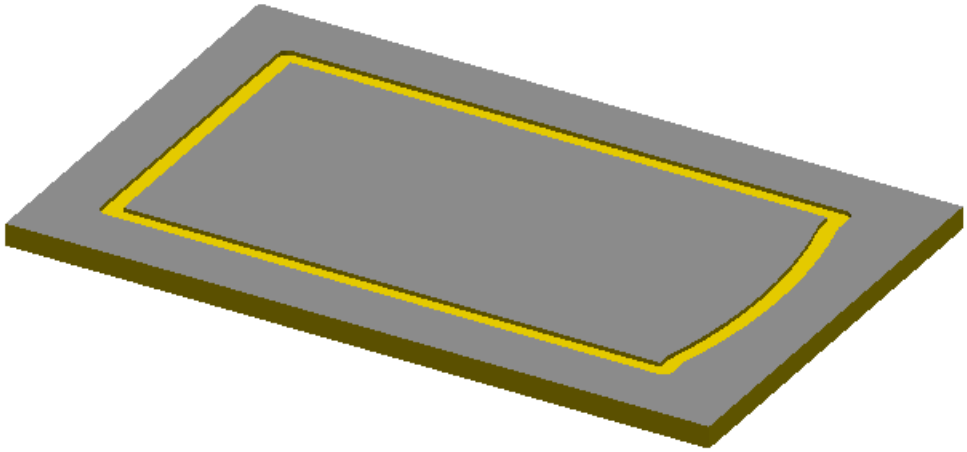
AlphaCAM

VBA

Router Example

The Project

In this project we are going to draw a door front with an arched panel from information supplied by the user and then machine the door complete. The project will contain a single form and two modules.



Creating the New Project

Open a new VBA Project by selecting **New VBA Project** from the **VBA Macros** option in the **Utils** pull down menu. This will open the VBA editor and create a new empty project. Double click on the name property in the Properties Window and rename the project to *CathedralDoor*. From the **File** pull down menu select **Save** and save the project as *CathedralDoor.arb* in the following folder *C:\licomdir\VBMacros\StartUp\VBA Training*.

Creating the Form

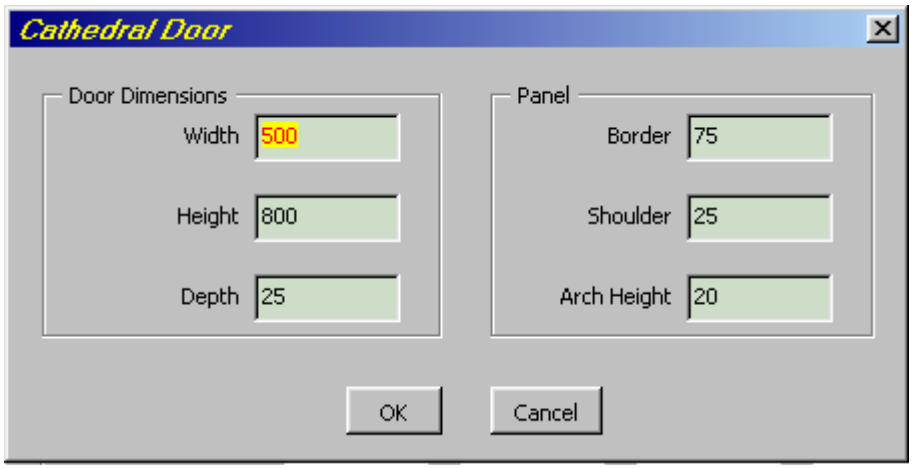
Insert a new form into your project by selecting the **Insert** pull down menu and clicking on **UserForm**.

The new form will need to contain the following controls

- 2 Frames
- 6 Text Boxes
- 6 Labels
- 2 Command Buttons

New controls are added to the form by single clicking on the desired control in the **toolbox** and then single clicking on the **userform**. If your **toolbox** is not visible it can be made visible by selecting **Toolbox** from the **View** pull down menu.

The form you are going to create will look like this



Set the caption property for each control as shown in the image above.

To make these new controls easy to identify it is best if they are given names that relate to what they are, and what you want to use them for. To set the name property for each control you single click on the control you wish to name in the **UserForm window** and then double click on the name property in the **Properties Window**. This will allow you easy access to each control at any time from within the project.

