declaration
● Call

For more information about Call Thing, press the More button.
For information about other Things, press the Contents button.

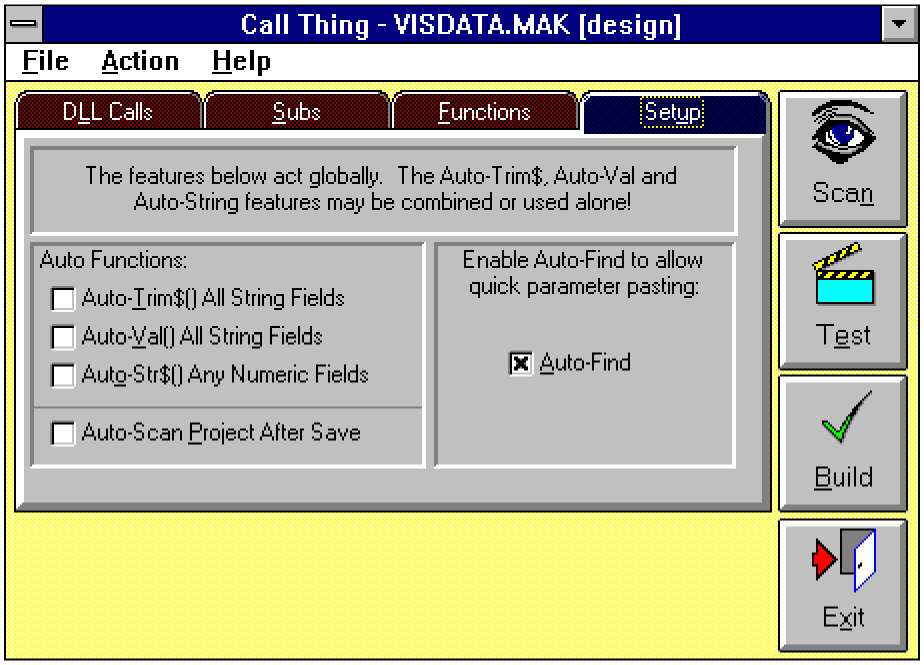
Select the procedure you want to work with by clicking on its name.

The data type of the function's return value is indicated by a type identifier. Or, if the function was declared with explicit typing (i.e., with an As statement), the return type is displayed in brackets.

Tip. You can move quickly to an item in a long list by typing the first few characters of its name. To clear the search buffer and start over, press the Escape key.

Parameters, if any, for the currently selected routine.

Click where you need help



The features below act globally. The Auto-Trim$, Auto-Val and Auto-String features may be combined or used alone!

Auto Functions:

- [ ] Auto-Trim$() All String Fields
- [ ] Auto-Val() All String Fields
- [ ] Auto-Str$() Any Numeric Fields
- [ ] Auto-Scan Project After Save

Enable Auto-Find to allow quick parameter pasting:

- [✓] Auto-Find

For more information about Call Thing, press the More button.
For information about other Things, press the Contents button.

Enable this option if you want generated code to pass the return value of any function that returns a string to the Trim$() function:

```
Trim$(StripFileName$())
```

Enable this option if you want generated code to pass the return value of any function that returns a string to the Val() function:

```
Val(StripFileName$())
```

Enable this option if you want generated code to pass the return value of any function that returns a numeric to the Str$() function:

```
Str$(Validate_FileSpec(AForm, MustExist))
```

When this option is enabled, Call Thing will detect a Save Project command in Visual Basic and, after a slight pause, automatically rescan the project.

Note. You must have scanned the project manually at least once.

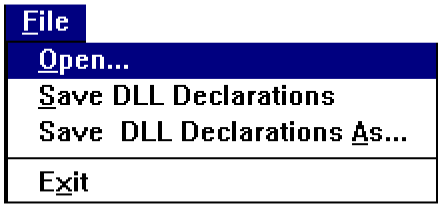
Enable this option if you want to be able to insert declarations and calls without leaving your VB code window.

When Auto-Find is enabled, you can insert a declaration or call from within the Visual Basic editor by typing the first few characters of the routine's name and then pressing Alt + C.

Note. You must type enough of the routine's name to distinguish it from other routines in Call Thing's scanned lists.

If you are in a declarations section, Call Thing will insert a declaration for the routine. If you are in a routine, Call Thing will insert a call.

Click where you need help



For more information about Call Thing, press the More button.
For information about other Things, press the Contents button.

Choose File, Open to open and scan an external BAS, MAK, or TXT file, or a Call Thing Library (CTF) file.

Call Thing scans the file automatically. When scanning is complete, the name of the scanned file appears on the title bar and the DLL Calls, Subs, and Functions tabs list the routines found in the file.

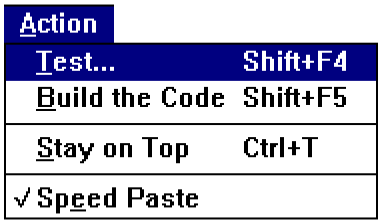
Note. Call Thing can scan any ASCII file. It cannot scan a Visual Basic file saved in binary format.

Choose File, Save DLL Declarations to save the current list of declared DLL functions as a Call Thing Library (CTF) file. The next time you want to use some of these declarations, just open the library from Call Thing.

Tip. Call Thing Libary files are simple ASCII text files. You can combine files by appending one to the end of another or by using a text editor to cut and paste. However, the size of a library file is limited to 3000 items. If you combine libraries, take care not to exceed this limit.

Choose File, Save DLL Declarations As to save the current Call Thing Library under a new name or in a new location.

Click where you need help



For more information about Call Thing, press the More button.
For information about other Things, press the Contents button.

Have you ever tried to lay out a series of buttons or check boxes on a crowded form? Nudge is the perfect tool for this job. It can move, grow, or shrink controls in increments of one or more pixels. Controls can be "nudged" diagonally and in any of the four polar directions.

Nudge also positions, aligns, and sizes groups of controls. What's especially handy is that you can just drop a number of controls on your VB form, size one of them the way you want them all to look, and then tell Nudge to make the others the same size as the first. This tool is also good for getting labels to line up with text boxes. You line one label up with its text box and Nudge will do the same for all the rest.

The positions of controls can be locked so a mouse touch does not inadvertently disturb the layout. All actions can be undone or redone-stepping backward and forward through every change made during the current session.

