```
Option Explicit
Dim staX As Long    'Need in PlanSpanLast to know where to end centerline in Span 1

'Copyright 2010 Ted Cashin

Private Sub cboBarrierSidewalk_Change()

Dim strBarrier As String

strBarrier = cboBarrierSidewalk.Value

Select Case strBarrier
    Case "Barrier"
        lblParapetHeight.Visible = False
        txtParapetHeight.Visible = False
        lblSidewalkWidth.Visible = False
        txtSidewalkWidth.Visible = False
    Case "Sidewalk"
        lblParapetHeight.Visible = True
        txtParapetHeight.Visible = True
        lblSidewalkWidth.Visible = True
        txtSidewalkWidth.Visible = True
End Select

End Sub

Private Sub cboScale_Change()
txtScale.Tag = "Default"
Select Case cboScale.Value
    Case "10"
        txtScale = "120"
    Case "20"
        txtScale = "240"
    Case "30"
        txtScale = "360"
    Case "40"
        txtScale = "480"
    Case "50"
        txtScale = "600"
    Case "60"
        txtScale = "720"
    Case "1/8"
        txtScale = "96"
    Case "3/16"
        txtScale = "64"
    Case "1/4"
        txtScale = "48"
    Case "3/8"
        txtScale = "32"
    Case "1/2"
        txtScale = "24"
    Case "3/4"
        txtScale = "16"
End Select
txtScale.Tag = ""
End Sub


Private Sub chkNoVC_Click()
'When No Vertical Curve is checked, hide LVC box and LVC2 data
If chkNoVC.Value = True Then
    lblLVC1.Visible = False
    txtLVC1.Visible = False
    chkPVI2.Value = False
```

```
    chkPVI2.Visible = False
Else
    lblLVC1.Visible = True
    txtLVC1.Visible = True
    chkPVI2.Visible = True
    chkPVI2_Click
End If
End Sub

Private Sub chkPVI2_Click()
'Shows or hides info for a second vertical curve
 Select Case chkPVI2.Value
    Case True
        lblStaPVI2.Visible = True
        txtStaPVI2.Visible = True
        lblLVC2.Visible = True
        txtLVC2.Visible = True
        lblGrade3.Visible = True
        txtGrade3.Visible = True
    Case Else
        lblStaPVI2.Visible = False
        txtStaPVI2.Visible = False
        lblLVC2.Visible = False
        txtLVC2.Visible = False
        lblGrade3.Visible = False
        txtGrade3.Visible = False
End Select
End Sub


Private Sub txtNumSpans_Change()
Dim NumSpans As Integer
NumSpans = Val(txtNumSpans.Value)

'Show or hide the fields for spans lengths based on value in txtNumSpans
Select Case NumSpans
    Case 2
        txtSpan1.Visible = True
        txtSpan2.Visible = True
        txtSpan3.Visible = False
        txtSpan4.Visible = False
        txtSpan5.Visible = False
        txtSpan6.Visible = False
        txtSpan7.Visible = False
        txtSpan8.Visible = False

        lblSpan1.Visible = True
        lblSpan2.Visible = True
        lblSpan3.Visible = False
        lblSpan4.Visible = False
        lblSpan5.Visible = False
        lblSpan6.Visible = False
        lblSpan7.Visible = False
        lblSpan8.Visible = False

    Case 3
        txtSpan1.Visible = True
        txtSpan2.Visible = True
        txtSpan3.Visible = True
        txtSpan4.Visible = False
        txtSpan5.Visible = False
        txtSpan6.Visible = False
        txtSpan7.Visible = False
        txtSpan8.Visible = False
```

```
        lblSpan1.Visible = True
        lblSpan2.Visible = True
        lblSpan3.Visible = True
        lblSpan4.Visible = False
        lblSpan5.Visible = False
        lblSpan6.Visible = False
        lblSpan7.Visible = False
        lblSpan8.Visible = False

    Case 4
        txtSpan1.Visible = True
        txtSpan2.Visible = True
        txtSpan3.Visible = True
        txtSpan4.Visible = True
        txtSpan5.Visible = False
        txtSpan6.Visible = False
        txtSpan7.Visible = False
        txtSpan8.Visible = False

        lblSpan1.Visible = True
        lblSpan2.Visible = True
        lblSpan3.Visible = True
        lblSpan4.Visible = True
        lblSpan5.Visible = False
        lblSpan6.Visible = False
        lblSpan7.Visible = False
        lblSpan8.Visible = False

    Case 5
        txtSpan1.Visible = True
        txtSpan2.Visible = True
        txtSpan3.Visible = True
        txtSpan4.Visible = True
        txtSpan5.Visible = True
        txtSpan6.Visible = False
        txtSpan7.Visible = False
        txtSpan8.Visible = False

        lblSpan1.Visible = True
        lblSpan2.Visible = True
        lblSpan3.Visible = True
        lblSpan4.Visible = True
        lblSpan5.Visible = True
        lblSpan6.Visible = False
        lblSpan7.Visible = False
        lblSpan8.Visible = False

    Case 6
        txtSpan1.Visible = True
        txtSpan2.Visible = True
        txtSpan3.Visible = True
        txtSpan4.Visible = True
        txtSpan5.Visible = True
        txtSpan6.Visible = True
        txtSpan7.Visible = False
        txtSpan8.Visible = False

        lblSpan1.Visible = True
        lblSpan2.Visible = True
        lblSpan3.Visible = True
        lblSpan4.Visible = True
        lblSpan5.Visible = True
        lblSpan6.Visible = True
```

```
        lblSpan7.Visible = False
        lblSpan8.Visible = False

    Case 7
        txtSpan1.Visible = True
        txtSpan2.Visible = True
        txtSpan3.Visible = True
        txtSpan4.Visible = True
        txtSpan5.Visible = True
        txtSpan6.Visible = True
        txtSpan7.Visible = True
        txtSpan8.Visible = False

        lblSpan1.Visible = True
        lblSpan2.Visible = True
        lblSpan3.Visible = True
        lblSpan4.Visible = True
        lblSpan5.Visible = True
        lblSpan6.Visible = True
        lblSpan7.Visible = True
        lblSpan8.Visible = False

    Case 8
        txtSpan1.Visible = True
        txtSpan2.Visible = True
        txtSpan3.Visible = True
        txtSpan4.Visible = True
        txtSpan5.Visible = True
        txtSpan6.Visible = True
        txtSpan7.Visible = True
        txtSpan8.Visible = True

        lblSpan1.Visible = True
        lblSpan2.Visible = True
        lblSpan3.Visible = True
        lblSpan4.Visible = True
        lblSpan5.Visible = True
        lblSpan6.Visible = True
        lblSpan7.Visible = True
        lblSpan8.Visible = True

    Case Else
        MsgBox Format(NumSpans) & " span(s) is not supported)"
    End Select

End Sub


Private Sub txtScale_Change()
'If scale is manually changed, make the combo box selection blank
'If scale is being changed by the combo box, then set the Tag = "Default"
If txtScale.Tag = "" Then
    cboScale.Value = ""
End If

End Sub



Private Sub UserForm_Activate()
'Hide controls
lblStaPVI2.Visible = False
txtStaPVI2.Visible = False
lblLVC2.Visible = False
```

```
txtLVC2.Visible = False
lblGrade3.Visible = False
txtGrade3.Visible = False
txtSpan2.Visible = False
lblSpan2.Visible = False
txtSpan3.Visible = False
lblSpan3.Visible = False
txtSpan4.Visible = False
lblSpan4.Visible = False
txtSpan5.Visible = False
lblSpan5.Visible = False
txtSpan6.Visible = False
lblSpan6.Visible = False
txtSpan7.Visible = False
lblSpan7.Visible = False
txtSpan8.Visible = False
lblSpan8.Visible = False


'Initialize combo boxes
cboSkewDir.AddItem "Forward / / /", -1
cboSkewDir.AddItem "Backward \ \ \", 1
'Initial value
cboSkewDir.Value = "Forward / / /"

cboBentType.AddItem "Pile Bent"
cboBentType.AddItem "Concrete"
'Initial value
cboBentType.Value = "Concrete"

cboScale.AddItem "10"
cboScale.AddItem "20"
cboScale.AddItem "30"
cboScale.AddItem "40"
cboScale.AddItem "50"
cboScale.AddItem "60"
cboScale.AddItem "1/8"
cboScale.AddItem "3/16"
cboScale.AddItem "1/4"
cboScale.AddItem "3/8"
cboScale.AddItem "1/2"
cboScale.AddItem "3/4"
'Initial value
cboScale.Value = "10"

cboBarrierSidewalk.AddItem "Barrier"
cboBarrierSidewalk.AddItem "Sidewalk"
'Initial value
cboBarrierSidewalk.Value = "Barrier"
txtSidewalkWidth.Text = "5.5"
txtParapetHeight.Text = "2.8333"

'Initialize values
txtStaPVI1.Text = "4603.65"
txtElevPVI1.Text = "328.43"
chkNoVC = False
txtLVC1.Text = "1150"
txtGrade1.Text = "-6.2465"
txtGrade2.Text = "3.4404"
chkPVI2.Value = True
txtGrade3.Text = "1.2131"
txtStaPVI2.Text = "5690.15"
txtLVC2.Text = "700"
txtWidth.Text = "38"
```

```
txtBeginBr.Text = "4825"
txtSkew.Text = "75"
cboSkewDir.ListIndex = 0
txtEndBentWidth.Text = "3"
txtBeamDepth = "5.25"
txtCapWidth.Text = "4"
txtCapDepth.Text = "4"
cboScale.Value = "30"
txtNumSpans.Text = "6"
txtSpan1.Text = "95"
txtSpan2.Text = "120"
txtSpan3.Text = "120"
txtSpan4.Text = "120"
txtSpan5.Text = "120"
txtSpan6.Text = "125"


End Sub

Private Sub cmdDraw_Click()
Dim SpanNo As Long
Dim NumSpans As Long
Dim staBack As Double
Dim staAhead As Double


NumSpans = Int(Val(txtNumSpans.Value))

DrawVC          'Draw vertical curve diagram using DrawVC sub function
DrawElevSpan1   'Draw Elevation View of Span 1
DrawPlanSpan1   'Draw Plan View of Span 1

staBack = txtBeginBr.Value
staAhead = staBack + Val(txtSpan1.Value)
SpanNo = 1

'After Span 1 has been drawn, now loop through all but the last span.
'The last span has its own code and is drawn after these If's.
If NumSpans > 2 Then                'Draw Span 2
    staBack = staAhead
    staAhead = staBack + Val(txtSpan2.Value)
    SpanNo = 2
    Call DrawPlanSpanInt(SpanNo, staBack, staAhead)
    Call DrawElevSpanInt(SpanNo, staBack, staAhead)
End If

If NumSpans > 3 Then                'Draw Span 3
    staBack = staAhead
    staAhead = staAhead + Val(txtSpan3.Value)
    SpanNo = 3
    Call DrawPlanSpanInt(SpanNo, staBack, staAhead)
    Call DrawElevSpanInt(SpanNo, staBack, staAhead)
End If

If NumSpans > 4 Then                'Draw Span 4
    staBack = staAhead
    staAhead = staAhead + Val(txtSpan4.Value)
    SpanNo = 4
    Call DrawPlanSpanInt(SpanNo, staBack, staAhead)
    Call DrawElevSpanInt(SpanNo, staBack, staAhead)
End If

If NumSpans > 5 Then                'Draw Span 5
    staBack = staAhead
```

```
    staAhead = staAhead + Val(txtSpan5.Value)
    SpanNo = 5
    Call DrawPlanSpanInt(SpanNo, staBack, staAhead)
    Call DrawElevSpanInt(SpanNo, staBack, staAhead)
End If

If NumSpans > 6 Then                'Draw Span 6
    staBack = staAhead
    staAhead = staAhead + Val(txtSpan7.Value)
    SpanNo = 6
    Call DrawPlanSpanInt(SpanNo, staBack, staAhead)
    Call DrawElevSpanInt(SpanNo, staBack, staAhead)
End If

If NumSpans > 7 Then                'Draw Span 7
    staBack = staAhead
    staAhead = staAhead + Val(txtSpan7.Value)
    SpanNo = 7
    Call DrawPlanSpanInt(SpanNo, staBack, staAhead)
    Call DrawElevSpanInt(SpanNo, staBack, staAhead)
End If

'Draw last span
staBack = staAhead
staAhead = Val(txtBeginBr.Value) + BrLength()
SpanNo = SpanNo + 1
Call DrawPlanSpanLast(SpanNo, staBack, staAhead)
Call DrawElevSpanLast(SpanNo, staBack, staAhead)


End Sub

Public Function elev(Sta As Double) As Double
 Dim x As Double 'Distance along VC 1
 Dim y As Double 'Distance along VC 2

Dim LVC As Double           'in feet
Dim LVC2 As Double          'in feet
Dim G1 As Double            'in decimal form
Dim G2 As Double            'in decimal form
Dim G3 As Double            'in feet
Dim ElevPVI As Double       'in feet
Dim ElevPVI2 As Double      'in feet
Dim StaPVI As Double        'in feet
Dim StaPVI2 As Double


StaPVI = Val(txtStaPVI1.Value)
ElevPVI = Val(txtElevPVI1.Value)

If chkNoVC.Value = True Then        'No vertical curve
    If txtGrade1.Value = "" Then    'Elevs based off Grade 2
        G2 = Val(txtGrade2.Value) / 100
        x = Sta - StaPVI
        elev = ElevPVI + G2 * x
        Exit Function
    Else                            'Elevs based off of Grade 1
        x = StaPVI - Sta
        G1 = Val(txtGrade1.Value) / 100
        elev = ElevPVI - G1 * x
        Exit Function
    End If
End If
```

```
'Calculate Elevation based on 1 vertical curve

G1 = Val(txtGrade1.Value) / 100
G2 = Val(txtGrade2.Value) / 100
LVC = Val(txtLVC1.Value)
x = Sta - StaPVI + LVC / 2

If x < 0 Then                          'Elev is before VC1 on Grade 1
    elev = ElevPVI - G1 * (StaPVI - Sta)
Else
    If x < LVC Then                    'Elev is on VC1
        elev = ElevPVI - G1 * LVC / 2 + G1 * x + x ^ 2 * (G2 - G1) / 2 / LVC
    Else                               'Elev is past VC on Grade 2
        elev = ElevPVI + G2 * (Sta - StaPVI)
    End If
End If

'Calculate Elevation based on 2 vertical curves
If chkPVI2.Value = True Then
    StaPVI2 = Val(txtStaPVI2.Value)
    LVC2 = Val(txtLVC2.Value)
    G3 = Val(txtGrade3.Value) / 100
    y = Sta - StaPVI2 + LVC2 / 2
    ElevPVI2 = ElevPVI + (StaPVI2 - StaPVI) * G2
    If y < 0 Then                      'Elev is between VC1 and VC2 on Grade 2
        Exit Function                  'Calculated Elev is already good
    Else
        If y < LVC2 Then               'Elev is on VC2
            elev = ElevPVI2 - G2 * LVC2 / 2 + G2 * y + y ^ 2 * (G3 - G2) / 2 / LVC2
        Else                           'Elev is past VC2 on Grade 3
            elev = ElevPVI2 + G3 * (Sta - StaPVI2)
        End If
    End If
End If

End Function

Private Sub DrawVC()
Dim ptOrigin As Point3d     'Drawing origin
Dim ptA As Point3d          'working point
Dim ptB As Point3d          'working point
Dim ptLow As Point3d        'keeps track of lowest point for 2 VC's
Dim G1 As Double            'Grade1 as a percent
Dim G2 As Double
Dim G3 As Double
Dim ElevPVI2 As Double      'To be calculated (not on User Form)
Dim pica As Double          'spacing unit (1/16th inch on sheet)
Dim linelength As Double    'length of grade lines in diagram (1.5 inch)
Dim stemlength As Double    'length of vertical stem (1 inch)
Dim gradelength As Double   'length of line under GRADE DATA text
Dim curvelength As Double   'length of line under VERTICAL CURVE DATA text
Dim angle As Double         'angle of grade line
Dim oLine As LineElement    'line element holder
Dim oText As TextElement    'text element holder
Dim strText As String       'string for text
Dim oMatrix As Matrix3d     'rotation matrix for placing text
Dim ptScale As Point3d      'scaling point for cells
Dim oCell As CellElement    'cell element holder



pica = 1 / 16 / 12
linelength = 0.125
stemlength = 1 / 12
```

```
gradelength = 2 / 12
curvelength = 3.5 / 12

'Text and line settings
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
ActiveSettings.TextStyle.Height = 2 * pica
ActiveSettings.TextStyle.Width = 2 * pica
SetLine ("Dimension")

'Settings for scale
ptScale = ActiveSettings.Scale


'Three different types drawings can be created:
'   1. Grade only
'      A. PVI line at start of grade line
'         i. Positive grade (stem on bottom)
'        ii. Negative grade (stem on top)
'      B. PVI line at end of grade line
'         i. Positive grade (stem on bottom)
'        ii. Negative grade (stem on top)
'   2. Single vertical curve (sags and crests are drawn differently)
'      A. Sag curve (stem on bottom)
'      B. Crest curve (stem on top)
'   3. Two vertical curves (add 2nd curve to single curve drawing)
'      A. 2nd curve is a sag (stem on bottom)
'      B. 2nd curve is a crest (stem on top)

'Set origin point for VC diagram
ptOrigin.x = 10
ptOrigin.y = 10
ptOrigin.Z = 0
ptA = ptOrigin
ptB = ptOrigin
ptScale = Point3dOne



If chkNoVC.Value = True Then        'No vertical curve
    If txtGrade1.Value = "" Then    'PVI line goes first
        G2 = Val(txtGrade2.Value)
            If G2 > 0 Then          'PVI line goes first on bottom
                angle = Atn(G2 / 100 / linelength) 'Calculate angle of line
                'Draw stem
