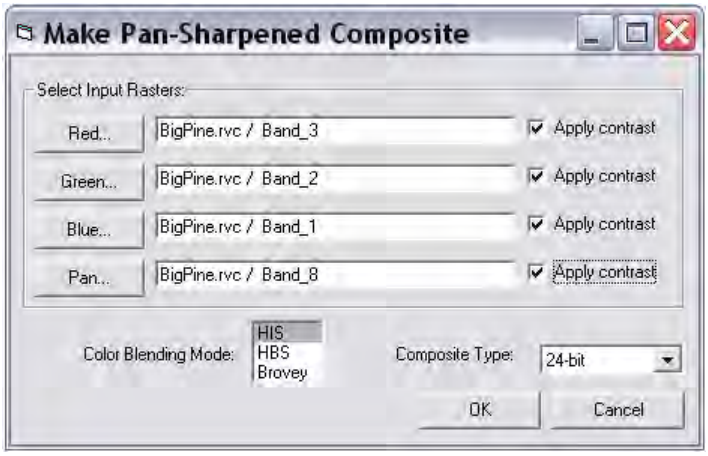
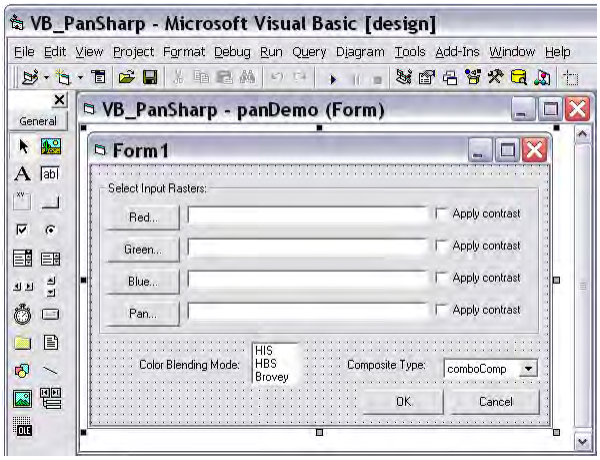
Build SML Dialogs Using Visual Basic

SML scripts that you develop for colleagues or clients can be easy for them to use if you provide custom dialog windows to simplify entry or selection of process settings. There are several ways in which you can develop and use custom dialogs with an SML script in the TNT products. In addition to using either the OSF/Motif classes in SML or a dialog specification in XML-format, you can also create the dialog within an ActiveX component program that is called by and communicates with your SML script. You can use any programming language that supports ActiveX (such as Visual Basic, C++, or Java) to develop the dialog program. If you have experience with one of these languages or prefer to use the form layout tools provided with Visual Basic, this may be the easiest way for you to create custom dialogs for your SML scripts.



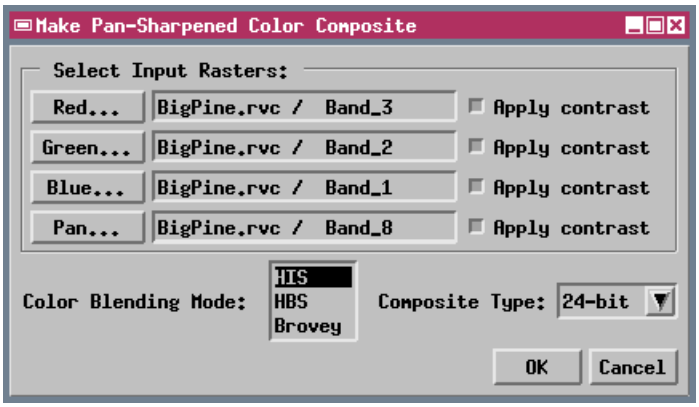
Control dialog for the VB_PanSharp demonstration created in Visual Basic, composed of Windows control components.



The design mode in the Visual Basic module of Microsoft Visual Studio provides a graphical editor for adding and arranging the controls in your dialog (referred to in Visual Basic as a form) as well as dialogs for setting their properties.

To demonstrate the use of an ActiveX component dialog with SML, MicroImages has created a demonstration called VB_PanSharp, which creates a pan-sharpened color composite raster using input objects and processing parameters set in a Visual Basic dialog. This demonstration is derived from the PanSharpComp SML script (which uses a dialog specification in XML) that is distributed as a sample with the *Building Dialogs in SML* tutorial booklet. Both Visual Basic and XML versions of the control dialog are illustrated below for comparison.

In the VB_PanSharp application, the SML script imports the Visual Basic class VBform, which provides the control properties, methods, and data structures associated with the dialog (form) in the VB module panDemo. The code in these two modules enables two-way communication between the two programs. The SML script is able to obtain information on the dialog settings from members of the imported Visual Basic class, and events in the Visual Basic dialog (such as pressing one of its buttons) can trigger actions in the SML script.



Control dialog for the original PanSharpComp SML script, defined by an XML dialog specification.

The Visual Basic code for this application is shown on the other side of this page, along with download and installation instructions. The accompanying color plate entitled *ActiveX Callbacks to SML* provides more detailed information on methods of communication between an ActiveX component program and an SML script.