For this project we are going to use the name properties for the controls as follows

- Userform ~ frmMain
- Frame1 ~ fraDoor
- Label1 ~ lblWidth
- Textbox1 ~ txtWidth
- Label2 ~ lblHeight
- Textbox2 ~ txtHeight
- Label3 ~ lblDepth
- Textbox3 ~ txtDepth
- Command1 ~ cmdOK

Frame2 ~ fraPanel
Label4 ~ lblBorder
Textbox4 ~ txtBorder
Label5 ~ lblShoulder
Textbox5 ~ txtShoulder
Label6 ~ lblArch
Textbox6 ~ txtArch
Command2 ~ cmdCancel

It is quite likely that you will not have inserted the controls onto the form into a suitable order for moving around the form using the tab key on the keyboard. This can be changed by using the following procedure.

Set the Tab Order for the form by right clicking on the background of the form and selecting **Tab Order** from the **pop up window**.

fraDoor
fraPanel
cmdOK
cmdCancel

Set the Tab Order for the frame *fraDoor* by right clicking on the background of the frame and selecting **Tab Order** from the **pop up window**.

lblWidth
txtWidth
lblHeight
txtHeight
lblDepth
txtDepth

Set the Tab Order for the frame *fraPanel* by right clicking on the background of the frame and selecting **Tab Order** from the **pop up window**.

lblBorder
txtBorder
lblShoulder
txtShoulder
lblArch
txtArch

Writing the Code

The first thing we need to do is create a new menu so that we can run our project. To do this we need to insert a new **module** into the project. This is done by selecting the **Insert** pull down menu and clicking on **Module**. The module we are creating is special module because AlphaCAM needs to access it on initialisation to add a new menu. For this to happen the module has to have the name **Events**. To rename the module double click on the name property in the properties window and type in the word Events. In the code window for the *Events* module we are going to add two new functions, one to add the new menu and one to show the userform when the new menu item is selected. The code is as follows ~

```
Public Function InitAlphacamAddIn(acamversion As Long) As Integer
    Dim fr As Frame
    Set fr = App.Frame
    With fr
        ' set up itemname  and menuname as new string variables
        Dim ItemName As String, MenuName As String
        ItemName = "Cathedral door": MenuName = "Standard Doors"
        ' create the new menu
        .AddMenuItem2 ItemName, "ShowfrmMain", acamMenuNEW, MenuName
    End With
    InitAlphacamAddIn = 0
End Function

Function ShowFrmMain()
    ' show the main form
    Load frmMain
    frmMain.Show
End Function
```

To see the new menu you will need to save the project, then close and restart **AlphaCAM**. This will allow **AlphaCAM** to read your new **module** and add the new menu.

Select the new option and your form will be displayed.

At the moment there is no code associated with the form so the only way to exit the form is to select the **'X'** in the top right corner of the form.

To continue editing the project we will need to reopen the project. Do this by selecting **Open VBA Project** from **VB Macros** option in the **Utils** pulldown menu.

Each control on the form and the form itself can have its own unique piece of code. The easiest way to access the code for each control is by double clicking on the desired control in the **Userform Window**. This will open a new window called the **Code Window** and insert an empty subroutine based on the control's default event.

We are going to set some default values for the form by using the form's initialize event.

Double click on the background of the form in the **Userform Window**.

This will open the **Code Window** with the form's default event, which is the **click event** as shown below.

```
Private Sub UserForm_Click()

End Sub
```

To change this to the initialize event there is a drop down box at the top right corner of the **Code Window** listing all the events available for the currently active control. Click on this and position your mouse over the word **Initialize** and then click again. This will create the following new code for you.

```
Private Sub UserForm_Initialize()

End Sub
```

Modify this new sub so that it looks like the following code

```
Private Sub UserForm_Initialize()
    ' set defaults for door
    txtWidth = 500
    txtHeight = 800
    txtDepth = 25

    ' set defaults for panel
    txtBorder = 75
    txtShoulder = 25
    txtArch = 20

    ' set focus to first text box and highlight
    txtWidth.SetFocus
    txtWidth.SelStart = 0
    txtWidth.SelLength = 999
End Sub
```

Write the code for the click event for the command button *cmdCancel* so that it will end the project. The easiest way to create the new **sub** is to double click on the word *frmMain* in the **Project Explorer Window.** This will open the userform so that you can double click on the *Cancel* button on the form. This will create the following new code for you.

```
Private Sub cmdCancel_Click()

End Sub
```

Modify this new sub so that it looks like the following code

```
Private Sub cmdCancel_Click()
    End      ' end VBA macro
End Sub
```

To enable the **Esc** key on the keyboard to act in the same way as the *cmdCancel* button you can set the *cancel* property for the *cmdCancel* button to *True*.