                ptA.y = ptOrigin.y - stemlength
                Set oLine = CreateLineElement2(Nothing, ptOrigin, ptA)
                ActiveModelReference.AddElement oLine
                'Draw grade line
                ptB.x = ptOrigin.x + linelength * Cos(angle)
                ptB.y = ptOrigin.y + linelength * Sin(angle)
                Set oLine = CreateLineElement2(Nothing, ptOrigin, ptB)
                ActiveModelReference.AddElement oLine
                'Place grade text
                ptB.x = ptOrigin.x + 0.5 * linelength * Cos(angle) - pica *
Sin(angle)
                ptB.y = ptOrigin.y + 0.5 * linelength * Sin(angle) + pica *
Cos(angle)
                strText = ""
                If G2 > 0 Then strText = "+"
                strText = strText & Format(Str(G2), "###0.0000") & "%"
                oMatrix = Matrix3dFromAxisAndRotationAngle(2, angle)
                Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
                ActiveModelReference.AddElement oText
                'Place PVI arrow
```

```
                ptA.y = ptA.y + 4 * pica
                oMatrix = Matrix3dFromAxisAndRotationAngle(2, Pi)
                Set oCell = CreateCellElement2("AR8", ptA, ptScale, True, oMatrix)
                ActiveModelReference.AddElement oCell
                ptB.x = ptA.x + 4 * pica
                ptB.y = ptA.y
                Set oLine = CreateLineElement2(Nothing, ptA, ptB)
                ActiveModelReference.AddElement oLine
                'Place PVI sta and elev text on separate lines
                ptA.x = ptB.x + 2 * pica
                strText = "STA. " & Format(txtStaPVI1.Value, "###+##.00")
                ActiveSettings.TextStyle.Justification =
msdTextJustificationLeftCenter
                Set oText = CreateTextElement1(Nothing, strText, ptA,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                strText = "EL. " & Format(txtElevPVI1.Value, "###0.00")
                ptA.y = ptA.y - 3 * pica
                Set oText = CreateTextElement1(Nothing, strText, ptA,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                'Place title text 1/2" below bottom of vertical line
                ptB.x = ptOrigin.x + 0.5 * linelength * Cos(angle)
                ptB.y = ptOrigin.y - stemlength - 0.5 / 12
                strText = "GRADE DATA"
                ActiveSettings.TextStyle.Height = 3 * pica
                ActiveSettings.TextStyle.Width = 3 * pica
                ActiveSettings.TextStyle.Justification =
msdTextJustificationCenterBottom
                Set oText = CreateTextElement1(Nothing, strText, ptB,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                'Place 2" long line 1/16" inch under title text
                ptB.x = ptB.x + gradelength / 2
                ptB.y = ptB.y - pica
                ptA.x = ptB.x - gradelength
                ptA.y = ptB.y
                Set oLine = CreateLineElement2(Nothing, ptA, ptB)
                ActiveModelReference.AddElement oLine
            Else                        'PVI line goes first on top
                angle = Atn(G2 / 100 / linelength) 'Calculate angle of line
                'Draw stem
                ptA.y = ptOrigin.y + stemlength
                Set oLine = CreateLineElement2(Nothing, ptOrigin, ptA)
                ActiveModelReference.AddElement oLine
                'Draw grade line
                ptB.x = ptOrigin.x + linelength * Cos(angle)
                ptB.y = ptOrigin.y + linelength * Sin(angle)
                Set oLine = CreateLineElement2(Nothing, ptOrigin, ptB)
                ActiveModelReference.AddElement oLine
                'Place grade text
                ptB.x = ptOrigin.x + 0.5 * linelength * Cos(angle) + pica *
Sin(angle)
                ptB.y = ptOrigin.y + 0.5 * linelength * Sin(angle) + pica *
Cos(angle)
                strText = ""
                If G2 > 0 Then strText = "+"
                strText = strText & Format(Str(G2), "###0.0000") & "%"
                oMatrix = Matrix3dFromAxisAndRotationAngle(2, angle)
                Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
                ActiveModelReference.AddElement oText
                'Place PVI arrow
                ptA.y = ptA.y - 2 * pica
                oMatrix = Matrix3dFromAxisAndRotationAngle(2, Pi)
```

```
                Set oCell = CreateCellElement2("AR8", ptA, ptScale, True, oMatrix)
                ActiveModelReference.AddElement oCell
                ptB.x = ptA.x + 4 * pica
                ptB.y = ptA.y
                Set oLine = CreateLineElement2(Nothing, ptA, ptB)
                ActiveModelReference.AddElement oLine
                'Place PVI sta and elev text on separate lines
                ptA.x = ptB.x + 2 * pica
                strText = "STA. " & Format(txtStaPVI1.Value, "###+##.00")
                ActiveSettings.TextStyle.Justification =
msdTextJustificationLeftCenter
                Set oText = CreateTextElement1(Nothing, strText, ptA,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                strText = "EL. " & Format(txtElevPVI1.Value, "###0.00")
                ptA.y = ptA.y - 3 * pica
                Set oText = CreateTextElement1(Nothing, strText, ptA,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                'Place title text 1/2" below bottom of grade line
                ptB.x = ptOrigin.x + 0.5 * linelength * Cos(angle)
                ptB.y = ptOrigin.y + linelength * Sin(angle) - 0.5 / 12
                strText = "GRADE DATA"
                ActiveSettings.TextStyle.Height = 3 * pica
                ActiveSettings.TextStyle.Width = 3 * pica
                ActiveSettings.TextStyle.Justification =
msdTextJustificationCenterBottom
                Set oText = CreateTextElement1(Nothing, strText, ptB,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                'Place 2" long line 1/16" inch under title text
                ptB.x = ptB.x + gradelength / 2
                ptB.y = ptB.y - pica
                ptA.x = ptB.x - gradelength
                ptA.y = ptB.y
                Set oLine = CreateLineElement2(Nothing, ptA, ptB)
                ActiveModelReference.AddElement oLine
            End If
    Else                            'PVI line goes last
        G1 = Val(txtGrade1.Value)
        If G1 > 0 Then              'PVI line goes last and on top
                angle = Atn(G1 / 100 / linelength) 'Calculate angle of line
                'Draw stem
                ptA.y = ptOrigin.y + stemlength
                Set oLine = CreateLineElement2(Nothing, ptOrigin, ptA)
                ActiveModelReference.AddElement oLine
                'Draw grade line
                ptB.x = ptOrigin.x - linelength * Cos(angle)
                ptB.y = ptOrigin.y - linelength * Sin(angle)
                Set oLine = CreateLineElement2(Nothing, ptOrigin, ptB)
                ActiveModelReference.AddElement oLine
                'Place grade text
                ptB.x = ptOrigin.x - 0.5 * linelength * Cos(angle) - pica *
Sin(angle)
                ptB.y = ptOrigin.y - 0.5 * linelength * Sin(angle) + pica *
Cos(angle)
                strText = ""
                If G1 > 0 Then strText = "+"
                strText = strText & Format(Str(G1), "###0.0000") & "%"
                oMatrix = Matrix3dFromAxisAndRotationAngle(2, angle)
                Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
                ActiveModelReference.AddElement oText
                'Place PVI arrow
                ptA.y = ptA.y - 2 * pica
```

```
                Set oCell = CreateCellElement2("AR8", ptA, ptScale, True,
Matrix3dIdentity)
                ActiveModelReference.AddElement oCell
                ptB.x = ptA.x - 4 * pica
                ptB.y = ptA.y
                Set oLine = CreateLineElement2(Nothing, ptA, ptB)
                ActiveModelReference.AddElement oLine
                'Place PVI sta and elev text on separate lines
                ptA.x = ptB.x - 2 * pica
                strText = "STA. " & Format(txtStaPVI1.Value, "###+##.00")
                ActiveSettings.TextStyle.Justification =
msdTextJustificationRightCenter
                Set oText = CreateTextElement1(Nothing, strText, ptA,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                strText = "EL. " & Format(txtElevPVI1.Value, "###0.00")
                ptA.y = ptA.y - 3 * pica
                Set oText = CreateTextElement1(Nothing, strText, ptA,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                'Place title text 1/2" below bottom of grade line
                ptB.x = ptOrigin.x - 0.5 * linelength * Cos(angle)
                ptB.y = ptOrigin.y - linelength * Sin(angle) - 0.5 / 12
                strText = "GRADE DATA"
                ActiveSettings.TextStyle.Height = 3 * pica
                ActiveSettings.TextStyle.Width = 3 * pica
                ActiveSettings.TextStyle.Justification =
msdTextJustificationCenterBottom
                Set oText = CreateTextElement1(Nothing, strText, ptB,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                'Place 2" long line 1/16" inch under title text
                ptB.x = ptB.x + gradelength / 2
                ptB.y = ptB.y - pica
                ptA.x = ptB.x - gradelength
                ptA.y = ptB.y
                Set oLine = CreateLineElement2(Nothing, ptA, ptB)
                ActiveModelReference.AddElement oLine

        Else                                'PVI line goes last and on bottom
                angle = Atn(G1 / 100 / linelength) 'Calculate angle of line
                'Draw stem
                ptA.y = ptOrigin.y - stemlength
                Set oLine = CreateLineElement2(Nothing, ptOrigin, ptA)
                ActiveModelReference.AddElement oLine
                'Draw grade line
                ptB.x = ptOrigin.x - linelength * Cos(angle)
                ptB.y = ptOrigin.y - linelength * Sin(angle)
                Set oLine = CreateLineElement2(Nothing, ptOrigin, ptB)
                ActiveModelReference.AddElement oLine
                'Place grade text
                ptB.x = ptOrigin.x - 0.5 * linelength * Cos(angle) + pica *
Sin(angle)
                ptB.y = ptOrigin.y - 0.5 * linelength * Sin(angle) + pica *
Cos(angle)
                strText = ""
                If G1 > 0 Then strText = "+"
                strText = strText & Format(Str(G1), "###0.0000") & "%"
                oMatrix = Matrix3dFromAxisAndRotationAngle(2, angle)
                Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
                ActiveModelReference.AddElement oText
                'Place PVI arrow
                ptA.y = ptA.y + 4 * pica
```

```
                Set oCell = CreateCellElement2("AR8", ptA, ptScale, True,
Matrix3dIdentity)
                ActiveModelReference.AddElement oCell
                ptB.x = ptA.x - 4 * pica
                ptB.y = ptA.y
                Set oLine = CreateLineElement2(Nothing, ptA, ptB)
                ActiveModelReference.AddElement oLine
                'Place PVI sta and elev text on separate lines
                ptA.x = ptB.x - 2 * pica
                strText = "STA. " & Format(txtStaPVI1.Value, "###+##.00")
                ActiveSettings.TextStyle.Justification =
msdTextJustificationRightCenter
                Set oText = CreateTextElement1(Nothing, strText, ptA,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                strText = "EL. " & Format(txtElevPVI1.Value, "###0.00")
                ptA.y = ptA.y - 3 * pica
                Set oText = CreateTextElement1(Nothing, strText, ptA,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                'Place title text 1/2" below bottom of grade line
                ptB.x = ptOrigin.x - 0.5 * linelength * Cos(angle)
                ptB.y = ptOrigin.y - stemlength - 0.5 / 12
                strText = "GRADE DATA"
                ActiveSettings.TextStyle.Height = 3 * pica
                ActiveSettings.TextStyle.Width = 3 * pica
                ActiveSettings.TextStyle.Justification =
msdTextJustificationCenterBottom
                Set oText = CreateTextElement1(Nothing, strText, ptB,
Matrix3dIdentity)
                ActiveModelReference.AddElement oText
                'Place 2" long line 1/16" inch under title text
                ptB.x = ptB.x + gradelength / 2
                ptB.y = ptB.y - pica
                ptA.x = ptB.x - gradelength
                ptA.y = ptB.y
                Set oLine = CreateLineElement2(Nothing, ptA, ptB)
                ActiveModelReference.AddElement oLine
        End If
    End If
Exit Sub
End If



'Whether there are 2 VC's or 1, the drawing starts out the same:
G1 = Val(txtGrade1.Value)
G2 = Val(txtGrade2.Value)
If G2 - G1 > 0 Then             'Sag curve; PVI line goes on bottom
    'Draw stem
    ptA.y = ptOrigin.y - stemlength
    Set oLine = CreateLineElement2(Nothing, ptOrigin, ptA)
    ActiveModelReference.AddElement oLine
    'Draw back grade line
    angle = Atn(G1 / 100 / linelength) 'Calculate angle of back grade
    ptB.x = ptOrigin.x - linelength * Cos(angle)
    ptB.y = ptOrigin.y - linelength * Sin(angle)
    Set oLine = CreateLineElement2(Nothing, ptOrigin, ptB)
    ActiveModelReference.AddElement oLine
    'Place back grade text
    ptB.x = ptOrigin.x - 0.5 * linelength * Cos(angle)
    ptB.y = ptOrigin.y - 0.5 * linelength * Sin(angle) + pica
    strText = ""
    If G1 > 0 Then strText = "+"
```

```
    strText = strText & Format(Str(G1), "###0.0000") & "%"
    oMatrix = Matrix3dFromAxisAndRotationAngle(2, angle)
    Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
    ActiveModelReference.AddElement oText
    'Draw ahead grade line
    angle = Atn(G2 / 100 / linelength) 'Calculate angle of line
    ptB.x = ptOrigin.x + linelength * Cos(angle)
    ptB.y = ptOrigin.y + linelength * Sin(angle)
    Set oLine = CreateLineElement2(Nothing, ptOrigin, ptB)
    ActiveModelReference.AddElement oLine
    'Place ahead grade text
    ptB.x = ptOrigin.x + 0.5 * linelength * Cos(angle)
    ptB.y = ptOrigin.y + 0.5 * linelength * Sin(angle) + pica
    strText = ""
    If G2 > 0 Then strText = "+"
    strText = strText & Format(Str(G2), "###0.0000") & "%"
    oMatrix = Matrix3dFromAxisAndRotationAngle(2, angle)
    Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
    ActiveModelReference.AddElement oText
    'Place PVI arrow
    ptA.y = ptA.y + 4 * pica
    Set oCell = CreateCellElement2("AR8", ptA, ptScale, True, Matrix3dIdentity)
    ActiveModelReference.AddElement oCell
    ptB.x = ptA.x - 4 * pica
    ptB.y = ptA.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    'Place PVI sta and elev text on separate lines
    ptA.x = ptB.x - 2 * pica
    strText = "STA. " & Format(txtStaPVI1.Value, "###+##.00")
    ActiveSettings.TextStyle.Justification = msdTextJustificationRightCenter
    Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
    ActiveModelReference.AddElement oText
    strText = "EL. " & Format(txtElevPVI1.Value, "###0.00")
    ptA.y = ptA.y - 3 * pica
    Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
    ActiveModelReference.AddElement oText
    'Place LVC info 1/2" below bottom of stem
    ptB.x = ptOrigin.x
    ptB.y = ptOrigin.y - stemlength - 10 * pica
    strText = Format(txtLVC1.Value, "####") & " FT. V.C."
    ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
    Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
    ActiveModelReference.AddElement oText

Else                            'Crest curve; PVI line goes on top
    'Draw stem
    ptA.y = ptOrigin.y + stemlength
    Set oLine = CreateLineElement2(Nothing, ptOrigin, ptA)
    ActiveModelReference.AddElement oLine
    'Draw back grade line
    angle = Atn(G1 / 100 / linelength) 'Calculate angle of line
    ptB.x = ptOrigin.x - linelength * Cos(angle)
    ptB.y = ptOrigin.y - linelength * Sin(angle)
    Set oLine = CreateLineElement2(Nothing, ptOrigin, ptB)
    ActiveModelReference.AddElement oLine
    'Place back grade text
    ptB.x = ptOrigin.x - 0.5 * linelength * Cos(angle)
    ptB.y = ptOrigin.y - 0.5 * linelength * Sin(angle) + pica
    strText = ""
    If G1 > 0 Then strText = "+"
    strText = strText & Format(Str(G1), "###0.0000") & "%"
    oMatrix = Matrix3dFromAxisAndRotationAngle(2, angle)
    Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
```

```
    ActiveModelReference.AddElement oText
    'Draw ahead grade line
    angle = Atn(G2 / 100 / linelength) 'Calculate angle of line
    ptB.x = ptOrigin.x + linelength * Cos(angle)
    ptB.y = ptOrigin.y + linelength * Sin(angle)
    Set oLine = CreateLineElement2(Nothing, ptOrigin, ptB)
    ActiveModelReference.AddElement oLine
    'Place ahead grade text
    ptB.x = ptOrigin.x + 0.5 * linelength * Cos(angle)
    ptB.y = ptOrigin.y + 0.5 * linelength * Sin(angle) + pica
    strText = ""
    If G2 > 0 Then strText = "+"
    strText = strText & Format(Str(G2), "###0.0000") & "%"
    oMatrix = Matrix3dFromAxisAndRotationAngle(2, angle)
    Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
    ActiveModelReference.AddElement oText
    'Place PVI arrow
    ptA.y = ptA.y - 2 * pica
    Set oCell = CreateCellElement2("AR8", ptA, ptScale, True, Matrix3dIdentity)
    ActiveModelReference.AddElement oCell
    ptB.x = ptA.x - 4 * pica
    ptB.y = ptA.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    'Place PVI sta and elev text on separate lines
    ptA.x = ptB.x - 2 * pica
    strText = "STA. " & Format(txtStaPVI1.Value, "###+##.00")
    ActiveSettings.TextStyle.Justification = msdTextJustificationRightCenter
    Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
    ActiveModelReference.AddElement oText
    strText = "EL. " & Format(txtElevPVI1.Value, "###0.00")
    ptA.y = ptA.y - 3 * pica
    Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
    ActiveModelReference.AddElement oText
    'Place LVC info 1/2" below stem
    ptB.x = ptOrigin.x
    ptB.y = ptOrigin.y - 10 * pica
    strText = Format(txtLVC1.Value, "####") & " FT. V.C."
    ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
    Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
    ActiveModelReference.AddElement oText
End If



If chkPVI2.Value = True Then            'Two vertical curves
    'Calculate Elev of PVI2 since it is not given on User Form
    ElevPVI2 = Val(txtElevPVI1.Value) + _
        G2 * (Val(txtStaPVI2.Value) - Val(txtStaPVI1.Value)) / 100
    'Store Y value of LVC info for later to determine Y of title text
    ptLow = ptB
    'Reset origin to end of previous grade
    angle = Atn(G2 / 100 / linelength) 'Calculate angle of Grade2
    ptOrigin.x = ptOrigin.x + linelength * Cos(angle)
    ptOrigin.y = ptOrigin.y + linelength * Sin(angle)
    'Draw ahead grade line
    G3 = Val(txtGrade3.Value)
    angle = Atn(G3 / 100 / linelength) 'Calculate angle of Grade 3
    ptB.x = ptOrigin.x + linelength * Cos(angle)
    ptB.y = ptOrigin.y + linelength * Sin(angle)
    Set oLine = CreateLineElement2(Nothing, ptOrigin, ptB)
    ActiveModelReference.AddElement oLine
    'Place ahead grade text
    ptB.x = ptOrigin.x + 0.5 * linelength * Cos(angle)
```

```
ptB.y = ptOrigin.y + 0.5 * linelength * Sin(angle) + pica
strText = ""
If G3 > 0 Then strText = "+"
strText = strText & Format(Str(G3), "###0.0000") & "%"
oMatrix = Matrix3dFromAxisAndRotationAngle(2, angle)
Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
ActiveModelReference.AddElement oText

If G3 - G2 > 0 Then             'Sag curve; PVI line goes on bottom
    'Draw stem
    ptA.x = ptOrigin.x
    ptA.y = ptOrigin.y - stemlength
    Set oLine = CreateLineElement2(Nothing, ptOrigin, ptA)
    ActiveModelReference.AddElement oLine
    'Place PVI arrow
    ptA.y = ptA.y + 4 * pica
    oMatrix = Matrix3dFromAxisAndRotationAngle(2, Pi)
    Set oCell = CreateCellElement2("AR8", ptA, ptScale, True, oMatrix)
    ActiveModelReference.AddElement oCell
    ptB.x = ptA.x + 4 * pica
    ptB.y = ptA.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    'Place PVI sta and elev text on separate lines
    ptA.x = ptB.x + 2 * pica
    strText = "STA. " & Format(txtStaPVI2.Value, "###+##.00")
    ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
    Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
    ActiveModelReference.AddElement oText
    strText = "EL. " & Format(ElevPVI2, "###0.00")
    ptA.y = ptA.y - 3 * pica
    Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
    ActiveModelReference.AddElement oText
    'Place LVC info 1/2" below bottom of stem
    ptB.x = ptOrigin.x
    ptB.y = ptOrigin.y - stemlength - 10 * pica
    strText = Format(txtLVC2.Value, "####") & " FT. V.C."
    ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
    Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
    ActiveModelReference.AddElement oText

Else                            'Crest curve; PVI line goes on top
    'Draw stem
    ptA.x = ptOrigin.x
    ptA.y = ptOrigin.y + stemlength
    Set oLine = CreateLineElement2(Nothing, ptOrigin, ptA)
    ActiveModelReference.AddElement oLine
    'Place PVI arrow
    ptA.y = ptA.y - 2 * pica
    oMatrix = Matrix3dFromAxisAndRotationAngle(2, Pi)
    Set oCell = CreateCellElement2("AR8", ptA, ptScale, True, oMatrix)
    ActiveModelReference.AddElement oCell
    ptB.x = ptA.x + 4 * pica
    ptB.y = ptA.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    'Place PVI sta and elev text on separate lines
    ptA.x = ptB.x + 2 * pica
    strText = "STA. " & Format(txtStaPVI2.Value, "###+##.00")
    ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
    Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
    ActiveModelReference.AddElement oText
    strText = "EL. " & Format(ElevPVI2, "###0.00")
    ptA.y = ptA.y - 3 * pica
```

```
        Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
        ActiveModelReference.AddElement oText
        'Place LVC info 1/2" below stem
        ptB.x = ptOrigin.x
        ptB.y = ptOrigin.y - 10 * pica
        strText = Format(txtLVC2.Value, "####") & " FT. V.C."
        ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
        Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
        ActiveModelReference.AddElement oText
    End If
    'Set center of title text for 2 VC's
    ptB.x = ptOrigin.x - 0.5 * linelength
    'Should Y of title text be based on VC1 or VC2?
    If ptLow.y < ptB.y Then ptB.y = ptLow.y
End If

'Place title text 1/2" below bottom of grade line
ptB.y = ptB.y - 11 * pica
strText = "VERTICAL CURVE DATA"
ActiveSettings.TextStyle.Height = 3 * pica
ActiveSettings.TextStyle.Width = 3 * pica
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText
'Place 3.5" long line 1/16" inch under title text
ptB.x = ptB.x + curvelength / 2
ptB.y = ptB.y - pica
ptA.x = ptB.x - curvelength
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine


End Sub

Private Sub SetLine(strType As String)

Dim oLevel As Level
Dim oLineStyle As LineStyle

Select Case strType
    Case "Concrete"
        Set oLevel = ActiveDesignFile.Levels.Find("Level 2")
        Set ActiveSettings.Level = oLevel
        ActiveSettings.LineWeight = 4
        Set oLineStyle = ActiveDesignFile.LineStyles.Find("0")
        Set ActiveSettings.LineStyle = oLineStyle
        ActiveSettings.Color = 0 'White

    Case "ConcHid" 'for beams on deck plan
        Set oLevel = ActiveDesignFile.Levels.Find("Level 3")
        Set ActiveSettings.Level = oLevel
        ActiveSettings.LineWeight = 3
        Set oLineStyle = ActiveDesignFile.LineStyles.Find("3")
        Set ActiveSettings.LineStyle = oLineStyle
        ActiveSettings.Color = 6 'Orange

    Case "Center"
        Set oLevel = ActiveDesignFile.Levels.Find("Level 5")
        Set ActiveSettings.Level = oLevel
        ActiveSettings.LineWeight = 1
        Set oLineStyle = ActiveDesignFile.LineStyles.Find("6")
        Set ActiveSettings.LineStyle = oLineStyle
        ActiveSettings.Color = 2 'Lime
```

```
    Case "Dimension"
        Set oLevel = ActiveDesignFile.Levels.Find("Level 7")
        Set ActiveSettings.Level = oLevel
        ActiveSettings.LineWeight = 1
        Set oLineStyle = ActiveDesignFile.LineStyles.Find("0")
        Set ActiveSettings.LineStyle = oLineStyle
        ActiveSettings.Color = 3 'Red

    Case "Ground"
        Set oLevel = ActiveDesignFile.Levels.Find("Level 13")
        Set ActiveSettings.Level = oLevel
        ActiveSettings.LineWeight = 2
        Set oLineStyle = ActiveDesignFile.LineStyles.Find("0")
        Set ActiveSettings.LineStyle = oLineStyle
        ActiveSettings.Color = 2 'Lime

    Case "Rebar"
        Set oLevel = ActiveDesignFile.Levels.Find("Level 4")
        Set ActiveSettings.Level = oLevel
        ActiveSettings.LineWeight = 2
        Set oLineStyle = ActiveDesignFile.LineStyles.Find("0")
        Set ActiveSettings.LineStyle = oLineStyle
        ActiveSettings.Color = 1 'Blue
    Case Else
        MsgBox "No such line type for " & strType
    End Select
End Sub

Private Sub SetText(strType As String)

Dim pica As Double
pica = Val(txtScale.Value) / 192

Select Case strType
    Case "Normal"
        ActiveSettings.TextStyle.Height = 2 * pica
        ActiveSettings.TextStyle.Width = 2 * pica
        ActiveSettings.LineWeight = 2
        ActiveSettings.Color = 3 'Red
    Case "Title"
        ActiveSettings.TextStyle.Height = 3 * pica
        ActiveSettings.TextStyle.Width = 3 * pica
        ActiveSettings.LineWeight = 2
        ActiveSettings.Color = 3 'Red
    Case Else
        MsgBox "No type of text called " & strType
End Select

End Sub

Public Sub DrawElevSpan1()

Dim skew As Double          'Skew angle (in radians; if backwards, supplement of
skew)
Dim ptA As Point3d          'Working points
Dim ptB As Point3d
Dim ptC As Point3d
Dim ptOrigin As Point3d     'Origin point
Dim StaAh As Double
Dim StaBk As Double
Dim pica As Double          'Scaled spacing constant (1/16th inch on sheet)
Dim hBeam As Double         'Height of beam
Dim hSlab As Double         'Height of slab
Dim wSidewalk As Double     'Width of sidewalk
```

```
Dim wEndBent As Double      'Width of end bent cap
Dim lEndPost As Double      'Length of end post (calculated)
Dim hEndPost As Double      'Height of end post
Dim hParapet As Double      'Height of parapet above sidewalk
Dim hWingCurb As Double     'height of wingwall above grade (10")
Dim hBarrier As Double      'height of barrier (2'-8")
Dim hPileBox As Double      'height of pile box (2')
Dim lPileBox As Double      'length of pile box (2'-6")
Dim lWing As Double         'length of wingwall (calculated)

Dim oLine As LineElement    'line holder
Dim oCell As CellElement    'cell holder
Dim oText As TextElement    'text holder
Dim strText As String       'working string
Dim oMatrix As Matrix3d     'rotation matrix holder for cells and text
Dim oCircle As EllipseElement   'ellipse holder


ptOrigin.x = Val(txtBeginBr.Value)
ptOrigin.y = elev(ptOrigin.x)
ptOrigin.Z = 0
ptA = ptOrigin
ptB = ptOrigin
ptC = ptOrigin

'Assume some values
hSlab = 8 / 12
hPileBox = 2
lPileBox = 2.5
hBarrier = 32 / 12
hWingCurb = 10 / 12
lEndPost = 4
hEndPost = hBarrier

'Get values from form
hBeam = Val(txtBeamDepth.Value)
wEndBent = Val(txtEndBentWidth.Value)
skew = Pi * Val(txtSkew.Value) / 180
If Left(cboSkewDir.Value, 1) = "B" Then skew = Pi - skew
pica = Val(txtScale.Value) / 192
If cboBarrierSidewalk.Value = "Sidewalk" Then
    wSidewalk = Val(txtSidewalkWidth.Value)
    hParapet = Val(txtParapetHeight.Value)
    If hParapet = 2.25 Then
        hEndPost = 3.5 + 0.5 + 0.03 * wSidewalk
        hBarrier = 2.25 + 0.5 + 0.03 * wSidewalk
    Else
        hBarrier = hParapet + 0.5 + 0.03 * wSidewalk
        hEndPost = hBarrier
    End If
End If

'Calculate wingwall length based on beam height and skew, rounded to nearest 3"
lWing = Round(8 * (hSlab + hBeam + 2 / 12 + 1) / Sin(skew)) / 4

'Draw wing
SetLine ("Concrete")
ptA.x = ptOrigin.x + Val(txtEndBentWidth.Value) / Sin(skew)
ptA.y = elev(ptA.x) + hWingCurb
ptB.x = ptA.x
ptB.y = ptA.y - hWingCurb - hSlab - hBeam - 2
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw endslope (10' horizontal slope; vertical adjusted for skew)
```

```
SetLine ("Ground")
ptC.x = ptB.x + 2 / Sin(skew)
ptC.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptB, ptC)
ActiveModelReference.AddElement oLine
ptA.x = ptC.x + 20
ptA.y = ptC.y - 10 * Sin(skew)
Set oLine = CreateLineElement2(Nothing, ptC, ptA)
ActiveModelReference.AddElement oLine
'Draw 2:1 slope box with 3/8" horizontal line and 3/16" vertical
SetLine ("Dimension")
ptA.x = ptA.x - 2 * pica / Sin(skew) / Sin(Radians(26.565))
ptC.x = ptA.x - 6 * pica
ptC.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptC, ptA)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x - 3 * pica
ptA.y = ptA.y - pica
strText = "2"
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterTop
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.x = ptA.x - 4 * pica
ptA.y = ptA.y + pica
strText = "1"
ActiveSettings.TextStyle.Justification = msdTextJustificationRightBottom
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.x = ptA.x + 2 * pica
ptA.y = ptA.y + pica
strText = "*"
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftBottom
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.x = ptC.x
ptA.y = ptC.y + 3 * pica
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptC, ptA)
ActiveModelReference.AddElement oLine
'Back to drawing wing
SetLine ("Concrete")
ptA.x = ptB.x - lWing
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'If wing is longer than 11', draw pile box
If lWing > 11 Then
    ptA.y = ptA.y + hPileBox
    ptB.x = ptA.x + lPileBox
    ptB.y = ptB.y + hPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x + lPileBox
    ptA.y = ptA.y - hPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x - lPileBox
End If
'Now back to drawing the wing
ptB.x = ptA.x
ptB.y = elev(ptB.x) + hWingCurb
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
```

```
ptA.x = ptA.x + lWing
ptA.y = elev(ptA.x) + hWingCurb
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw End Post
ptB.x = ptA.x
ptB.y = elev(ptB.x) + hEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x - lEndPost
ptA.y = elev(ptA.x) + hEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = elev(ptB.x) + hWingCurb
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw Span 1 (does not include intermediate bent)
ptB.x = ptOrigin.x + Val(txtEndBentWidth.Text) / Sin(skew)
ptB.y = elev(ptB.x)
ptA.x = ptOrigin.x + Val(txtSpan1.Text)
ptA.y = elev(ptA.x)
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y - hSlab
ptB.y = ptB.y - hSlab
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y - hBeam
ptB.y = ptB.y - hBeam
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y + hBeam + hSlab + hBarrier
ptB.y = ptB.y + hBeam + hSlab + hBarrier
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw Station and Elev Marker
SetLine ("Dimension")
ptA.x = ptOrigin.x
ptA.y = ptOrigin.y + hBarrier + 2 * pica
ptB.x = ptA.x
ptB.y = ptA.y + 24 * pica
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Add STA text above line and EL text below
ptA.x = ptA.x - pica
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftBottom
strText = "STA. " & Format(ptOrigin.x, "###+##.00")
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Pi / 2)
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText
ptA.x = ptA.x + 2 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftTop
strText = "EL. " & Format(elev(ptB.x), "##0.00")
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText
strText = "**"
ptB.y = ptB.y + 2 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
```

```
ActiveModelReference.AddElement oText
strText = "*  "
ptB.y = ptB.y + 14 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationRightBottom
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText
strText = "2:1 SLOPE MEASURED NORMAL TO END BENT."