For more information about Nudge Thing, press the More button.
For information about other Things, press the Contents button.

Nudge - Nudging

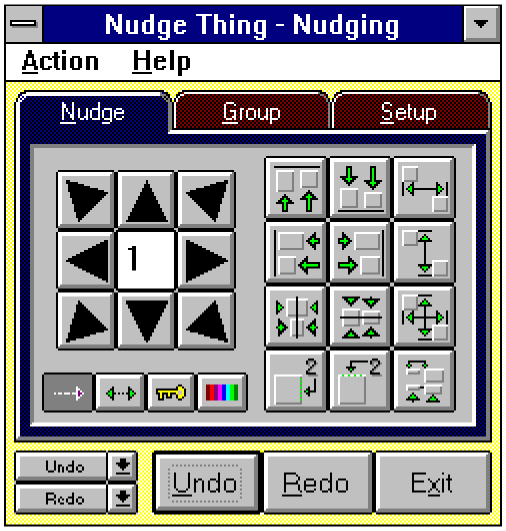
Nudge - Stretching

Group

Setup

Action

Click where you need help



For more information about Nudge Thing, press the More button.
For information about other Things, press the Contents button.

Directional arrows used in nudging and stretching.

You can also use the arrow keys on your keyboard. While focus is on Nudge Thing, pressing the Right, Left, Up, and Down cursor keys has the same effect as clicking on corresponding arrow buttons. While focus is on your form, pressing Shift + *cursor key* has the same effect.

Tip. If nudging the controls does not move them fast enough, you can shove them around the form in increments of ten pixels at a time by holding down the Ctrl key while clicking the arrow buttons.

Number of pixels that each nudge will move the selected controls or each stretch will grow or shrink them.

When this button is depressed, Nudge Thing is in Nudge mode. Clicking a directional arrow button moves the selected controls in the direction indicated by the arrow.

Movements are in increments of pixels. How many pixels depends on what number you type in the box at the center of the arrow buttons.

Undo the most recent action.

Redo the action most recently undone.

Click to undo multiple actions.

A list box appears with a list of all actions Nudge has performed during the current sesssion. Actions are listed in reverse order with the most recent at the top of the list. Drag the mouse down the list to select the actions you want to undo.

Note. Nudge keeps a separate undo list for each form you work on during a session.

Click to redo multiple actions.

A list box appears with a list of all actions Nudge has undone. Actions are listed in reverse order with the most recent at the top of the list. Drag the mouse down the list to select the actions you want to redo.

Note. Nudge keeps a separate redo list for each form you work on during a session.

Lock button. When this button is down, you cannot use the mouse to move controls on your form. You can move them only with Nudge Thing.

Note. The lock applies to new controls as well. You can add a new control while the control lock is on, but you cannot move it once you've dropped it on the form.

To re-enable manual movement of controls, release the control lock button by clicking it again. The button moves to the up position, indicating that the form is unlocked.

Click to set colors for the selected controls.

Align the tops of selected controls with the top of a reference control.

Align the bottoms of selected controls with the bottom of a reference control.

Align the left sides of selected controls with the left side of a reference control.

Align right sides of selected controls with the right side of a reference control.

Align the vertical axes of a column of selected controls with the vertical axis of a reference control.

Align the horizontal axes of a row of selected controls with the horizontal axis of a reference control.

Match the relative positions and spacing of controls in selected pairs with the relative positions and spacing of two controls in a reference pair.

Make selected controls the same *width* as a reference control.

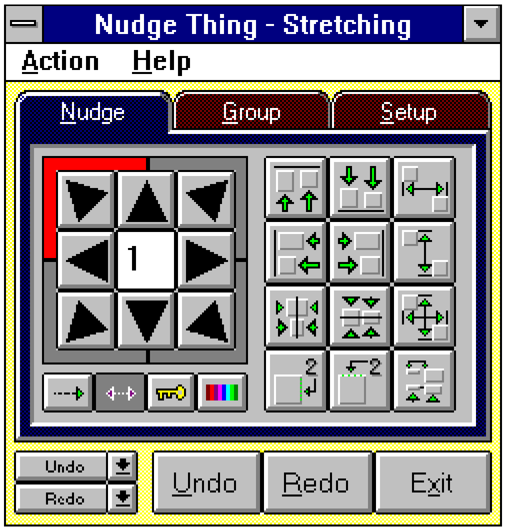
Make selected controls the same *height* as a reference control.

Make selected controls the same *width and height* as a reference control.

Square selected controls using the *height* of the controls as the measurement for all four sides.

Square selected controls using the *width* of the controls as the measurement for all four sides.

Click where you need help



For more information about Nudge Thing, press the More button.
For information about other Things, press the Contents button.

When this button is depressed, Nudge Thing is in Stretch mode. Clicking a directional arrow button stretches or shrinks the selected controls in the direction indicated by the arrow.

The highlighted corner of the frame around the directional arrows indicates which sides of the controls are free to move. Clicking an arrow button moves the sides in or out, shrinking or expanding the selected controls.

Changes are in increments of pixels. How many pixels depends on what number you type in the box at the center of the arrow buttons.

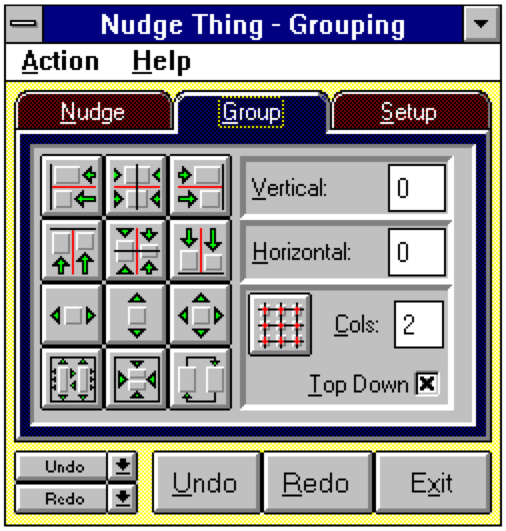
The highlighted corner of the frame around the directional arrows indicates which sides of the controls are free to move. Clicking an arrow button moves the sides in or out, shrinking or expanding the selected controls.

Click here to make the top and right sides of the selected controls free to move.

Click here to make the bottom and right sides of the selected controls free to move.