Save the project and switch to AlphaCAM so that you can test the new code by selecting *Cathedral Door* from the *Standard Doors* pulldown menu.

To write the main code to draw and machine the door panel it would be possible to put all the code in the click event for the *cmdOK* button. The main problem with doing this is that in a large project it would become difficult to follow and debug the code. It would also mean that you may have to write the same piece of code more than once because a similar feature is required. To overcome this problem it is best to create a new **Module**, and write common functions into it. Calls to these functions can then be made from any point in the project.

Insert a new module into the project and set its name property to **Main**.

In this module we are going to write a new function to draw and machine the door. This can be achieved in two ways, either select **Procedure** from **Insert** menu or type the following code directly into the module.

```
Function CreateCathedralDoor()

End Function
```

This function will need to know the dimensions for the door as the user has typed them into the form. To do so, we edit the name of the function to include the variables we want to pass to it. This is shown below.

```
Function CreateCathedralDoor( _
    Height As Double, Width As Double, Depth As Double, _
    Border As Double, Shoulder As Double, Arch As Double)

End Function
```

To make a call to this function and send the required values to it edit the click event for the *cmdOk* button to hide the form and make a call to the function. The easiest way to create the new **sub** is to double click on the word *frmMain* in the **Project Explorer Window.** This will open the userform so that you can double click on the *OK* button on the form. This will create the following new code for you.

```
Private Sub cmdOK_Click()

End Sub
```

Modify this new sub so that it looks like the following code

```
Private Sub cmdOK_Click()
    frmMain.Hide      ' hide the form
    DoEvents          ' stop the project from processing until it has completed all previous tasks

    ' call the function to create the cathedral door
    CreateCathedralDoor val(txtHeight), val(txtWidth), val(txtDepth), _
                        val(txtBorder), val(txtShoulder), val(txtArch)
End Sub
```

The following features will need to be edited into the *CreateCathedralDoor* function.

1. Define the active drawing
2. Create a work volume
3. Create a material
4. Draw the outside of the door
5. Machine the outside of the door
6. Draw the panel
7. Machine the panel

Double click on the word *Main* in the **Project Explorer Window** to see the code for the module *Main*. Make the following edits to the *CreateCathedralDoor* function.

1. Defining the Active Drawing

```
' define the active drawing
Dim drw as Drawing
Set drw = App.ActiveDrawing
```

2. Creating the work volume

```
' create the work volume
Dim WorkVol As Path
Set WorkVol = drw.CreateRectangle(0, 0, Height, Width)
WorkVol.SetWorkVolume 0, -Depth
```

3. Creating the material

```
' create the material
Dim Material As Path
Set Material = drw.CreateRectangle(-1, -1, Height + 1, Width + 1)
Material.SetMaterial 0, -Depth
```

4. Creating the outside of the door

```
' create the outside of the door
Dim DoorGeo As Path
Set DoorGeo = drw.CreateRectangle(0, 0, Height, Width)
DoorGeo.ToolInOut = acamOUTSIDE
```

Switch to AlphaCAM and test the project by running it from the pulldown menu.

5. Machining the outside of the door

```
 '  machine the outside of the door
DoorGeo.Selected = True
Dim cut_door As Paths
Dim Tool As MillTool
On Error Resume Next        ' ignore any errors
   Set Tool = App.SelectTool(App.LicomdatPath & "licomdat\rtools.alp\Flat - 10mm.art")
On Error GoTo 0             ' cancel error ignore
' if the tool is not found show the normal tool select dialog box
If Tool Is Nothing Then
   Set Tool = App.SelectTool("$User")  ' allow the user to select a tool
End If
' create the machining data for the outside of the door
Dim cut_door_data As MillData
Set cut_door_data = App.CreateMillData
With cut_door_data
   .FinalDepth = -Depth - 5
   .MaterialTop = 0
   .NumberOfCuts = 1
   .OffsetNumber = 1
   .RapidDownTo = 5
   .SafeRapidLevel = 50
   .Stock = 0
   .XYCorners = acamCornersSTRAIGHT
   Set cut_door = .RoughFinish
End With
' apply lead in and lead out
cut_door.Item(1).SetLeadInOutAuto acamLeadARC, acamLeadARC, 1.2, _
   1.2, 90, False, False, 0
```

Switch to AlphaCAM and test the project by running it from the pulldown menu.