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftBottom
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptB.y = ptB.y - 3 * pica
strText = "**  "
ActiveSettings.TextStyle.Justification = msdTextJustificationRightBottom
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText
strText = "STATIONS AND ELEVATIONS ARE ALONG PROFILE GRADE LINE AT THE"
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftBottom
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptB.y = ptB.y - 3 * pica
strText = "INTERSECTION OF PROFILE GRADE LINE AND B.F.P.R. OR c BENTS."
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftBottom
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText
'Add Bent No in circle 0.5 in below wing
SetLine ("Dimension")
ptB.y = ptOrigin.y - hSlab - hBeam - 2 - 8 * pica
ActiveModelReference.AddElement oText
Set oCircle = CreateEllipseElement2(Nothing, ptB, 2 * pica, 2 * pica,
Matrix3dIdentity)
ActiveModelReference.AddElement oCircle
strText = "1"
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText
End Sub

Public Sub DrawPlanSpan1()

Dim skew As Double          'Skew angle (in radians; if backwards, supplement of
skew)
Dim ptA As Point3d          'Working points
Dim ptB As Point3d
Dim ptC As Point3d
Dim ptOrigin As Point3d     'Origin point
Dim ptScale As Point3d
Dim StaAh As Double
Dim StaBk As Double
Dim pica As Double          'Scaled spacing constant (1/16th inch on sheet)
Dim dScale As Double        'Scale factor for placing arrowheads
Dim dAngle As Double        'Angle for callouts (45 degrees)
Dim offset As Double        'offset from centerline
Dim longpost As Double      'length of longer end post
Dim shortpost As Double     'length of shorter end post
Dim xLeftPost As Double     'X coordinate of begin left post
Dim xRightPost As Double    'X coordinate of begin right post

Dim wBridge As Double       'width of Bridge, gutter to gutter
Dim hSlab As Double         'height of slab (for wingwall length calc)
Dim hBeam As Double         'height of beam (or wingwall length calc)
Dim wSlot As Double         'width between back of end post and face of wing (4")
Dim wLip As Double          'width of lip behind barrier (1.5")
Dim wEndBent As Double      'width of end bent cap
```

```
Dim lEndPost As Double      'length of end post (could be calculated?)
Dim wEndPost As Double      'width of End Post (1'-1")
Dim wPaveRest As Double     'width of paving rest (8")
Dim wBarrier As Double      'width of barrier (1'-6")
Dim wBarrTop As Double      'width of top of barrier (9")
Dim wSidewalk As Double     'width of sidewalk
Dim lTrans As Double        'length of barrier transition (10')
Dim wPileBox As Double      'outside of pile box to face of wing (9")
Dim lPileBox As Double      'length of pile box (2'-6")
Dim wWing As Double         'width of wingwall (1')
Dim lWing As Double         'length of wingwall (calculated)
Dim lSpan As Double         'length of Span 1

Dim oLine As LineElement    'line holder
Dim oCell As CellElement    'cell holder
Dim oText As TextElement    'text holder
Dim strText As String       'working string for dimensions and notes
Dim strCell As String       'working string for cell names
Dim oMatrix As Matrix3d     'rotation matrix holder for cells and text
Dim oCircle As EllipseElement   'ellipse holder


'Assume some values
wBarrier = 1.5
wBarrTop = 0.75
wLip = 1.5 / 12
wSlot = 4 / 12
wEndPost = 13 / 12
lEndPost = 4
wPaveRest = 8 / 12
wWing = 1
wPileBox = 0.75
lPileBox = 2.5
lTrans = 10
hSlab = 8 / 12
wSidewalk = 0

'Get values from form
wBridge = Val(txtWidth.Value)
wEndBent = Val(txtEndBentWidth.Value)
hBeam = Val(txtBeamDepth.Value)
skew = Pi * Val(txtSkew.Value) / 180
If Left(cboSkewDir.Value, 1) = "B" Then skew = Pi - skew
pica = Val(txtScale.Value) / 192
lSpan = Val(txtSpan1.Value)
If cboBarrierSidewalk.Value = "Sidewalk" Then
    wSidewalk = Val(txtSidewalkWidth.Value)
    wBarrier = 13 / 12
    wBarrTop = 13 / 12
End If

'Place origin at begin bridge at 5.5" plus half of bridge width above plan
ptOrigin.x = Val(txtBeginBr.Value)
ptOrigin.y = elev(ptOrigin.x) + 88 * pica + 0.5 * wBridge + wSidewalk + wBarrier
ptOrigin.Z = 0
ptA = ptOrigin
ptB = ptOrigin
ptC = ptOrigin

'Define scaling point for placement of cells
ptScale.x = Val(txtScale.Value)
ptScale.y = Val(txtScale.Value)
ptScale.Z = 1
strCell = "AR8"
```

```
CellExists (strCell)

'Calculate wingwall length based on beam height and skew, rounded to nearest 3"
lWing = Round(8 * (hSlab + hBeam + 2 / 12 + 1) / Sin(skew)) / 4

'Calculate length of end posts
longpost = Round(4 * (0.125 + (wPaveRest + wEndBent + 2 / 12) / Sin(skew) + _
    (wEndPost + wSlot) / Abs(Tan(skew)))) / 4
If longpost < 4 Then longpost = 4
shortpost = Round(4 * (0.125 + (wPaveRest + wEndBent + 2 / 12) / Sin(skew) - _
    (wEndPost + wSlot) / Abs(Tan(skew)))) / 4
If shortpost < 4 Then shortpost = 4

'Draw BFPR and paving rest
offset = 0.5 * wBridge + wSidewalk + wBarrier - wBarrTop
ptA.x = ptOrigin.x - offset / Tan(skew)
ptA.y = ptOrigin.y - offset
ptB.x = ptOrigin.x + offset / Tan(skew)
ptB.y = ptOrigin.y + offset
SetLine ("Concrete")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x - wPaveRest / Sin(skew)
ptB.x = ptB.x - wPaveRest / Sin(skew)
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw right wing wall
offset = offset + wEndPost + wSlot
ptA.x = ptOrigin.x - offset / Tan(skew) + wEndBent / Sin(skew)
ptA.y = ptOrigin.y - offset
ptB.x = ptA.x
ptB.y = ptA.y - wWing
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x - lWing
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = ptB.y + wWing
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'If wing is longer than 11', draw pile box
If lWing > 11 Then
    'Draw outside portion of box
    ptC.x = ptA.x
    ptC.y = ptA.y - wPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x + lPileBox
    ptA.y = ptC.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptC.x = ptA.x
    ptC.y = ptC.y + wPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    'Draw inner portion of box
    ptC.x = ptB.x
    ptC.y = ptB.y + wPileBox
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    ptA.y = ptC.y
```

```
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptC.x = ptC.x + lPileBox
    ptC.y = ptB.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
End If
'Draw last piece of wing
ptA.x = ptB.x + lWing
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptB, ptA)
ActiveModelReference.AddElement oLine

'Draw right End Post
ptB.x = ptA.x
ptB.y = ptA.y + wSlot + wEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'If forward skew, use short endpost, otherwise long endpost
lEndPost = shortpost
If skew > 0.5 * Pi Then lEndPost = longpost
ptA.x = ptA.x - lEndPost
'Store location of right post for a dimension line later on
xRightPost = ptA.x
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = ptA.y - wEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + lEndPost
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw right barrier
offset = 0.5 * wBridge          'face of sidewalk or barrier
ptA.y = ptA.y + wEndPost        'ptA is at end of inside end of end post
ptC.x = ptOrigin.x + lSpan - offset / Tan(skew)
ptC.y = ptOrigin.y - offset
If wSidewalk = 0 Then
    'Draw transition
    ptB.x = ptA.x + 10
    ptB.y = ptA.y + wBarrier - wBarrTop
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    'Put ptC at end of span at top inside face of barrier
    ptC.y = ptC.y - wBarrier + wBarrTop
Else
    'Draw gutter at face of sidewalk
    ptB.x = ptOrigin.x - offset / Tan(skew)
    ptB.y = ptOrigin.y - offset
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    'Put ptC at end of span at inside face of parapet
    ptC.x = ptC.x - wSidewalk / Tan(skew)
    ptC.y = ptC.y - wSidewalk
End If
'Draw inside top of rail
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
```

```
'Draw back of barrier
ptA.y = ptA.y - wBarrTop
ptC.y = ptC.y - wBarrTop
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
'Draw edge of deck
ptA.y = ptA.y - wLip
ptC.y = ptC.y - wLip
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine


'Draw left wing (change signs on delta y's)
offset = 0.5 * wBridge + wBarrier - wBarrTop + wEndPost + wSlot
If cboBarrierSidewalk.Value = "Sidewalk" Then offset = offset + wSidewalk
ptA.x = ptOrigin.x + offset / Tan(skew) + wEndBent / Sin(skew)
ptA.y = ptOrigin.y + offset
ptB.x = ptA.x
ptB.y = ptA.y + wWing
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x - lWing
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = ptB.y - wWing
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'If wing is longer than 11', draw pile box
If lWing > 11 Then
    'Draw outside portion of box
    ptC.x = ptA.x
    ptC.y = ptA.y + wPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x + lPileBox
    ptA.y = ptC.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptC.x = ptA.x
    ptC.y = ptC.y - wPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    'Draw inner portion of box
    ptC.x = ptB.x
    ptC.y = ptB.y - wPileBox
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    ptA.y = ptC.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptC.x = ptC.x + lPileBox
    ptC.y = ptB.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
End If
'Draw last piece of wing
ptA.x = ptB.x + lWing
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptB, ptA)
ActiveModelReference.AddElement oLine

'Draw left End Post
```

```
ptB.x = ptA.x
ptB.y = ptA.y - wSlot - wEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'If backward skew, use short endpost, otherwise long endpost
lEndPost = shortpost
If skew < 0.5 * Pi Then lEndPost = longpost
ptA.x = ptA.x - lEndPost
'Store location of end post for dimensioning later on
xLeftPost = ptA.x
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = ptA.y + wEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + lEndPost
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw left barrier (change sign of y offsets)
offset = 0.5 * wBridge          'face of sidewalk or barrier
ptA.y = ptA.y - wEndPost        'ptA is at end of inside end of end post
ptC.x = ptOrigin.x + lSpan + offset / Tan(skew)
ptC.y = ptOrigin.y + offset
If wSidewalk = 0 Then
    'Draw transition
    ptB.x = ptA.x + 10
    ptB.y = ptA.y - wBarrier + wBarrTop
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    'Put ptC at end of span at top inside face of barrier
    ptC.y = ptC.y + wBarrier - wBarrTop
Else
    'Draw gutter at face of sidewalk
    ptB.x = ptOrigin.x + offset / Tan(skew)
    ptB.y = ptOrigin.y + offset
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    'Put ptC at end of span at inside face of parapet
    ptC.x = ptC.x + wSidewalk / Tan(skew)
    ptC.y = ptC.y + wSidewalk
End If
'Draw inside top of rail
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
'Draw back of barrier
ptA.y = ptA.y + wBarrTop
ptC.y = ptC.y + wBarrTop
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
'Draw edge of deck
ptA.y = ptA.y + wLip
ptC.y = ptC.y + wLip
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine

'Draw bridge width dimensions at beginning of bridge
'Short extension lines end 1" from end of end post
'Long extension lines end 1.5" from end of end post (gutter-to-gutter only)
```

```
'Start at left outside edge of bridge and draw extensions lines from
'ptA near the bridge to ptB away from the bridge
SetLine ("Dimension")
'Short extension line for left edge of deck
ptA.x = xLeftPost - pica
ptA.y = ptOrigin.y + 0.5 * wBridge + wSidewalk + wBarrier + wLip
'Set end point based on which side of bridge is furthest back
ptB.x = xRightPost - 26 * pica
If skew > 0.5 * Pi Then ptB.x = xLeftPost - 26 * pica
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Place arrowhead for left outside edge of bridge
ptB.x = ptB.x + 2 * pica
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(-90))
Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
'Place callout
'Draw barrier dimension on left
ptA.x = ptB.x
ptA.y = ptB.y + 6 * pica
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptC.x = ptA.x + 4 * pica
ptC.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
ptC.x = ptC.x + pica
strText = strDim(wBarrier + wLip)
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
Set oText = CreateTextElement1(Nothing, strText, ptC, Matrix3dIdentity)
ActiveModelReference.AddElement oText

'Short extension line for right edge of deck
ptA.y = ptOrigin.y - 0.5 * wBridge - wSidewalk - wBarrier - wLip
ptA.x = xRightPost - pica
ptB.x = ptB.x - 2 * pica
ptB.y = ptA.y
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Place arrowhead on bottom right
ptB.x = ptB.x + 2 * pica
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Pi / 2)
Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
'Draw barrier dimension on right
ptA.x = ptB.x
ptA.y = ptB.y - 6 * pica
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptC.x = ptA.x + 4 * pica
ptC.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
ptC.x = ptC.x + pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
strText = strDim(wBarrier + wLip)
SetText ("Normal")
Set oText = CreateTextElement1(Nothing, strText, ptC, Matrix3dIdentity)
ActiveModelReference.AddElement oText
```

```
'Gutter to Gutter dimension is 0.5" further out
ptA.x = ptB.x - 8 * pica
ptA.y = ptOrigin.y - 0.5 * wBridge
ptB.x = ptA.x
ptB.y = ptOrigin.y + 0.5 * wBridge
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Arrowheads
ActiveModelReference.AddElement oLine
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(90))
Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(-90))
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
'Dimension and GUTTER TO GUTTER
ptA.x = ptA.x - pica
ptA.y = ptOrigin.y
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
strText = strDim(wBridge)
SetText ("Normal")
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(90))
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText
strText = "GUTTER TO GUTTER"
ptA.x = ptA.x + 2 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterTop
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText

'Place half-width bridge dimensions 0.5in right of overall width