Click here to make the bottom and left sides of the selected controls free to move.

Click where you need help



For more information about Nudge Thing, press the More button.
For information about other Things, press the Contents button.

Specify how much vertical space, in pixels, you want between controls when you choose a grouping action.

Specify how much horizontal space, in pixels, you want between controls when you choose a grouping action.

Type the number of columns you want controls arranged in when you click the Columns, Shrink to Fit, or Expand to Fit buttons.

Click to arrange the selected controls in columns.

When Nudge Thing organizes columns, it places controls into columns in the order in which you have selected them. How it fills out the columns depends on whether the Top Down box is checked.

If Top Down is **unchecked**, Nudge Thing fills columns from left to right, in row order. When the number of controls is not evenly divisible by the number of columns, blank spaces are left in the bottom row.

If Top Down is **checked**, Nudge Thing fills columns from top down, in column order. When the number of controls is not evenly divisible by the number of columns, blank spaces are left in the last column.

Group and space selected controls *vertically* and align them with the *left* edge of a reference control.

Group and space selected controls *vertically* and align them on the *vertical axis* of a reference control.

Group and space selected controls *vertically* and align them with the *right* edge of a reference control.

Group and space selected controls *horizontally* and align them with the *top* edge of a reference control.

Group and space selected controls *horizontally* and align them on the *horizontal axis* of a reference control.

Group and space selected controls *horizontally* and align them with the *bottom* edge of a reference control.

Swap the positions of two selected controls.

Center the selected controls *horizontally* on the form or within their container.

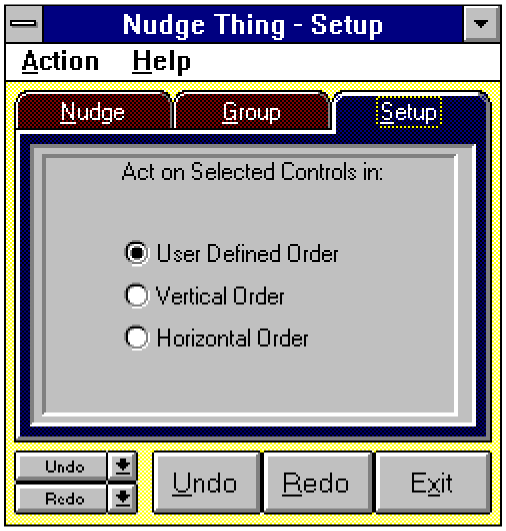
Center the selected controls *vertically* on the form or within their container.

Center the selected controls *horizontally* and *vertically* on the form or within their container.

Expand all controls within the selected container so that they completely fill the container.

Shrink the container to fit the controls within it.

Click where you need help



Act on Selected Controls in:

[● User Defined Order]
○ Vertical Order
○ Horizontal Order

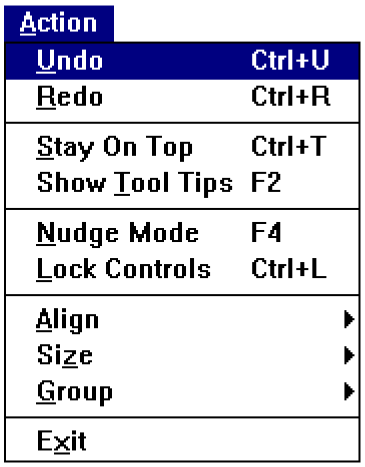
For more information about Nudge Thing, press the More button.
For information about other Things, press the Contents button.

(Default) Click if you want the first control you select to be the reference control.

Click if you want the control that is highest on the form to be the reference control.

Click if you want the control the is farthest to the left to be the reference control.

Click where you need help



For more information about Nudge Thing, press the More button.
For information about other Things, press the Contents button.

Show Tool Tips is a toggle command. Choosing Show Tool Tips toggles the option on or off. When the option is on, a check mark appears in front of the words "Show Tool Tips."

If the option is on, a balloon caption appears whenever the mouse cursor pauses over an unlabeled button.

Nudge Mode is a toggle command. Choosing the option toggles between Nudge Mode and Stretch Mode. The words displayed for the menu item change each time you choose the item.

- When you choose Nudge Mode, clicking the arrow buttons moves the selected controls.
- When you choose Stretch Mode, clicking the arrow buttons shrinks or expands the selected controls.

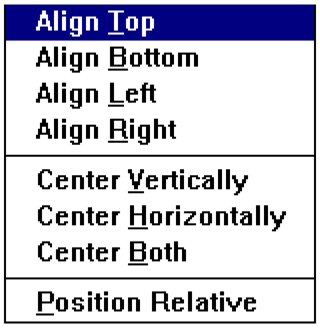
Lock Controls is a toggle command. Choosing Lock Controls toggles the option on or off. When the option is on, a check mark appears in front of the words "Lock Controls."

When the option is on, you cannot use the mouse to move controls on your form. You can move them only with Nudge Thing.

Note. The lock applies to new controls as well. You can add a new control while the control lock is on, but you cannot move it once you've dropped it on the form.

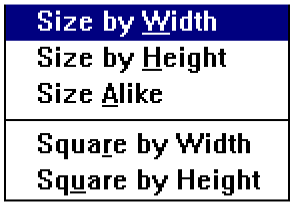
To re-enable manual movement of controls, release the lock by choosing Control Lock again.

Click where you need help



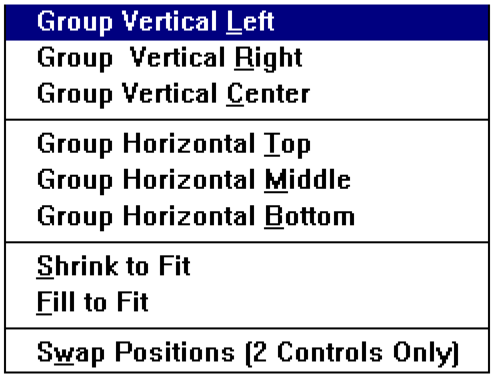
For more information about Nudge Thing, press the More button.
For information about other Things, press the Contents button.

Click where you need help



For more information about Nudge Thing, press the More button.
For information about other Things, press the Contents button.

Click where you need help



For more information about Nudge Thing, press the More button.
For information about other Things, press the Contents button.