Visual Basic Source Code for VB_PanSharp Demo

The SML script and Visual Basic files necessary to run this demonstration, along with the Visual Basic source code files shown below, are available for free download at: www.microimages.com/freestuf/smlscripts.htm. After downloading and unzipping the VB_PanSharp file, run the Setup program in the Package subdirectory to register the ActiveX component program.

VB_PanSharp: code for Form panDemo

```
Option Explicit
Event OnbtnRed()
Event OnbtnGreen()
Event OnbtnBlue()
Event OnbtnPan()
Event OnbtnOK()
Event OnbtnCancel()

Private Sub btnBlue_Click()
    RaiseEvent OnbtnBlue
End Sub

Private Sub btnCancel_Click()
    RaiseEvent OnbtnCancel
    Me.Visible = False
End Sub

Private Sub btnGreen_Click()
    RaiseEvent OnbtnGreen
End Sub

Private Sub btnOK_Click()
    RaiseEvent OnbtnOK
End Sub

Private Sub btnPan_Click()
    RaiseEvent OnbtnPan
End Sub

Private Sub btnRed_Click()
    RaiseEvent OnbtnRed
End Sub
```

declare event names for the dialog's push-buttons

assign event name to the predefined button-press event for each push-button

VB_PanSharp: class VBform

class called ("imported") by the SML script

```
Option Explicit
Private WithEvents mDlg As panDemo
Public Event OnbtnRed()
Public Event OnbtnGreen()
Public Event OnbtnBlue()
Public Event OnbtnPan()
Public Event OnClose()
Public Event OnbtnOK()

Private Sub Class_Initialize()
    Set mDlg = New panDemo
    mDlg.comboComp.ListIndex = 1
    mDlg.listBlend.ListIndex = 0
End Sub

Private Sub Class_Terminate()
    On Error Resume Next
    mDlg.Hide
    Unload mDlg
End Sub

Public Sub RedSelected()
    mDlg.checkRed.Enabled = True
    mDlg.btnGreen.Enabled = True
End Sub

Public Sub GreenSelected()
    mDlg.checkGreen.Enabled = True
    mDlg.btnBlue.Enabled = True
End Sub

Public Sub BlueSelected()
    mDlg.checkBlue.Enabled = True
    mDlg.btnPan.Enabled = True
End Sub

Public Sub PanSelected()
    mDlg.checkPan.Enabled = True
    mDlg.btnOK.Enabled = True
End Sub
```

get button event names and definitions for dialog from panDemo code

when this class is initialized, set the panDemo object as source for the dialog and set default selections for list and combobox controls

when this class terminates, close the dialog and clear it from memory

when red band selected, enable contrast option for red and button for green

when green band selected, enable contrast option for green and button for blue

when blue band selected, enable contrast option for blue and button for pan

when pan band selected, enable contrast option for pan and button for OK

```
Public Sub ShowDialog()
   mDlg.Caption = "Make Pan-Sharpened Composite"
   mDlg.Show vbModal
End Sub

Public Sub HideDialog()
    mDlg.Hide
End Sub

Public Property Let RedName(ByVal str As String)
    mDlg.txtRed.Text = str
End Property
Public Property Let BlueName(ByVal str As String)
    mDlg.txtBlue.Text = str
End Property
Public Property Let GreenName(ByVal str As String)
    mDlg.txtGreen.Text = str
End Property
Public Property Let PanName(ByVal str As String)
    mDlg.txtPan.Text = str
End Property

Public Property Get depth() As Integer
    depth = mDlg.comboComp.ListIndex
End Property

Public Property Get conred() As Integer
    conred = mDlg.checkRed.Value
End Property
Public Property Get conblue() As Integer
    conblue = mDlg.checkBlue.Value
End Property
Public Property Get congreen() As Integer
    congreen = mDlg.checkGreen.Value
End Property
Public Property Get conpan() As Integer
    conpan = mDlg.checkPan.Value
End Property

Public Property Get method() As String
    Select Case mDlg.listBlend.ListIndex
    Case 0
        method = "HIS"
    Case 1
        method = "HBS"
    Case 2
        method = "Brovey"
    End Select
End Property

Public Sub CloseForm()
    mDlg.Hide
    Unload mDlg
End Sub

Private Sub mDlg_OnbtnBlue()
    RaiseEvent OnbtnBlue
End Sub
Private Sub mDlg_OnbtnRed()
    RaiseEvent OnbtnRed
End Sub
Private Sub mDlg_OnbtnGreen()
    RaiseEvent OnbtnGreen
End Sub
Private Sub mDlg_OnbtnPan()
    RaiseEvent OnbtnPan
End Sub
Private Sub mDlg_OnbtnOK()
    RaiseEvent OnbtnOK
End Sub
Private Sub mDlg_OnCancel()
    Class_Terminate
End Sub
```

show modal dialog

hide modal dialog

assign raster name (passed in from SML script) to text box next to each input raster button

get the selected bit-depth value from the combobox control as a public variable accessible to SML

get the status of the contrast checkbox controls as public variables accessible to SML

get the blending method from the list control as a public variable accessible to SML

close the dialog and remove it from memory

activate SML callback for each dialog push-button when it is pressed