ptB.x = ptB.x + 8 * pica
ptA.x = ptB.x
'Store value of end of centerline as global to use in LastSpan
staX = ptB.x - 2 * pica
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(90))
Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(-90))
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
ptB.y = ptB.y - wBridge
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(90))
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
ptA.x = ptA.x - pica
ptA.y = ptA.y + 0.25 * wBridge
strText = strDim(0.5 * wBridge)
SetText ("Normal")
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(90))
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText
ptA.y = ptA.y - 0.5 * wBridge
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText
```

```
'Put ptB.x back to end of short extension lines and C at long ones
ptB.x = ptB.x - 2 * pica
ptC.x = ptB.x - 8 * pica

If cboBarrierSidewalk.Text = "Sidewalk" Then
    'Sidewalk needs short line at face of parapet and long line at gutter
    'Extension line at face of left parapet
    ptA.x = xLeftPost - pica
    ptA.y = ptOrigin.y + 0.5 * wBridge + wSidewalk
    ptB.y = ptA.y
    SetLine ("Dimension")
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    'Extension at left gutter, needs to be longer
    offset = 0.5 * wBridge
    ptA.x = ptOrigin.x + offset / Tan(skew) - (wPaveRest + pica) / Sin(skew)
    ptA.y = ptOrigin.y + offset
    ptC.y = ptA.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    'Extension at right gutter, also longer
    ptA.x = ptOrigin.x - offset / Tan(skew) - (wPaveRest + pica) / Sin(skew)
    ptA.y = ptOrigin.y - offset
    ptC.y = ptA.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    'Extension at face of right parapet, needs to be shorter
    ptA.x = xRightPost - pica
    ptA.y = ptOrigin.y - 0.5 * wBridge - wSidewalk
    ptB.y = ptA.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    'Sidewalk dimension line on left side
    ptB.x = ptB.x + 2 * pica
    ptB.y = ptOrigin.y + 0.5 * wBridge + wSidewalk
    ptC.x = ptB.x
    ptC.y = ptB.y - wSidewalk
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(90))
    Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, oMatrix)
    ActiveModelReference.AddElement oCell
    oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(-90))
    Set oCell = CreateCellElement2(strCell, ptC, ptScale, False, oMatrix)
    ActiveModelReference.AddElement oCell
    'Now a callout for the sidewalk
    ptB.y = ptB.y - 0.5 * wSidewalk
    ptA.x = ptB.x - 4 * pica
    ptA.y = ptB.y
    Set oLine = CreateLineElement2(Nothing, ptB, ptA)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x - pica
    strText = strDim(wSidewalk) & " SIDEWALK"
    SetText ("Normal")
    ActiveSettings.TextStyle.Justification = msdTextJustificationRightCenter
    Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
    ActiveModelReference.AddElement oText
    'Now do right side
    ptB.y = ptOrigin.y - 0.5 * wBridge
    ptC.y = ptB.y - wSidewalk
    SetLine ("Dimension")
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(90))
```

```
    Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, oMatrix)
    ActiveModelReference.AddElement oCell
    oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(-90))
    Set oCell = CreateCellElement2(strCell, ptC, ptScale, False, oMatrix)
    ActiveModelReference.AddElement oCell
    'Now a callout for the sidewalk
    ptB.y = ptB.y - 0.5 * wSidewalk
    ptA.x = ptB.x - 4 * pica
    ptA.y = ptB.y
    Set oLine = CreateLineElement2(Nothing, ptB, ptA)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x - pica
    strText = strDim(wSidewalk) & " SIDEWALK"
    SetText ("Normal")
    ActiveSettings.TextStyle.Justification = msdTextJustificationRightCenter
    Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
    ActiveModelReference.AddElement oText

Else                    'Barrier only

    'Long extension line at left gutter
    ptA.x = xLeftPost - pica
    ptA.y = ptOrigin.y + 0.5 * wBridge
    ptC.y = ptA.y
    SetLine ("Dimension")
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    'Long extension line at right gutter
    ptA.x = xRightPost - pica
    ptA.y = ptOrigin.y - 0.5 * wBridge
    ptC.y = ptA.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
End If

'Begin Bridge callout placed 0.375 inch above centerline
dAngle = -135
If skew < Pi / 2 Then dAngle = 135
ptA.x = ptOrigin.x + 6 * pica
'For forward skew the callout is below the centerline pointing up.
ptA.y = ptOrigin.y + 6 * pica
'Otherwise the callout is above the centerline pointing down.
If skew < Pi / 2 Then ptA.y = ptOrigin.y - 6 * pica
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptOrigin)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x + 24 * pica
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(dAngle))
Set oCell = CreateCellElement2(strCell, ptOrigin, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
ptA.x = ptA.x + 2 * pica
ptA.y = ptA.y + pica
strText = "BEGIN BRIDGE"
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftBottom
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.y = ptA.y - 2 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftTop
strText = "STA. " & Format(ptOrigin.x, "##+#0.00")
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
```

```
ActiveModelReference.AddElement oText

'BFPR label placed at quarter point of gutter-to-gutter spacing
offset = wBridge / 4
'For forward skew, place this label above centerline (avoid Begin Bridge conflict)
ptA.x = ptOrigin.x + offset / Abs(Tan(skew))
ptA.y = ptOrigin.y - offset
If skew < 0.5 * Pi Then ptA.y = ptOrigin.y + offset
'Place arrowhead
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(180))
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
'Place callout line
ptB.x = ptA.x + 8 * pica
ptB.y = ptA.y
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Place text
strText = "B.F.P.R. BENT 1"
SetText ("Normal")
ptB.x = ptB.x + pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText

'Draw span dimension 0.5 inch above outside of wing
'left extension line only; span 2 will get the right extension line
ptA.x = ptOrigin.x
'Beginning of extension line varies for forward, backward, or normal cases
Select Case Round(Degrees(skew), 0)
    Case Is < 85    'for forward skew, extension line is 1/16th from paving rest
        ptA.y = ptOrigin.y + (wPaveRest + pica) / Cos(skew)
    Case Is > 95    'for backward skew, extension line is 1/16th from BFPR
        ptA.y = ptOrigin.y - pica / Cos(skew)
    Case Else       'for normal bridge, start extension line at outside of wing
        ptA.y = ptOrigin.y + 0.5 * wBridge + wSidewalk + _
            wBarrier + wSlot + wWing + pica
End Select
ptB.x = ptA.x
ptB.y = ptOrigin.y + 0.5 * wBridge + wSidewalk + wBarrier + wSlot + wWing + 19 *
pica
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Place span dimension line 0.5 + 0.125 inch below end of extension line
'bridge length will be done later
ptA.y = ptB.y - 10 * pica
ptB.x = ptB.x + lSpan
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(180))
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, Matrix3dIdentity)
ActiveModelReference.AddElement oCell
'Place dimension text
ptB.x = ptA.x + 0.5 * lSpan
ptB.y = ptA.y + pica
strText = strDim(lSpan)
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
```

```
ActiveModelReference.AddElement oText

'Place destination ahead arrow at quarter point of span
offset = wBridge / 4
ptA.x = staX - 14 * pica
ptA.y = ptOrigin.y + offset
ptScale.x = Val(txtScale.Value) / 120
ptScale.y = ptScale.x
strCell = "DESTLT"
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, Matrix3dIdentity)
ActiveModelReference.AddElement oCell

End Sub

Private Sub DrawPlanSpanInt(SpanNo As Long, staBack As Double, staAhead As Double)

Dim skew As Double          'Skew angle (in radians; if backwards, supplement of
skew)
Dim ptA As Point3d          'Working points
Dim ptB As Point3d
Dim ptC As Point3d
Dim ptOrigin As Point3d     'Origin point
Dim ptScale As Point3d
Dim pica As Double          'Scaled spacing constant (1/16th inch on sheet)
Dim dScale As Double        'Scale factor for placing arrowheads
Dim dAngle As Double        'Angle for callouts (45 degrees)
Dim offset As Double        'offset from centerline

Dim wBridge As Double       'width of Bridge, gutter to gutter
Dim wSidewalk As Double     'width of sidewalk
Dim wLip As Double          'width of lip behind barrier (1.5")
Dim wBarrier As Double      'width of barrier (1'-6")
Dim wBarrTop As Double      'width of top of barrier (9")
Dim wWing As Double         'width of wingwall (for placement of dimension line)
Dim wSlot As Double         'back of barrier to face of wing (for placement of dim
line)
Dim lSpan As Double         'length of Span 1k

Dim oLine As LineElement    'line holder
Dim oCell As CellElement    'cell holder
Dim oText As TextElement    'text holder
Dim oArc As ArcElement      'arc for skew angle
Dim strText As String       'working string for dimensions and notes
Dim strCell As String       'working string for cell names
Dim oMatrix As Matrix3d     'rotation matrix holder for cells and text

'Assume values
wBarrier = 1.5
wBarrTop = 0.75
wLip = 1.5 / 12
wWing = 1
wSlot = 4 / 12
wSidewalk = 0

'Get values from form
wBridge = Val(txtWidth.Value)
skew = Pi * Val(txtSkew.Value) / 180
If Left(cboSkewDir.Value, 1) = "B" Then skew = Pi - skew
pica = Val(txtScale.Value) / 192
If cboBarrierSidewalk.Text = "Sidewalk" Then
    wSidewalk = Val(txtSidewalkWidth.Value)
    wBarrier = 13 / 12
    wBarrTop = 13 / 12
End If
```

```
'Place origin at 5.5" plus half of bridge width above begin of bridge elev
ptOrigin.x = staBack
ptOrigin.y = elev(Val(txtBeginBr.Value)) + 88 * pica + _
    0.5 * wBridge + wSidewalk + wBarrier
ptOrigin.Z = 0
ptA = ptOrigin
ptB = ptOrigin
ptC = ptOrigin

'Define scaling point for placement of cells
ptScale.x = Val(txtScale.Value)
ptScale.y = Val(txtScale.Value)
ptScale.Z = 1
strCell = "AR8"
CellExists (strCell)

'Calculate length of this span
lSpan = staAhead - staBack

'Draw back bent line (ahead bent line will be in next span)
offset = 0.5 * wBridge + wSidewalk
ptA.x = ptOrigin.x - offset / Tan(skew)
ptA.y = ptOrigin.y - offset
ptB.x = ptOrigin.x + offset / Tan(skew)
ptB.y = ptOrigin.y + offset
SetLine ("Concrete")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw left barrier joint
ptC.x = ptB.x
ptC.y = ptB.y + wBarrier + wLip
Set oLine = CreateLineElement2(Nothing, ptB, ptC)
ActiveModelReference.AddElement oLine
'Draw right barrier joint
ptC.x = ptA.x
ptC.y = ptA.y - wBarrier - wLip
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine

'Draw dimension lines
'Draw back extension line
ptA.x = ptOrigin.x
'For skewed bridge, extension line goes to centerline bridge (1/16th off bent)
ptA.y = ptOrigin.y + pica / Abs(Cos(skew))
'For normal bridge extension line starts off of edge of deck
If Degrees(skew) > 85 And Degrees(skew) < 95 Then
       ptA.y = ptOrigin.y + 0.5 * wBridge + wSidewalk + wBarrier + wLip + pica
End If
ptB.x = ptOrigin.x
ptB.y = ptOrigin.y + 0.5 * wBridge + wSidewalk + wBarrier + wSlot + wWing + 11 *
pica
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Place span dimension line 0.5 + 0.125 inch below end of extension line
'bridge length will be done later
ptA.y = ptB.y - 2 * pica
ptB.x = ptB.x + lSpan
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(180))
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
```

```
ActiveModelReference.AddElement oCell
Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, Matrix3dIdentity)
ActiveModelReference.AddElement oCell
'Place dimension text
ptB.x = ptA.x + 0.5 * lSpan
ptB.y = ptA.y + pica
strText = strDim(lSpan)
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText

'CL BENT label placed at quarter point of gutter-to-gutter spacing below CL
offset = wBridge / 4
ptA.x = ptOrigin.x - offset / Tan(skew)
ptA.y = ptOrigin.y - offset
'Place arrowhead
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(180))
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
'Place callout line
ptB.x = ptA.x + 8 * pica
ptB.y = ptA.y
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Place text
strText = "c BENT " & Format(SpanNo, "####")
ptB.x = ptB.x + pica
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText

'Show skew angle in span 2
If SpanNo = 2 Then
    'For forward skew, angle is shown at back bent above centerline
    If skew <= 0.5 * Pi Then
        'Draw a 0.5in radius arc from centerline to bent
        ptA.x = ptOrigin.x + 8 * pica
        ptA.y = ptOrigin.y
        ptB.x = ptOrigin.x + 8 * pica * Cos(skew)
        ptB.y = ptOrigin.y + 8 * pica * Sin(skew)
        SetLine ("Dimension")
        Set oArc = CreateArcElement1(Nothing, ptA, ptOrigin, ptB)
        ActiveModelReference.AddElement oArc
        'Figure out angle correction for arrowheads
        'arrowhead is 0.8pica long, radius is 8 pica
        'delta angle = atn(0.8*pica/(8*pica)) => 5.7106 degrees
        'Arrowhead at bent
        dAngle = 0.5 * Pi + skew - Radians(5.7106)
        oMatrix = Matrix3dFromAxisAndRotationAngle(2, dAngle)
        Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, oMatrix)
        ActiveModelReference.AddElement oCell
        'Arrowhead at centerline bridge
        dAngle = Radians(5.7106) - 0.5 * Pi
        oMatrix = Matrix3dFromAxisAndRotationAngle(2, dAngle)
        Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
        ActiveModelReference.AddElement oCell
        'Draw callout at halfway up skew angle, sticking out 0.25 inch
        ptA.x = ptOrigin.x + 8 * pica * Cos(skew / 2)
        ptA.y = ptOrigin.y + 8 * pica * Sin(skew / 2)
        ptB.x = ptOrigin.x + 12 * pica * Cos(skew / 2)
        ptB.y = ptOrigin.y + 12 * pica * Sin(skew / 2)
```

```
        Set oLine = CreateLineElement2(Nothing, ptA, ptB)
        ActiveModelReference.AddElement oLine
        ptA.x = ptB.x + 4 * pica
        ptA.y = ptB.y
        Set oLine = CreateLineElement2(Nothing, ptA, ptB)
        ActiveModelReference.AddElement oLine
        ptA.x = ptA.x + pica
        strText = strAngle(skew) & ", TYP."