Tab Thing can save programming time by letting you set the tab order for all controls on a form, or all controls within a frame or panel, in one quick shot. You can specify a starting tab number for controls within a group. If a control has a TabStop property, you can toggle the tab stop on or off.

Tab Thing offers two methods for setting tab order. You can choose the point-and-click method, and order tab indexes by clicking on controls. Or you can use the list box method, where you set tab order by dragging items up or down in a list.

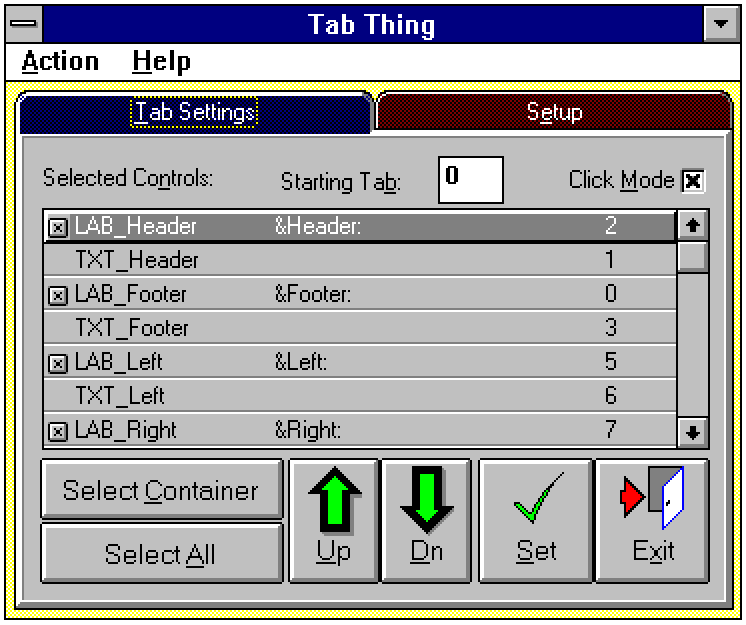
For more information about Tab Thing, press the More button.
For information about other Things, press the Contents button.

Tab Settings

Setup

Action

Click where you need help



Selected Controls:

Starting Tab: 0

Click Mode ☒

☒	LAB_Header	&Header:	2
	TXT_Header		1
☒	LAB_Footer	&Footer:	0
	TXT_Footer		3
☒	LAB_Left	&Left:	5
	TXT_Left		6
☒	LAB_Right	&Right:	7

For more information about Tab Thing, press the More button.
For information about other Things, press the Contents button.

Type the tab index number for the first control.

When this option is enabled, you select the controls you want to set by clicking on them.

You can click the controls in the same order in which you want a user to move through them and then immediately click Tab Thing's Set button. Or you can click controls in any order and then arrange the tab order in Tab Thing's list box.

When Click Mode is disabled (unchecked), clicking this button selects all controls in the container (panel or frame) currently selected on your form. If no container is selected, all controls on the form will be loaded into Tab Thing's list.

When Click Mode is disabled (unchecked), clicking this button selects all controls on your form and load their names in Tab Thing's list box.

An "x" means either the control has no TabStop property or it is set to False.

Note. A control may have a tab index number even when the TabStop property is False. Setting the property to False simply means that a user tabbing through controls on the form will skip over this one.

If the control has a TabStop property, you can set it to False by double-clicking on the control's name. An "x" appears before the name.

You can remove the "x" and set the TabStop property back to True by double-clicking a second time.

Controls are listed in proposed tab order. Move a control up or down in the order either by dragging its name with the mouse or by clicking the Up or Down buttons.

Either control captions or full control names are displayed, depending on the option selected at the Setup tab.

Current tab index number. Controls are listed in *proposed* tab order. The tab index number reflects *current* tab order.

Click to move the selected control up in the tab order.

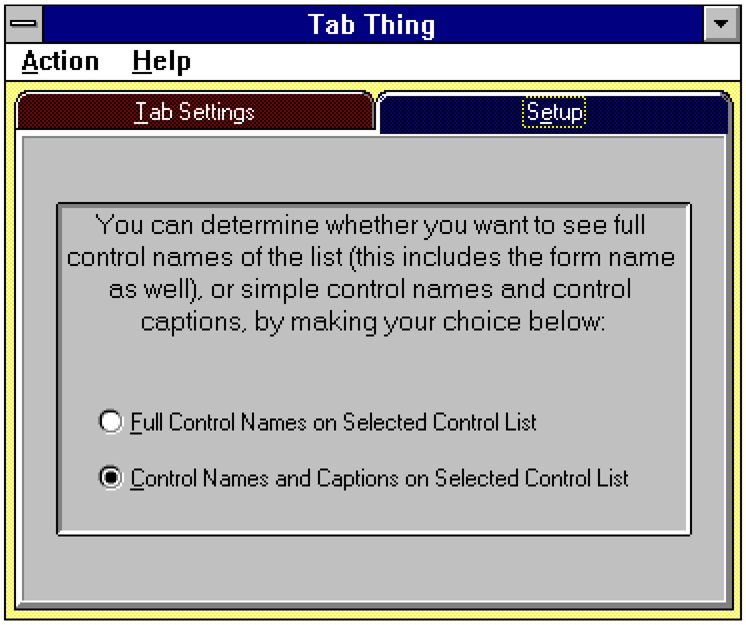
Tip. You can also change a control's position by dragging its name with the mouse.

Click to move the selected control down in the tab order.

Tip. You can also change a control's position by dragging its name with the mouse.

Click to set the tab order.

Click where you need help



You can determine whether you want to see full control names of the list (this includes the form name as well), or simple control names and control captions, by making your choice below:

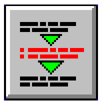
○ Full Control Names on Selected Control List

● Control Names and Captions on Selected Control List

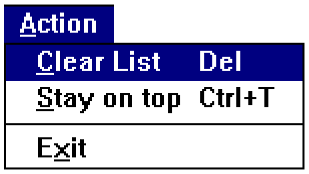
For more information about Tab Thing, press the More button.
For information about other Things, press the Contents button.

When this option is enabled, Tab Thing's list box will display full control names.

When this option is enabled, Tab Thing's list box will display the captions rather than the names of controls.



Click where you need help



For more information about Tab Thing, press the More button.
For information about other Things, press the Contents button.