5. Creating the panel

```
' create the panel
Dim tempgeo As Geo2D
Dim PanelGeo As Path
Dim PanelxStart As Double, PanelyStart As Double
Dim PanelxFin As Double, PanelyFin As Double
PanelxStart = Border: PanelyStart = Border
PanelxFin = Height - Border - Arch: PanelyFin = Width - Border
Set tempgeo = drw.Create2DGeometry(PanelxStart, PanelyStart)
With tempgeo
   .AddLine PanelxFin, PanelyStart
   .AddLine PanelxFin, PanelyStart + Shoulder
   .AddArc2Point PanelxFin + Arch, Width / 2, PanelxFin, PanelyFin - Shoulder
   .AddLine PanelxFin, PanelyFin
   .AddLine PanelxStart, PanelyFin
   Set PanelGeo = .CloseAndFinishLine
End With
PanelGeo.ToolInOut = acamINSIDE
PanelGeo.SetStartPoint PanelxStart + ((PanelxFin - PanelxStart) / 2), PanelyStart
```

Switch to AlphaCAM and test the project by running it from the pulldown menu.

6. Machining the panel

```
' machine the panel
PanelGeo.Selected = True
Dim cut_panel As Paths
Set Tool = Nothing
On Error Resume Next        ' ignore any errors
Set Tool = App.SelectTool(App.LicomdatPath &"licomdat\rtools.alp\Flat - 20mm.art")
On Error GoTo 0             ' cancel error ignore
' if the tool is not found show the normal tool select dialog box
If Tool Is Nothing Then
   Set Tool = App.SelectTool("$User")    ' allow the user to select a tool
End If
' create the machining data for the panel
Dim cut_panel_data As MillData
Set cut_panel_data = App.CreateMillData
With cut_panel_data
   .FinalDepth = -5
   .MaterialTop = 0
   .NumberOfCuts = 1
   .OffsetNumber = 1
   .RapidDownTo = 5
   .SafeRapidLevel = 50
   .Stock = 0
   .XYCorners = acamCornersSTRAIGHT
   Set cut_panel = .RoughFinish
End With
' apply lead in and lead out
cut_panel.Item(1).SetLeadInOutAuto acamLeadLINE, acamLeadLINE, Tool.Diameter, _
   Tool.Diameter, 0, True, True, 0
```

Switch to AlphaCAM and test the project by running it from the pulldown menu.

Add a new function to the module **Main** to check if the active drawing has any geometries in it. If it has, show a warning to allow the user to save any unsaved data if they want to.

```
Function FileNew()
    ' function to test if active drawing has any geometries
    ' and show a warning that any unsaved data will be lost
    Dim MsgText As String
    MsgText = "This will open a new drawing, press OK to continue"
    Dim MsgBoxReturn As Integer
    If App.ActiveDrawing.GetGeoCount > 0 Then
        MsgBoxReturn = MsgBox(MsgText, vbOKCancel)
        If MsgBoxReturn = vbOK Then
            App.New
        Else
          End     ' exit VBA macro
        End If
    End If
End Function
```

Modify the sub *ShowFrmMain* in the module **Events** to include a call to the new function.

```
Sub ShowFrmMain ()
    ' run function to test if the active drawing has any geometries
    FileNew
   '  show main dialog box
   Load frmMain
   frmMain.Show
End Sub
```

Add a new function to the module **Main** to refresh the screen.

```
Function Refresh()
    With App.ActiveDrawing
        .ThreeDViews = True
        .Options.ShowRapids = False
        .Options.ShowTools = False
        .Redraw
    End With
End Function
```

Modify the sub *ShowFrmMain* in the module **Events** to include a call to the new function.

```
Sub ShowFrmMain()
    ' run function to test if the active drawing has any geometries
   NewDrawing
   '  show main dialog box
   Load frmMain
   frmMain.Show
  ' run function to refresh the screen
    Refresh
End Sub
```