        SetText ("Normal")
        ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
        Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
        ActiveModelReference.AddElement oText

    Else
        'For backward skew, angle is shown below centerline
        ptC.x = ptOrigin.x + lSpan
        ptC.y = ptOrigin.y
        ptA.x = ptC.x - 8 * pica
        ptA.y = ptC.y
        ptB.x = ptC.x + 8 * pica * Cos(skew)
        ptB.y = ptC.y + 8 * pica * Sin(skew)
        SetLine ("Dimension")
        Set oArc = CreateArcElement1(Nothing, ptB, ptC, ptA)
        ActiveModelReference.AddElement oArc
        'Figure out angle correction for arrowheads
        'arrowhead is 0.8pica long, radius is 8 pica
        'delta angle = atn(0.8*pica/(8*pica)) => 5.7106 degrees
        'Arrowhead at bent
        dAngle = skew - 0.5 * Pi + Radians(5.7106)
        oMatrix = Matrix3dFromAxisAndRotationAngle(2, dAngle)
        Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, oMatrix)
        ActiveModelReference.AddElement oCell
        'Arrowhead at centerline bridge
        dAngle = Radians(-5.7106) - 0.5 * Pi
        oMatrix = Matrix3dFromAxisAndRotationAngle(2, dAngle)
        Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
        ActiveModelReference.AddElement oCell
        'Draw callout at halfway up skew angle, sticking out 0.25 inch
        ptA.x = ptC.x - 8 * pica * Cos(0.5 * (Pi - skew))
        ptA.y = ptC.y + 8 * pica * Sin(0.5 * (Pi - skew))
        ptB.x = ptC.x - 12 * pica * Cos(0.5 * (Pi - skew))
        ptB.y = ptC.y + 12 * pica * Sin(0.5 * (Pi - skew))
        Set oLine = CreateLineElement2(Nothing, ptA, ptB)
        ActiveModelReference.AddElement oLine
        ptA.x = ptB.x - 4 * pica
        ptA.y = ptB.y
        Set oLine = CreateLineElement2(Nothing, ptA, ptB)
        ActiveModelReference.AddElement oLine
        ptA.x = ptA.x - pica
        strText = strAngle(Pi - skew) & ", TYP."
        SetText ("Normal")
        ActiveSettings.TextStyle.Justification = msdTextJustificationRightCenter
        Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
        ActiveModelReference.AddElement oText
    End If
End If

'Draw left barrier
offset = 0.5 * wBridge
ptA.x = ptOrigin.x + offset / Tan(skew)
'Draw gutterline
ptA.y = ptOrigin.y + offset
ptB.x = ptA.x + lSpan
ptB.y = ptA.y
```

```
SetLine ("Concrete")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
If cboBarrierSidewalk.Value = "Sidewalk" Then
    ptA.x = ptA.x + wSidewalk / Tan(skew)
    ptA.y = ptA.y + wSidewalk
    ptB.x = ptA.x + lSpan
    ptB.y = ptA.y
End If
'Draw outside edge of bridge
ptA.y = ptA.y + wBarrier + wLip
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw back of barrier
ptA.y = ptA.y - wLip
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw top inside face of rail
ptA.y = ptA.y - wBarrTop
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine


'Draw right barrier
offset = 0.5 * wBridge
ptA.x = ptOrigin.x - offset / Tan(skew)
'Draw gutterline
ptA.y = ptOrigin.y - offset
ptB.x = ptA.x + lSpan
ptB.y = ptA.y
SetLine ("Concrete")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
If cboBarrierSidewalk.Value = "Sidewalk" Then
    ptA.x = ptA.x - wSidewalk / Tan(skew)
    ptA.y = ptA.y - wSidewalk
    ptB.x = ptA.x + lSpan
    ptB.y = ptA.y
End If
'Draw outside edge of bridge
ptA.y = ptA.y - wBarrier - wLip
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw back of barrier
ptA.y = ptA.y + wLip
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw top inside face of rail
ptA.y = ptA.y + wBarrTop
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

End Sub

Private Sub DrawPlanSpanLast(SpanNo As Long, staBack As Double, staAhead As Double)

Dim skew As Double          'Skew angle (in radians; if backwards, supplement of
skew)
```

```
Dim ptA As Point3d          'Working points
Dim ptB As Point3d
Dim ptC As Point3d
Dim ptOrigin As Point3d     'Origin point
Dim ptScale As Point3d
Dim pica As Double          'Scaled spacing constant (1/16th inch on sheet)
Dim dScale As Double        'Scale factor for placing arrowheads
Dim dAngle As Double        'Angle for callouts (45 degrees)
Dim offset As Double        'offset from centerline
Dim longpost As Double      'length of longer end post
Dim shortpost As Double     'length of shorter end post

Dim wBridge As Double       'width of Bridge, gutter to gutter
Dim wSidewalk As Double     'width of sidewalk
Dim hSlab As Double         'height of slab (for wingwall length calc)
Dim hBeam As Double         'height of beam (or wingwall length calc)
Dim wSlot As Double         'width between back of end post and face of wing (4")
Dim wLip As Double          'width of lip behind barrier (1.5")
Dim wEndBent As Double      'width of end bent cap
Dim lEndPost As Double      'length of end post (could be calculated?)
Dim wEndPost As Double      'width of End Post (1'-1")
Dim wPaveRest As Double     'width of paving rest (8")
Dim wBarrier As Double      'width of barrier (1'-6")
Dim wBarrTop As Double      'width of top of barrier (9")
Dim lTrans As Double        'length of barrier transition (10')
Dim wPileBox As Double      'outside of pile box to face of wing (9")
Dim lPileBox As Double      'length of pile box (2'-6")
Dim wWing As Double         'width of wingwall (1')
Dim lWing As Double         'length of wingwall (calculated)
Dim lSpan As Double         'length of span

Dim oLine As LineElement    'line holder
Dim oCell As CellElement    'cell holder
Dim oText As TextElement    'text holder
Dim strText As String       'working string for dimensions and notes
Dim strCell As String       'working string for cell names
Dim oMatrix As Matrix3d     'rotation matrix holder for cells and text
Dim oCircle As EllipseElement   'ellipse holder
Dim oArc As ArcElement      'arc holder

'Assume some values
wBarrier = 1.5
wBarrTop = 0.75
wLip = 1.5 / 12
wSlot = 4 / 12
wEndPost = 13 / 12
lEndPost = 4
wPaveRest = 8 / 12
wWing = 1
wPileBox = 0.75
lPileBox = 2.5
lTrans = 10
hSlab = 8 / 12

'Get values from form
wBridge = Val(txtWidth.Value)
wEndBent = Val(txtEndBentWidth.Value)
hBeam = Val(txtBeamDepth.Value)
skew = Pi * Val(txtSkew.Value) / 180
If Left(cboSkewDir.Value, 1) = "B" Then skew = Pi - skew
pica = Val(txtScale.Value) / 192
lSpan = staAhead - staBack
wSidewalk = 0
If cboBarrierSidewalk.Text = "Sidewalk" Then
```

```
    wSidewalk = Val(txtSidewalkWidth.Value)
    wBarrier = 13 / 12
    wBarrTop = 13 / 12
End If

'Place origin at end bridge at 5.5" plus half of bridge width above plan
ptOrigin.x = staAhead
ptOrigin.y = elev(Val(txtBeginBr.Value)) + 88 * pica + _
    0.5 * wBridge + wSidewalk + wBarrier
ptOrigin.Z = 0
ptA = ptOrigin
ptB = ptOrigin
ptC = ptOrigin

'Define scaling point for placement of cells
ptScale.x = Val(txtScale.Value)
ptScale.y = Val(txtScale.Value)
ptScale.Z = 1
strCell = "AR8"
CellExists (strCell)

'Calculate wingwall length based on beam height and skew, rounded to nearest 3"
lWing = Round(8 * (hSlab + hBeam + 2 / 12 + 1) / Sin(skew)) / 4

'Calculate length of end posts
longpost = Round(4 * (0.125 + (wPaveRest + wEndBent + 2 / 12) / Sin(skew) + _
    (wEndPost + wSlot) / Abs(Tan(skew)))) / 4
If longpost < 4 Then longpost = 4
shortpost = Round(4 * (0.125 + (wPaveRest + wEndBent + 2 / 12) / Sin(skew) - _
    (wEndPost + wSlot) / Abs(Tan(skew)))) / 4
If shortpost < 4 Then shortpost = 4


'Draw BFPR and paving rest
offset = 0.5 * wBridge + wBarrier - wBarrTop + wSidewalk
ptA.x = ptOrigin.x - offset / Tan(skew)
ptA.y = ptOrigin.y - offset
ptB.x = ptOrigin.x + offset / Tan(skew)
ptB.y = ptOrigin.y + offset
SetLine ("Concrete")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + wPaveRest / Sin(skew)
ptB.x = ptB.x + wPaveRest / Sin(skew)
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw right wing wall
offset = offset + wEndPost + wSlot
ptA.x = ptOrigin.x - offset / Tan(skew) - wEndBent / Sin(skew)
ptA.y = ptOrigin.y - offset
ptB.x = ptA.x
ptB.y = ptA.y - wWing
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + lWing
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = ptB.y + wWing
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
```

```
'If wing is longer than 11', draw pile box
If lWing > 11 Then
    'Draw outside portion of box
    ptC.x = ptA.x
    ptC.y = ptA.y - wPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x - lPileBox
    ptA.y = ptC.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptC.x = ptA.x
    ptC.y = ptC.y + wPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    'Draw inner portion of box
    ptC.x = ptB.x
    ptC.y = ptB.y + wPileBox
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    ptA.y = ptC.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptC.x = ptC.x - lPileBox
    ptC.y = ptB.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptA.x = ptB.x
    ptA.y = ptB.y - 2 * wPileBox + wWing
End If
'Draw 20' limits of Rip Rap off of ptA (don't change ptB)
'ptA will be on corner of wing if no box, otherwise corner of pile box
SetLine ("Dimension")
ptA.y = ptA.y - pica
ptC.x = ptA.x
ptC.y = ptA.y - 12 * pica
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
'It takes a line of at least 30 pica to fit the text. Therefore
'make the length of the line ("offset") be 20' or at least 30 pica
offset = 30 * pica
If 20 > 30 * pica Then offset = 20
ptA.x = ptA.x + offset
ptA.y = ptC.y + 2 * pica
ptC.y = ptC.y + 2 * pica
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(180))
Set oCell = CreateCellElement2(strCell, ptC, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, Matrix3dIdentity)
ActiveModelReference.AddElement oCell
ptC.x = ptC.x + 0.5 * offset
ptC.y = ptC.y + pica
strText = "20'-0" & """
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
Set oText = CreateTextElement1(Nothing, strText, ptC, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptC.y = ptC.y - 4 * pica
strText = "LIMITS OF"
Set oText = CreateTextElement1(Nothing, strText, ptC, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptC.y = ptC.y - 3 * pica
```

```
strText = "RIP RAP, TYP."