Clear Tab Thing's list box and start over.

Dent Thing indents code. It can act upon the current procedure, the current file, or the entire project.

There are two ways to use Dent. If you like to indent code while you work, you'll probably want to keep Dent running minimized on your desktop. Whenever you need to fix up the indentations in the routine you are working on, just press the key combination Ctrl+Shift+D. Your code will line up instantly.

On the other hand, you may go into overdrive from time to time and in one nonstop session of coding bang out an entire module or even a whole project. When the dust settles, you can ask Dent to step in and tidy up. Just tell it to indent the project or the file and kick back while Dent Thing does the work.

For more information about Dent Thing, press the More button.
For information about other Things, press the Contents button.

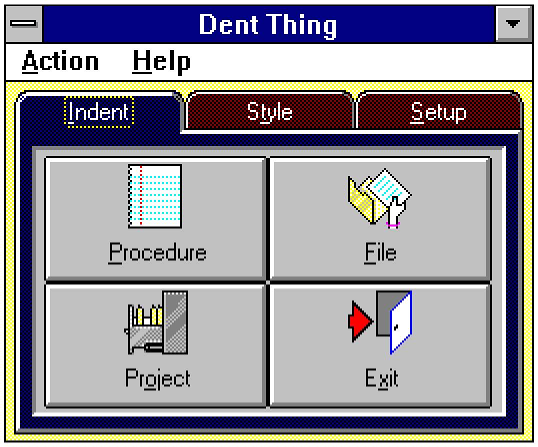
Indent

Style

Setup

Action

Click where you need help



For more information about Dent Thing, press the More button.
For information about other Things, press the Contents button.

Indent the current procedure.

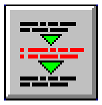
Tip. You can indent the current routine without leaving the VB code window. Just keep Dent Thing minimized on your desktop. To indent the current routine, press Ctrl+Shift+D.

Indent the current file.

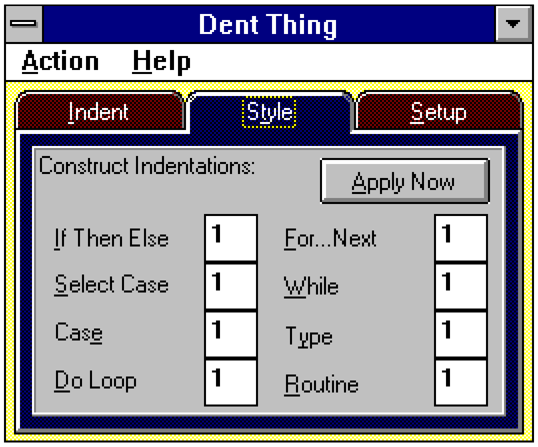
Note. While Dent Thing is working, keyboard and mouse input are disabled.

Indent the current project.

Note. While Dent Thing is working, keyboard and mouse input are disabled.



Click where you need help



Construct Indentations:

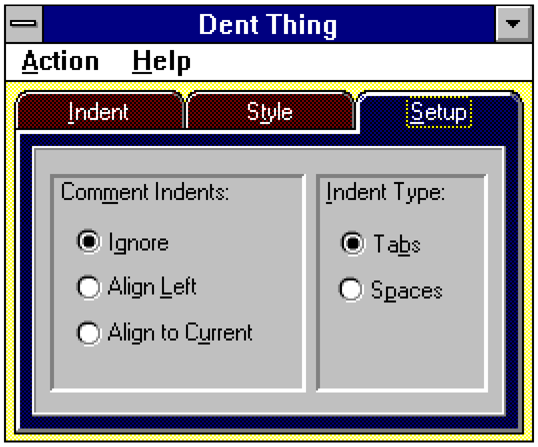
If Then Else	1	For...Next	1
Select Case	1	While	1
Case	1	Type	1
Do Loop	1	Routine	1

For more information about Dent Thing, press the More button.
For information about other Things, press the Contents button.

Click this button to apply your settings to the current routine.

Type the number of units you want to indent code subordinate to each structure. Select a unit (tabs or spaces) at the Setup tab.

Click where you need help



Comment Indents:
[● Ignore] ○ Align Left ○ Align to Current

Indent Type:
[● Tabs] ○ Spaces

For more information about Dent Thing, press the More button.
For information about other Things, press the Contents button.

(Default) Ignore commented lines, leaving them just as they are.

Align commented lines to the left edge of the editing window.

```
' Prompt to save changes
    If Ok_To_Discard_Changes() Then
' Terminate any Selection in progress
        Selecting = False
        MovingSelection = False
    .
    .
    .
    End If
```

Align commented lines with the current indentation.

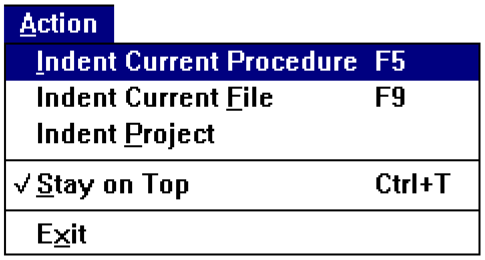
```
' Prompt to save changes
    If Ok_To_Discard_Changes() Then
        ' Terminate any Selection in progress
        Selecting = False
        MovingSelection = False
    .
    .
    .
    End If
```

Sets the unit of indentation to tabs. Control structures are indented the number of tabs specified at the Style tab.

Note. Dent uses the tab stop width you have set for the Visual Basic environment. Remember, though, that unlike most word processors and many ASCII editors, Visual Basic does not use real tabs. When you press the tab key in a code window, what actually gets inserted is a number of spaces--how many depends on the the tab stop width currently in use by the Visual Basic environment. Thus indenting with tabs does not mean that you can later adjust indentations by adjusting the tab stop width.

Sets the unit of indentation to spaces. Control structures are indented the number of spaces specified at the Style tab.

Click where you need help



For more information about Dent Thing, press the More button.
For information about other Things, press the Contents button.

BlockOut Thing comments out or uncomments blocks of code, making it easy to disable an extended selection all at once rather than having to comment line by line, one line at a time. You can optionally stamp comments with the date and time.

You can also use BlockOut to insert boilerplate comments. You may want to use this feature to place a "flower box" prolog at the top of every routine. You can save standard comments as templates for later reuse, and certain variable information -- date, time, project name, file name, and routine name -- can be entered automatically if you include BlockOut's field codes in your block comment.