Set oText = CreateTextElement1(Nothing, strText, ptC, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.y = ptA.y - 2 * pica
ptC.x = ptA.x
ptC.y = ptA.y + 12 * pica
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
SetLine ("Concrete")

'Draw last piece of wing
ptA.x = ptB.x - lWing
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptB, ptA)
ActiveModelReference.AddElement oLine


'Draw right End Post
ptB.x = ptA.x
ptB.y = ptA.y + wSlot + wEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'If forward skew, use long endpost, otherwise short endpost
lEndPost = longpost
If skew > 0.5 * Pi Then lEndPost = shortpost
ptA.x = ptA.x + lEndPost
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = ptA.y - wEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x - lEndPost
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw right barrier
offset = 0.5 * wBridge          'face of sidewalk or barrier
ptA.y = ptA.y + wEndPost        'ptA is at end of inside end of end post
ptC.x = ptOrigin.x - lSpan - offset / Tan(skew)
ptC.y = ptOrigin.y - offset
If wSidewalk = 0 Then
    'Draw transition
    ptB.x = ptA.x - 10
    ptB.y = ptA.y + wBarrier - wBarrTop
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    'Put ptC at end of span at top inside face of barrier
    ptC.y = ptC.y - wBarrier + wBarrTop
Else
    'Draw gutter at face of sidewalk
    ptB.x = ptOrigin.x - offset / Tan(skew)
    ptB.y = ptOrigin.y - offset
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    'Put ptC at end of span at inside face of parapet
    ptC.x = ptC.x - wSidewalk / Tan(skew)
    ptC.y = ptC.y - wSidewalk
End If
```

```
'Draw inside top of rail
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
'Draw back of barrier
ptA.y = ptA.y - wBarrTop
ptC.y = ptC.y - wBarrTop
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
'Draw edge of deck
ptA.y = ptA.y - wLip
ptC.y = ptC.y - wLip
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine

'Draw left wing (change signs on delta y's)
offset = 0.5 * wBridge + wBarrier - wBarrTop + wEndPost + wSlot
If cboBarrierSidewalk.Value = "Sidewalk" Then offset = offset + wSidewalk
ptA.x = ptOrigin.x + offset / Tan(skew) - wEndBent / Sin(skew)
ptA.y = ptOrigin.y + offset
ptB.x = ptA.x
ptB.y = ptA.y + wWing
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + lWing
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = ptB.y - wWing
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'If wing is longer than 11', draw pile box (don't change ptB)
If lWing > 11 Then
    'Draw outside portion of box
    ptC.x = ptB.x
    ptC.y = ptA.y + wPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x - lPileBox
    ptA.y = ptC.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptC.x = ptA.x
    ptC.y = ptC.y - wPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    'Draw inner portion of box
    ptC.x = ptB.x
    ptC.y = ptB.y - wPileBox
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    ptA.y = ptC.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptC.x = ptC.x - lPileBox
    ptC.y = ptB.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptC)
    ActiveModelReference.AddElement oLine
    ptA.x = ptB.x
    ptA.y = ptB.y + 2 * wPileBox + wWing
End If
'Draw last piece of wing
ptA.x = ptB.x - lWing
ptA.y = ptB.y
```

```
Set oLine = CreateLineElement2(Nothing, ptB, ptA)
ActiveModelReference.AddElement oLine
'Draw left End Post
ptB.x = ptA.x
ptB.y = ptA.y - wSlot - wEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'If backward skew, use long endpost, otherwise short endpost
lEndPost = longpost
If skew < 0.5 * Pi Then lEndPost = shortpost
ptA.x = ptA.x + lEndPost
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = ptA.y + wEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x - lEndPost
ptA.y = ptB.y

Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw left barrier (change sign of y offsets)
offset = 0.5 * wBridge          'face of sidewalk or barrier
ptA.y = ptA.y - wEndPost        'ptA is at end of inside end of end post
ptC.x = ptOrigin.x - lSpan + offset / Tan(skew)
ptC.y = ptOrigin.y + offset
If wSidewalk = 0 Then
    'Draw transition
    ptB.x = ptA.x - 10
    ptB.y = ptA.y - wBarrier + wBarrTop
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    'Put ptC at end of span at top inside face of barrier
    ptC.y = ptC.y + wBarrier - wBarrTop
Else
    'Draw gutter at face of sidewalk
    ptB.x = ptOrigin.x + offset / Tan(skew)
    ptB.y = ptOrigin.y + offset
    Set oLine = CreateLineElement2(Nothing, ptB, ptC)
    ActiveModelReference.AddElement oLine
    'Put ptC at end of span at inside face of parapet
    ptC.x = ptC.x + wSidewalk / Tan(skew)
    ptC.y = ptC.y + wSidewalk
End If
'Draw inside top of rail
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
'Draw back of barrier
ptA.y = ptA.y + wBarrTop
ptC.y = ptC.y + wBarrTop
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
'Draw edge of deck
ptA.y = ptA.y + wLip
ptC.y = ptC.y + wLip
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine
```

```
'Draw back bent line (ahead bent line will be in next span)
offset = 0.5 * wBridge + wSidewalk
ptA.x = ptOrigin.x - offset / Tan(skew) - lSpan
ptA.y = ptOrigin.y - offset
ptB.x = ptOrigin.x + offset / Tan(skew) - lSpan
ptB.y = ptOrigin.y + offset
SetLine ("Concrete")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw left barrier joint
ptC.x = ptB.x
ptC.y = ptB.y + wBarrier + wLip
Set oLine = CreateLineElement2(Nothing, ptB, ptC)
ActiveModelReference.AddElement oLine
'Draw right barrier joint
ptC.x = ptA.x
ptC.y = ptA.y - wBarrier - wLip
Set oLine = CreateLineElement2(Nothing, ptA, ptC)
ActiveModelReference.AddElement oLine


'CL Bent label placed at quarter point of gutter-to-gutter spacing below CL
SetLine ("Dimension")
offset = wBridge / 4
'For forward skew, place this label above centerline (avoid Begin Bridge conflict)
ptA.x = ptOrigin.x - lSpan - offset / Tan(skew)
ptA.y = ptOrigin.y - offset
'Place arrowhead
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(180))
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
'Place callout line
ptB.x = ptA.x + 8 * pica
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Place text
strText = "c BENT " & Format(SpanNo, "####")
SetText ("Normal")
ptB.x = ptB.x + pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText

'End Bridge callout placed 0.375 inch above centerline
dAngle = 45
If skew < Pi / 2 Then dAngle = -45
ptA.x = ptOrigin.x - 6 * pica
'For forward skew the callout is above the centerline pointing down.
ptA.y = ptOrigin.y - 6 * pica
'Otherwise the callout is below the centerline pointing up.
If skew < Pi / 2 Then ptA.y = ptOrigin.y + 6 * pica
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptOrigin)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x - 24 * pica
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(dAngle))
Set oCell = CreateCellElement2(strCell, ptOrigin, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
ptA.x = ptA.x - 2 * pica
ptA.y = ptA.y + pica
```

```
strText = "END BRIDGE"
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationRightBottom
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.y = ptA.y - 2 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationRightTop
strText = "STA. " & Format(ptOrigin.x, "##+#0.00")
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText

'BFPR label placed at quarter point of gutter-to-gutter spacing
offset = wBridge / 4
'For forward skew, place this label below centerline (avoid Begin Bridge conflict)
ptA.x = ptOrigin.x - offset / Abs(Tan(skew))
ptA.y = ptOrigin.y + offset
If skew < 0.5 * Pi Then ptA.y = ptOrigin.y - offset
'Place arrowhead
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, Matrix3dIdentity)
ActiveModelReference.AddElement oCell
'Place callout line
ptB.x = ptA.x - 8 * pica
ptB.y = ptA.y
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Place text
strText = "B.F.P.R. BENT " & Format(SpanNo + 1)
SetText ("Normal")
ptB.x = ptB.x - pica
ActiveSettings.TextStyle.Justification = msdTextJustificationRightCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText

'Draw CL Bridge callout with a 75-degree line
dAngle = Radians(75)
ptA.x = ptOrigin.x + 16 * pica
ptA.y = ptOrigin.y
oMatrix = Matrix3dFromAxisAndRotationAngle(2, -1 * dAngle)
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
ptB.x = ptA.x - 10 * pica / Tan(dAngle)
ptB.y = ptA.y + 10 * pica
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptB.x + 4 * pica
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + 2 * pica
strText = "c BRIDGE"
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.y = ptA.y - 3 * pica
strText = "= c CONSTRUCTION"
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.y = ptA.y - 3 * pica
strText = "= PROFILE GRADE LINE"
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
```

```
'Show skew angle in span 2
If SpanNo = 2 Then
    'For forward skew, angle is shown at back bent above centerline
    If skew <= 0.5 * Pi Then
        'Draw a 0.5in radius arc from centerline to bent
        ptA.x = ptOrigin.x + 8 * pica - lSpan
        ptA.y = ptOrigin.y
        ptB.x = ptOrigin.x + 8 * pica * Cos(skew) - lSpan
        ptB.y = ptOrigin.y + 8 * pica * Sin(skew)
        ptC.x = ptOrigin.x - lSpan
        ptC.y = ptOrigin.y
        SetLine ("Dimension")
        Set oArc = CreateArcElement1(Nothing, ptA, ptC, ptB)
        ActiveModelReference.AddElement oArc
        'Figure out angle correction for arrowheads
        'arrowhead is 0.8pica long, radius is 8 pica
        'delta angle = atn(0.8*pica/(8*pica)) => 5.7106 degrees
        'Arrowhead at bent
        dAngle = 0.5 * Pi + skew - Radians(5.7106)
        oMatrix = Matrix3dFromAxisAndRotationAngle(2, dAngle)
        Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, oMatrix)
        ActiveModelReference.AddElement oCell
        'Arrowhead at centerline bridge
        dAngle = Radians(5.7106) - 0.5 * Pi
        oMatrix = Matrix3dFromAxisAndRotationAngle(2, dAngle)
        Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
        ActiveModelReference.AddElement oCell
        'Draw callout at halfway up skew angle, sticking out 0.25 inch
        ptA.x = ptC.x + 8 * pica * Cos(skew / 2)
        ptA.y = ptC.y + 8 * pica * Sin(skew / 2)
        ptB.x = ptC.x + 12 * pica * Cos(skew / 2)
        ptB.y = ptC.y + 12 * pica * Sin(skew / 2)
        Set oLine = CreateLineElement2(Nothing, ptA, ptB)
        ActiveModelReference.AddElement oLine
        ptA.x = ptB.x + 4 * pica
        ptA.y = ptB.y
        Set oLine = CreateLineElement2(Nothing, ptA, ptB)
        ActiveModelReference.AddElement oLine
        ptA.x = ptA.x + pica
        strText = strAngle(skew) & ", TYP."
        SetText ("Normal")
        ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
        Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
        ActiveModelReference.AddElement oText

    Else
        'For backward skew, angle is shown at ahead bent above centerline
        'This keeps it from conflicting with the back CL bent callout
        'Draw a 0.5in radius arc from centerline to bent
        ptC.x = ptOrigin.x
        ptC.y = ptOrigin.y
        ptA.x = ptC.x - 8 * pica
        ptA.y = ptC.y
        ptB.x = ptC.x + 8 * pica * Cos(skew)
        ptB.y = ptC.y + 8 * pica * Sin(skew)
        SetLine ("Dimension")
        Set oArc = CreateArcElement1(Nothing, ptB, ptC, ptA)
        ActiveModelReference.AddElement oArc
        'Figure out angle correction for arrowheads
        'arrowhead is 0.8pica long, radius is 8 pica
        'delta angle = atn(0.8*pica/(8*pica)) => 5.7106 degrees
        'Arrowhead at bent
        dAngle = skew - 0.5 * Pi + Radians(5.7106)
```

```
        oMatrix = Matrix3dFromAxisAndRotationAngle(2, dAngle)
        Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, oMatrix)
        ActiveModelReference.AddElement oCell
        'Arrowhead at centerline bridge
        dAngle = Radians(-5.7106) - 0.5 * Pi
        oMatrix = Matrix3dFromAxisAndRotationAngle(2, dAngle)
        Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
        ActiveModelReference.AddElement oCell
        'Draw callout at halfway up skew angle, sticking out 0.25 inch
        ptA.x = ptC.x - 8 * pica * Cos(0.5 * (Pi - skew))
        ptA.y = ptC.y + 8 * pica * Sin(0.5 * (Pi - skew))
        ptB.x = ptC.x - 12 * pica * Cos(0.5 * (Pi - skew))
        ptB.y = ptC.y + 12 * pica * Sin(0.5 * (Pi - skew))
        Set oLine = CreateLineElement2(Nothing, ptA, ptB)
        ActiveModelReference.AddElement oLine
        ptA.x = ptB.x - 4 * pica
        ptA.y = ptB.y
        Set oLine = CreateLineElement2(Nothing, ptA, ptB)
        ActiveModelReference.AddElement oLine
        ptA.x = ptA.x - pica
        strText = strAngle(Pi - skew) & ", TYP."
        SetText ("Normal")
        ActiveSettings.TextStyle.Justification = msdTextJustificationRightCenter
        Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
        ActiveModelReference.AddElement oText
    End If
End If


'Draw span dimension 0.5 inch above outside of wing
'Right extension line
ptA.x = ptOrigin.x
'Beginning of extension line varies for forward, backward, or normal cases
Select Case Round(Degrees(skew), 0)
    Case Is < 85    'for forward skew, extension line is 1/16th from paving rest
        ptA.y = ptOrigin.y + pica / Cos(skew)
    Case Is > 95    'for backward skew, extension line is 1/16th from BFPR
        ptA.y = ptOrigin.y - (wPaveRest + pica) / Cos(skew)
    Case Else       'for normal bridge, start extension line at outside of wing
        ptA.y = ptOrigin.y + 0.5 * wBridge + wSidewalk + _
            wBarrier + wSlot + wWing + pica
End Select
ptB.x = ptA.x
ptB.y = ptOrigin.y + 0.5 * wBridge + wSidewalk + wBarrier + wSlot + wWing + 19 *
pica
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Left extension line
ptA.x = ptA.x - lSpan
'For skewed bridge, extension line goes to centerline bridge (1/16th off bent)
ptA.y = ptOrigin.y + pica / Abs(Cos(skew))
'For normal bridge extension line starts off of edge of deck
If Degrees(skew) > 85 And Degrees(skew) < 95 Then
        ptA.y = ptOrigin.y + 0.5 * wBridge + wSidewalk + wBarrier + wLip + pica
End If
ptB.x = ptA.x
ptB.y = ptOrigin.y + 0.5 * wBridge + wSidewalk + wBarrier + wSlot + wWing + 11 *
pica
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Place span dimension line 0.5 + 0.125 inch below end of extension line
ptA.y = ptB.y - 2 * pica
```

```
ptB.x = ptB.x + lSpan
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(180))
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, Matrix3dIdentity)
ActiveModelReference.AddElement oCell
'Place dimension text
ptB.x = ptA.x + 0.5 * lSpan
ptB.y = ptA.y + pica
strText = strDim(lSpan)
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText

'Draw bridge length dimension
ptA.x = Val(txtBeginBr.Value)
ptA.y = ptOrigin.y + 0.5 * wBridge + wSidewalk + wBarrier + wSlot + wWing + 17 *
pica
ptB.x = ptOrigin.x
ptB.y = ptA.y
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw arrowheads
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Radians(180))
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, oMatrix)
ActiveModelReference.AddElement oCell
Set oCell = CreateCellElement2(strCell, ptB, ptScale, False, Matrix3dIdentity)
ActiveModelReference.AddElement oCell
'Place dimension text
ptB.x = ptA.x + 0.5 * (ptB.x - ptA.x)
ptB.y = ptA.y + pica
strText = "TOTAL LENGTH OF BRIDGE = " & strDim(BrLength())
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText

'Place destination ahead arrow at quarter point of span
offset = wBridge / 4
ptA.x = ptOrigin.x + 8 * pica - offset / Tan(skew)
ptA.y = ptOrigin.y - offset - 2 * pica
ptScale.x = Val(txtScale.Value) / 120
ptScale.y = ptScale.x
strCell = "DESTRT"
Set oCell = CreateCellElement2(strCell, ptA, ptScale, False, Matrix3dIdentity)
ActiveModelReference.AddElement oCell

'Draw centerline
SetLine ("Center")
ptA.x = staX        'Calculated in Span1 and stored in global variable staX
ptA.y = ptOrigin.y
ptB.x = ptOrigin.x + 32 * pica
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Put PLAN under plan view
SetLine ("Dimension")
ptA.x = Val(txtBeginBr.Value) + 0.5 * BrLength()
```

```
ptA.y = elev(ptA.x) + 60 * pica
strText = "PLAN"
SetText ("Title")
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.x = ptA.x - 6 * pica
ptA.y = ptA.y - pica
ptB.x = ptA.x + 12 * pica
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

End Sub

Private Sub DrawElevSpanInt(SpanNo As Long, staBack As Double, staAhead As Double)

Dim ptA As Point3d          'Working points
Dim ptB As Point3d
Dim ptC As Point3d

Dim ptOrigin As Point3d     'Origin point

Dim wSidewalk As Double     'width of sidewalk
Dim pica As Double          'Scaled spacing constant (1/16th inch on sheet)
Dim hBeam As Double         'Height of beam
Dim hSlab As Double         'Height of slab
Dim hBarrier As Double      'height of barrier (2'-8")
Dim hParapet As Double      'height of parapet
Dim hBearing As Double      'height of bearing pad (2")
Dim wCap As Double          'width of cap
Dim hCap As Double          'height of cap
Dim hCol As Double          'height of column
Dim hFoot As Double         'height of footing
Dim lFoot As Double         'length of footing

Dim oLine As LineElement    'line holder
Dim oCell As CellElement    'cell holder
Dim oText As TextElement    'text holder
Dim strText As String       'working string
Dim oMatrix As Matrix3d     'rotation matrix holder for cells and text
Dim oCircle As EllipseElement   'ellipse holder

'Assume some values
hSlab = 8 / 12
hBarrier = 32 / 12
hBearing = 2 / 12
hCol = 24
hFoot = 3
lFoot = 10

ptOrigin.x = staBack
ptOrigin.y = elev(staBack)
ptOrigin.Z = 0
ptA = ptOrigin
ptB = ptOrigin
ptC = ptOrigin

'Get values from form
hBeam = Val(txtBeamDepth.Value)
hCap = Val(txtCapDepth.Value)
wCap = Val(txtCapWidth.Value)
pica = Val(txtScale.Value) / 192
If cboBarrierSidewalk.Value = "Sidewalk" Then
```

```
    wSidewalk = Val(txtSidewalkWidth.Value)
    hParapet = Val(txtParapetHeight.Value)
    hBarrier = hParapet + 0.5 + 0.03 * wSidewalk
End If

'Draw Superstructure
ptB.x = staBack
ptB.y = elev(ptB.x)
ptA.x = staAhead
ptA.y = elev(ptA.x)
SetLine ("Concrete")
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y - hSlab
ptB.y = ptB.y - hSlab
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y - hBeam
ptB.y = ptB.y - hBeam
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y + hBeam + hSlab + hBarrier
ptB.y = ptB.y + hBeam + hSlab + hBarrier
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw back joints in barrier and beam
ptA.x = ptOrigin.x
ptA.y = ptOrigin.y
ptB.x = ptA.x
ptB.y = ptA.y + hBarrier
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y - hSlab
ptB.y = ptA.y - hBeam
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw substructure
'Draw cap
ptA.x = ptOrigin.x - 0.5 * wCap
ptA.y = ptOrigin.y - hSlab - hBeam - hBearing - hCap
ptB.x = ptA.x + wCap
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y + hCap
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = ptA.y - hCap
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + wCap
ptB.x = ptA.x
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw column
If Left(cboBentType.Value, 1) = "P" Then wCap = 14 / 12
ptA.x = ptOrigin.x - 0.5 * wCap
ptA.y = ptA.y - hCap
ptB.x = ptA.x
ptB.y = ptA.y - hCol
```

```
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + wCap
ptB.x = ptA.x
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw footing if concrete type only
If Left(cboBentType.Value, 1) <> "P" Then
    ptA.x = ptOrigin.x - 0.5 * lFoot
    ptA.y = ptB.y
    ptB.x = ptA.x + lFoot
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    ptA.y = ptA.y - hFoot
    ptB.y = ptA.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    ptB.x = ptA.x
    ptB.y = ptA.y + hFoot
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x + lFoot
    ptB.x = ptA.x
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
End If


'Draw Station and Elev Marker
SetLine ("Dimension")
ptA.x = ptOrigin.x
ptA.y = ptOrigin.y + hBarrier + 2 * pica
ptB.x = ptA.x
ptB.y = ptA.y + 24 * pica
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Add STA text above line and EL text below
ptA.x = ptA.x - pica
strText = "STA. " & Format(ptOrigin.x, "###+##.00")
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftBottom
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Pi / 2)
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText
ptA.x = ptA.x + 2 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftTop
strText = "EL. " & Format(elev(ptB.x), "##0.00")
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText
strText = "**"
ptB.y = ptB.y + 2 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
ActiveModelReference.AddElement oText

'Add Bent No in circle 0.5 in below wing
ptB.y = ptOrigin.y - hSlab - hBeam - hBearing - hCap - hCol - hFoot - 2 - 8 * pica
strText = Format(SpanNo)
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText
SetLine ("Dimension")
```

```
Set oCircle = CreateEllipseElement2(Nothing, ptB, 2 * pica, 2 * pica,
Matrix3dIdentity)
ActiveModelReference.AddElement oCircle

End Sub


Public Sub DrawElevSpanLast(SpanNo As Long, staBack As Double, staAhead As Double)

Dim skew As Double          'Skew angle (in radians; if backwards, supplement of
skew)
Dim ptA As Point3d          'Working points
Dim ptB As Point3d
Dim ptC As Point3d
Dim ptOrigin As Point3d     'Origin point
Dim StaAh As Double
Dim StaBk As Double
Dim pica As Double          'Scaled spacing constant (1/16th inch on sheet)
Dim hBeam As Double         'Height of beam
Dim hSlab As Double         'Height of slab
Dim hEndPost As Double      'Height of end post
Dim hParapet As Double      'Height of parapet above sidewalk
Dim wEndBent As Double      'Width of end bent cap
Dim lEndPost As Double      'Length of end post (could be calculated?)