Additionally, BlockOut can remove commented code from a selection in the Visual Basic editing window. Although you'll want to use this feature carefully, it can come in handy when you've copied a chunk of code from another project, dropped it into the current project, and commented out everything you don't need.

For more information about BlockOut Thing, press the More button.
For information about other Things, press the Contents button.

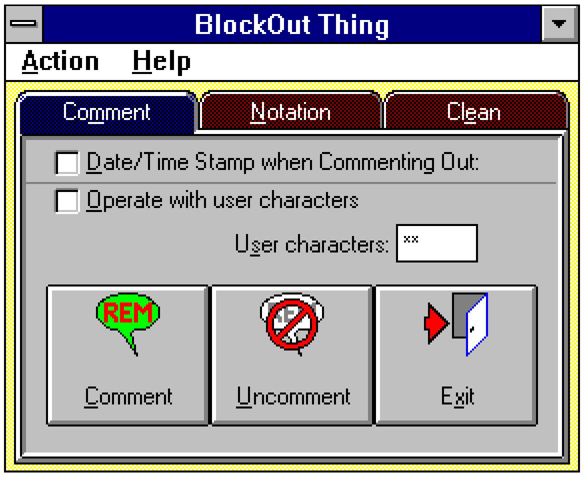
Comment

Notation

Clean

Action

Click where you need help



☐ Date/Time Stamp when Commenting Out:

☐ Operate with user characters

User characters: xx

For more information about BlockOut Thing, press the More button.
For information about other Things, press the Contents button.

Enabling this option causes BlockOut Thing to insert the date and time on the first and last lines of the commented section.

```
' [Start Block]: 6/30/94    4:17:03 PM
 '' Since the Swap statement is not supported by
 '' Visual Basic, this routine is used to perform
 '' the task of swapping two integer values.
 ''
 'Sub Swap_Values (Param1, Param2)
 '
 '    temp = Param1
 '    Param1 = Param2
 '    Param2 = temp
 '
 'End Sub
 ' [End Block]: 6/30/94    4:17:03 PM
```

Enabling this option causes BlockOut Thing to insert additional *user characters* after the apostrophe normally used to mark comments.

```
'**Sub Swap_Values (Param1, Param2)
 '**
 '**    temp = Param1
 '**    Param1 = Param2
 '**    Param2 = temp
 '**
 '**End Sub
```

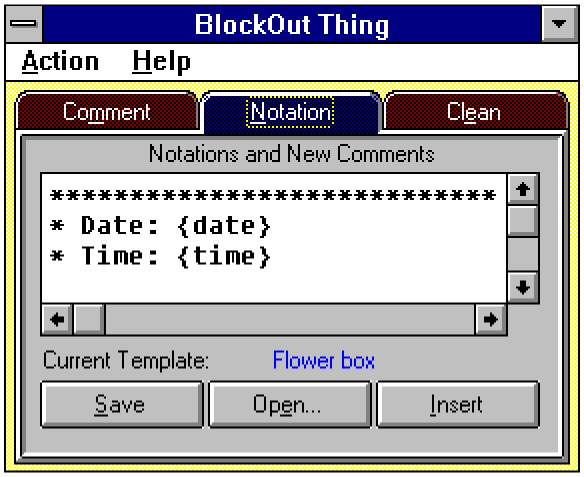
Comments marked with these additional characters are ignored by BlockOut's Clean option and also by Compress Thing's Remove Comments option.

Type the characters you want BlockOut to use as additional comment characters. Spaces are not permitted.

Comment out the current selection in the VB code window.

Remove comment markers from the current selection in the VB code window.

Click where you need help



```
****************************
* Date: {date}
* Time: {time}
```

Current Template: Flower box

For more information about BlockOut Thing, press the More button.
For information about other Things, press the Contents button.

Some of the information you may want to include can be entered automatically if you use *field codes* to represent variable data when you create a boilerplate comment. BlockOut Thing recognizes five fields:

```
{date}
{time}
{project}
{file}
{routine}
```

Note that field codes are enclosed within curly braces.

Type the comment you want to insert.

Tip. You may find it easier to create long comments in Notepad, copy them, and then paste them into BlockOut Thing's edit box.

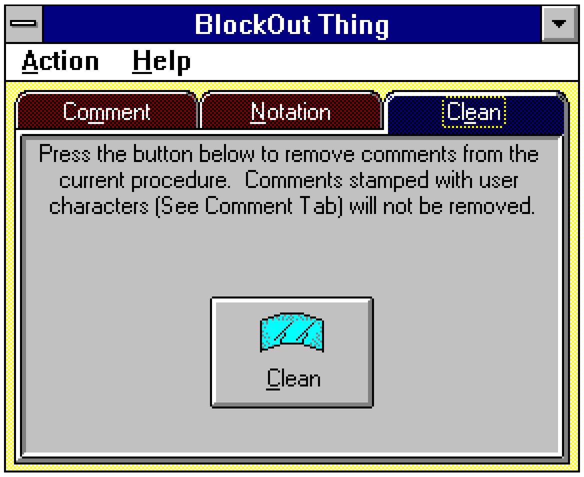
The name of the currently loaded notation template is displayed here.

Click to save the current notation text as a template for later reuse.

Click to open a saved notation template.

Click to paste the block comment at the current insertion point in the VB code window.

Click where you need help



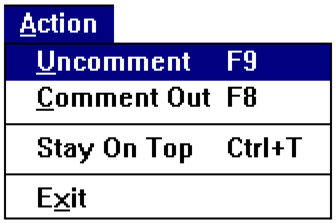
Press the button below to remove comments from the current procedure. Comments stamped with user characters (See Comment Tab) will not be removed.

For more information about BlockOut Thing, press the More button.
For information about other Things, press the Contents button.

All commented lines are removed from the selection highlighted in the VB code window.

Note. If you apply the Clean option and then change your mind, you can reverse the operation by immediately choosing Undo from Visual Basic's Edit menu.

Click where you need help



For more information about BlockOut Thing, press the More button.
For information about other Things, press the Contents button.

Compress Thing identifies and removes unreferenced routines, variables, constants, and instances of user-defined types, as well as any debugging statements, producing a clean set of output files, which you can then compile.

Compress Thing does not actually change your original source files. Duplicate copies of "compressed" forms and modules are placed in a separate directory, which you designate.

For more information about Compress Thing, press the More button.
For information about other Things, press the Contents button.

Summary

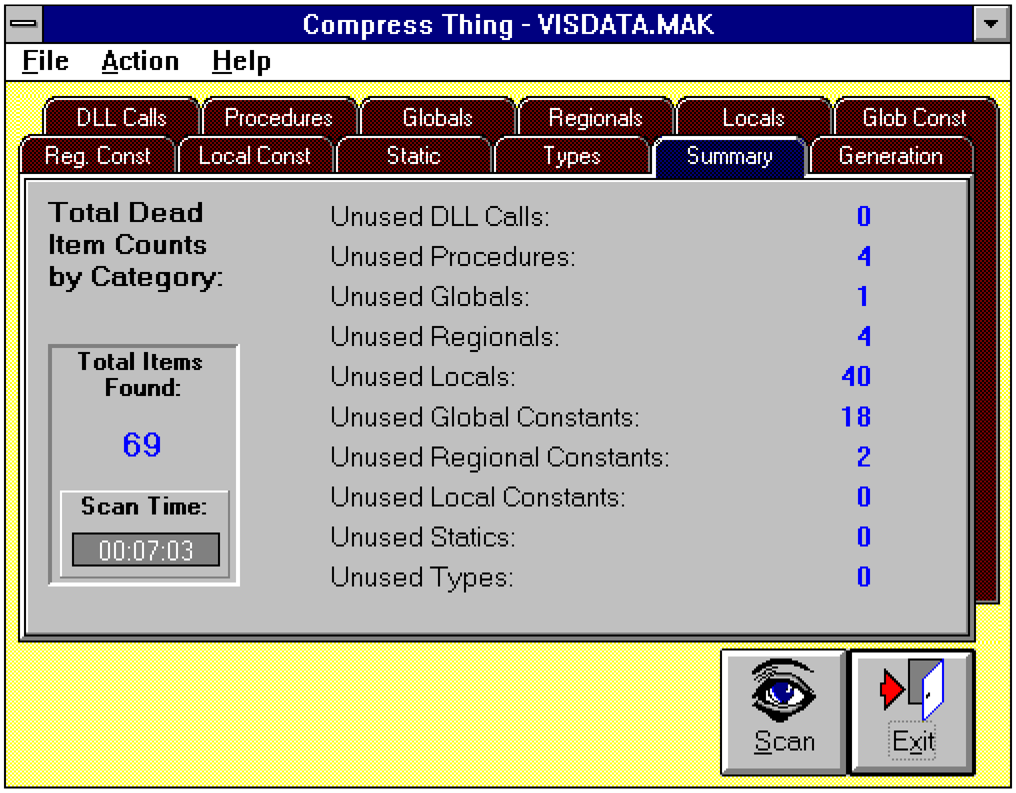
Dead

Generation

File

Action

Click where you need help



Total Dead Item Counts by Category:

Unused DLL Calls:	0
Unused Procedures:	4
Unused Globals:	1
Unused Regionals:	4
Unused Locals:	40
Unused Global Constants:	18
Unused Regional Constants:	2
Unused Local Constants:	0
Unused Statics:	0
Unused Types:	0

Total Items Found:

69

Scan Time:

00:07:03

For more information about Compress Thing, press the More button.
For information about other Things, press the Contents button.

Click to scan the current project.

Note. Compress Thing cannot read files saved in Visual Basic's binary file format. Before scanning a project, make sure that you have saved each file in text format.

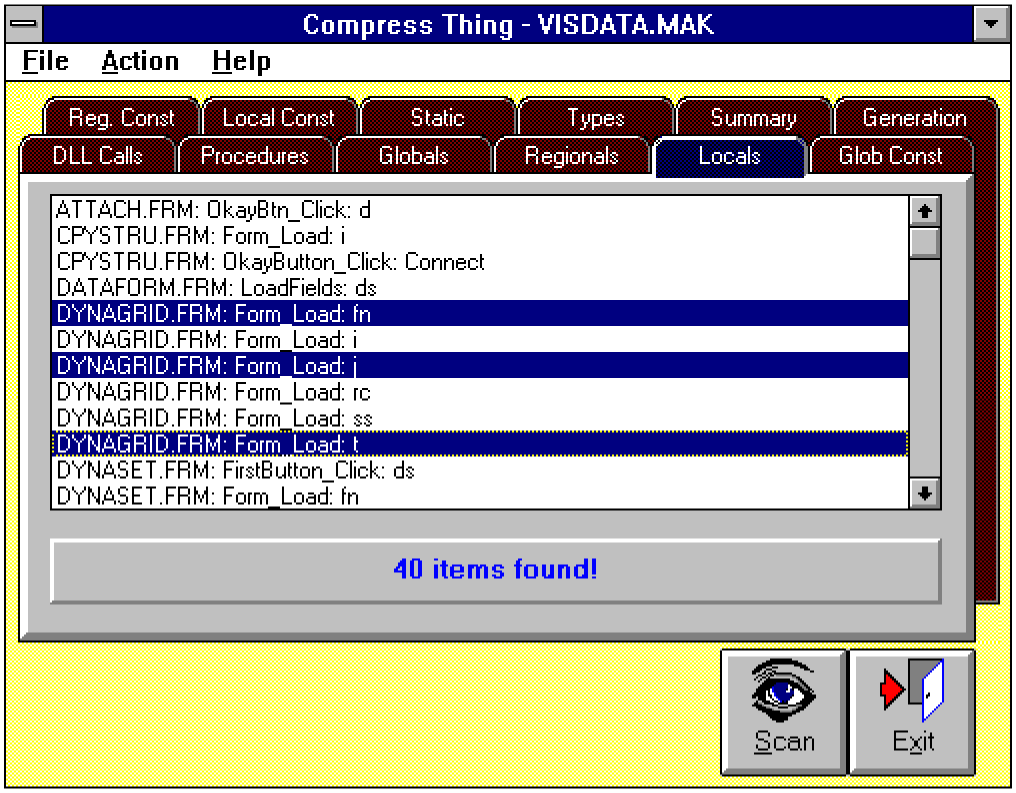
A status bar indicates Compress Thing's progress until scanning is complete. Then the name of the project appears on the title bar, and the Summary tab displays an itemized count of dead items found in the project.

Count by category of the dead items found. You can see the names of the dead items by selecting the appropriate tab.

Total number of dead items found in the project.

The time Compress Thing took to scan the project.

Click where you need help


Compress Thing - VISDATA.MAK

```
ATTACH.FRM: OkayBtn_Click: d
CPYSTRU.FRM: Form_Load: i
CPYSTRU.FRM: OkayButton_Click: Connect
DATAFORM.FRM: LoadFields: ds
DYNAGRID.FRM: Form_Load: fn
DYNAGRID.FRM: Form_Load: i
DYNAGRID.FRM: Form_Load: j
DYNAGRID.FRM: Form_Load: rc
DYNAGRID.FRM: Form_Load: ss
DYNAGRID.FRM: Form_Load: t
DYNASET.FRM: FirstButton_Click: ds
DYNASET.FRM: Form_Load: fn
```

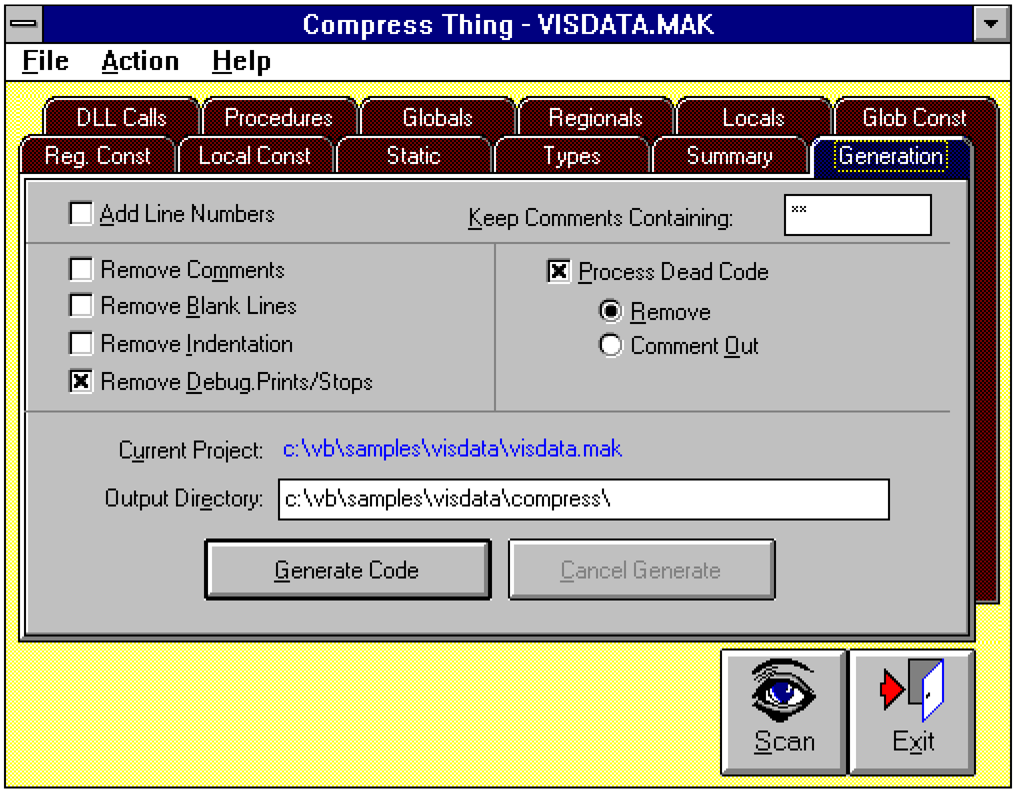
40 items found!

For more information about Compress Thing, press the More button.
For information about other Things, press the Contents button.

Unreferenced code.

Tell Compress Thing what to leave in the compressed output files by selecting items you want to keep. Unselected items will either be removed or commented out, depending on the option you select at the Generation tab.

Click where you need help



[ ] Add Line Numbers

Keep Comments Containing: **

[ ] Remove Comments

[ ] Remove Blank Lines

[ ] Remove Indentation

[✓] Remove Debug.Prints/Stops

[✓] Process Dead Code

● Remove

○ Comment Out

Current Project: c:\vb\samples\visdata\visdata.mak

Output Directory: c:\vb\samples\visdata\compress\

For more information about Compress Thing, press the More button.
For information about other Things, press the Contents button.

Isn't it odd that Visual Basic has a function to report a line number but no way to number lines? If you use the Erl() function in error-handling routines, you can get your lines numbered by compressing your project with this option turned on.

The output files will contain no blank lines. If your code has blank lines for white space, they will be removed.

All lines in the output files will begin at the extreme left hand margin.

Debug.Print and Stop statements will be deleted from the output code.

Specially marked comments, such as those marked with user characters by BlockOut Thing, will be retained in the output files. Type the marker characters in the edit box to the right. (Spaces are not permitted.) To remove all comments, delete all characters from the edit box.

Unreferenced code in Compress Thing's lists will either be removed or commented out. Choose how you want dead code handled by clicking the appropriate radio button.

Unreferenced code will be removed before output files are generated.

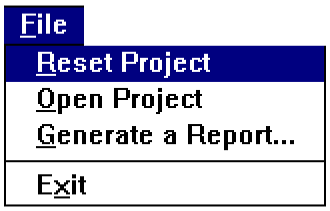
Unreferenced code will be commented out in the generated output files.

Path to the directory where the current project is stored.

Either accept the default, or type the drive and path to the directory where you want the new project files to be stored. If the directory does not exist, Compress Thing creates it.

Cancel generation at any time by clicking this button.

Click where you need help



For more information about Compress Thing, press the More button.
For information about other Things, press the Contents button.

Choose Reset Project if you have opened a project other than the one currently loaded in Visual Basic and then decide that you want to scan the project currently loaded in Visual Basic.

Choose Open Project to open a project other than the one currently loaded in Visual Basic.

Choose Generate a Report to save the results of the scan in a report file that you can print and read later. Compress Thing creates the report in ASCII text format, suitable for reading and printing with Notepad.

Click where you need help

Action

Stay on Top Ctrl+T

For more information about Compress Thing, press the More button.
For information about other Things, press the Contents button.