Dim hWingCurb As Double     'height of wingwall above grade (10")
Dim hBarrier As Double      'height of barrier (2'-8")
Dim hPileBox As Double      'height of pile box (2')
Dim lPileBox As Double      'length of pile box (2'-6")
Dim lSpan As Double         'length of span
Dim lWing As Double         'length of wingwall (calculated)
Dim wSidewalk As Double     'width of sidewalk
Dim wCap As Double
Dim hCap As Double
Dim hCol As Double
Dim hBearing As Double
Dim hFoot As Double
Dim lFoot As Double

Dim oLine As LineElement    'line holder
Dim oCell As CellElement    'cell holder
Dim oText As TextElement    'text holder
Dim strText As String       'working string
Dim oMatrix As Matrix3d     'rotation matrix holder for cells and text
Dim oCircle As EllipseElement   'ellipse holder


ptOrigin.x = staAhead
ptOrigin.y = elev(ptOrigin.x)
ptOrigin.Z = 0
ptA = ptOrigin
ptB = ptOrigin
ptC = ptOrigin

'Assume some values
hSlab = 8 / 12
hPileBox = 2
lPileBox = 2.5
lEndPost = 4
hBarrier = 32 / 12
hEndPost = hBarrier
hWingCurb = 10 / 12
hFoot = 3
lFoot = 10
hCol = 24
```

```
'Get values from form
hBeam = Val(txtBeamDepth.Value)
wEndBent = Val(txtEndBentWidth.Value)
hCap = Val(txtCapDepth.Value)
wCap = Val(txtCapWidth.Value)
skew = Pi * Val(txtSkew.Value) / 180
If Left(cboSkewDir.Value, 1) = "B" Then skew = Pi - skew
pica = Val(txtScale.Value) / 192
lSpan = staAhead - staBack
If cboBarrierSidewalk.Value = "Sidewalk" Then
    wSidewalk = Val(txtSidewalkWidth.Value)
    hParapet = Val(txtParapetHeight.Value)
    If hParapet = 2.25 Then
        hEndPost = 3.5 + 0.5 + 0.03 * wSidewalk
        hBarrier = 2.25 + 0.5 + 0.03 * wSidewalk
    Else
        hBarrier = hParapet + 0.5 + 0.03 * wSidewalk
        hEndPost = hBarrier
    End If
End If

'Calculate wingwall length based on beam height and skew, rounded to nearest 3"
lWing = Round(8 * (hSlab + hBeam + 2 / 12 + 1) / Sin(skew)) / 4

'Draw wing
SetLine ("Concrete")
ptA.x = ptOrigin.x - Val(txtEndBentWidth.Value) / Sin(skew)
ptA.y = elev(ptA.x) + hWingCurb
ptB.x = ptA.x
ptB.y = ptA.y - hWingCurb - hSlab - hBeam - 2
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw groundline (20' horizontal slope; vertical adjusted for skew)
SetLine ("Ground")
ptC.x = ptB.x - 2 / Sin(skew)
ptC.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptB, ptC)
ActiveModelReference.AddElement oLine
ptA.x = ptC.x - 20
ptA.y = ptC.y - 10 * Sin(skew)
Set oLine = CreateLineElement2(Nothing, ptC, ptA)
ActiveModelReference.AddElement oLine
'Draw 2:1 slope box with 3/8" horizontal line and 3/16" vertical
ptA.x = ptA.x + 2 * pica / Sin(skew) / Sin(Radians(26.565))
ptC.x = ptA.x + 6 * pica
ptC.y = ptA.y
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptC, ptA)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + 3 * pica
ptA.y = ptA.y - pica
strText = "2"
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterTop
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.x = ptA.x + 4 * pica
ptA.y = ptA.y + pica
strText = "1"
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftBottom
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.x = ptA.x - 2 * pica
```

```
ptA.y = ptA.y + pica
strText = "*"
ActiveSettings.TextStyle.Justification = msdTextJustificationRightBottom
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.x = ptC.x
ptA.y = ptC.y + 3 * pica
SetLine ("Dimension")
Set oLine = CreateLineElement2(Nothing, ptC, ptA)
ActiveModelReference.AddElement oLine
'Back to drawing wing
SetLine ("Concrete")
ptA.x = ptB.x + lWing
ptA.y = ptB.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'If wing is longer than 11', draw pile box
If lWing > 11 Then
    ptA.y = ptA.y + hPileBox
    ptB.x = ptA.x - lPileBox
    ptB.y = ptB.y + hPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x - lPileBox
    ptA.y = ptA.y - hPileBox
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x + lPileBox
End If
'Now back to drawing the wing
ptB.x = ptA.x
ptB.y = elev(ptB.x) + hWingCurb
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x - lWing
ptA.y = elev(ptA.x) + hWingCurb
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw End Post
ptB.x = ptA.x
ptB.y = elev(ptB.x) + hEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + lEndPost
ptA.y = elev(ptA.x) + hEndPost
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = elev(ptB.x) + hWingCurb
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw Span
ptB.x = ptOrigin.x - Val(txtEndBentWidth.Text) / Sin(skew)
ptB.y = elev(ptB.x)
ptA.x = ptOrigin.x - lSpan
ptA.y = elev(ptA.x)
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y - hSlab
ptB.y = ptB.y - hSlab
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
```

```
ptA.y = ptA.y - hBeam
ptB.y = ptB.y - hBeam
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y + hBeam + hSlab + hBarrier
ptB.y = ptB.y + hBeam + hSlab + hBarrier
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw back joints in barrier and beam
ptA.x = ptOrigin.x - lSpan
ptA.y = elev(ptA.x)
ptB.x = ptA.x
ptB.y = ptA.y + hBarrier
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y - hSlab
ptB.y = ptA.y - hBeam
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw substructure (concrete bent)
'Draw cap
ptA.x = ptOrigin.x - 0.5 * wCap - lSpan
ptA.y = elev(ptA.x) - hSlab - hBeam - hBearing - hCap
ptB.x = ptA.x + wCap
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.y = ptA.y + hCap
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptB.x = ptA.x
ptB.y = ptA.y - hCap
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + wCap
ptB.x = ptA.x
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Draw column
If Left(cboBentType.Value, 1) = "P" Then wCap = 14 / 12
ptA.x = ptOrigin.x - 0.5 * wCap - lSpan
ptA.y = ptA.y - hCap
ptB.x = ptA.x
ptB.y = ptA.y - hCol
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
ptA.x = ptA.x + wCap
ptB.x = ptA.x
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Draw footing
If Left(cboBentType.Value, 1) <> "P" Then
    ptA.x = ptOrigin.x - 0.5 * lFoot - lSpan
    ptA.y = ptB.y
    ptB.x = ptA.x + lFoot
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    ptA.y = ptA.y - hFoot
    ptB.y = ptA.y
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
```

```
    ActiveModelReference.AddElement oLine
    ptB.x = ptA.x
    ptB.y = ptA.y + hFoot
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
    ptA.x = ptA.x + lFoot
    ptB.x = ptA.x
    Set oLine = CreateLineElement2(Nothing, ptA, ptB)
    ActiveModelReference.AddElement oLine
End If

'Draw Station and Elev Marker at back bent
SetLine ("Dimension")
ptA.x = ptOrigin.x - lSpan
ptA.y = elev(ptA.x) + hBarrier + 2 * pica
ptB.x = ptA.x
ptB.y = ptA.y + 24 * pica
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

'Add STA text above line and EL text below
ptA.x = ptA.x - pica
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftBottom
strText = "STA. " & Format(ptOrigin.x, "###+##.00")
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Pi / 2)
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText
ptA.x = ptA.x + 2 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftTop
strText = "EL. " & Format(elev(ptB.x), "##0.00")
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText
strText = "**"
ptB.y = ptB.y + 2 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
ActiveModelReference.AddElement oText

'Add Bent No in circle 0.5 inch below footing
ptB.y = ptOrigin.y - hSlab - hBeam - hBearing - hCap - hCol - hFoot - 2 - 8 * pica
ActiveModelReference.AddElement oText
strText = Format(SpanNo)
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText
SetLine ("Dimension")
Set oCircle = CreateEllipseElement2(Nothing, ptB, 2 * pica, 2 * pica,
Matrix3dIdentity)
ActiveModelReference.AddElement oCircle


'Draw Station and Elev Marker at end of bridge
SetLine ("Dimension")
ptA.x = ptOrigin.x
ptA.y = ptOrigin.y + hBarrier + 2 * pica
ptB.x = ptA.x
ptB.y = ptA.y + 24 * pica
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine
'Add STA text above line and EL text below
ptA.x = ptA.x - pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftBottom
strText = "STA. " & Format(ptOrigin.x, "###+##.00")
```

```
SetText ("Normal")
oMatrix = Matrix3dFromAxisAndRotationAngle(2, Pi / 2)
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText
ptA.x = ptA.x + 2 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftTop
strText = "EL. " & Format(elev(ptB.x), "##0.00")
Set oText = CreateTextElement1(Nothing, strText, ptA, oMatrix)
ActiveModelReference.AddElement oText
strText = "**"
ptB.y = ptB.y + 2 * pica
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, oMatrix)
ActiveModelReference.AddElement oText

'Add Bent No in circle 0.5 in below wing
ptB.y = ptOrigin.y - hSlab - hBeam - 2 - 8 * pica
strText = Format(SpanNo + 1)
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText
SetLine ("Dimension")
Set oCircle = CreateEllipseElement2(Nothing, ptB, 2 * pica, 2 * pica,
Matrix3dIdentity)
ActiveModelReference.AddElement oCircle


'Add scale note
ptB.x = ptB.x + 16 * pica
strText = strScale()
SetText ("Normal")
ActiveSettings.TextStyle.Justification = msdTextJustificationLeftCenter
Set oText = CreateTextElement1(Nothing, strText, ptB, Matrix3dIdentity)
ActiveModelReference.AddElement oText

'Put ELEVATION under elevation view
ptA.x = Val(txtBeginBr.Value) + 0.5 * BrLength()
ptA.y = elev(ptA.x) - hSlab - hBeam - hCap - hCol - hFoot - 24 * pica
SetText ("Title")
ActiveSettings.TextStyle.Justification = msdTextJustificationCenterBottom
strText = "ELEVATION"
Set oText = CreateTextElement1(Nothing, strText, ptA, Matrix3dIdentity)
ActiveModelReference.AddElement oText
ptA.x = ptA.x - 14 * pica
ptA.y = ptA.y - pica
ptB.x = ptA.x + 28 * pica
ptB.y = ptA.y
Set oLine = CreateLineElement2(Nothing, ptA, ptB)
ActiveModelReference.AddElement oLine

End Sub

Function strDim(dDim As Double) As String

    Dim denom As Double     'level of precision for fractions (8 for 1/8)
    Dim nbrft As Double     'whole number of feet
    Dim inchin As Double    'decimal number of inches
    Dim nbrin As Double     'whole number of inches
    Dim fracin As Double    'decimal number of eighths (or whatever fractions it
is)
    Dim numer As Double     'numerator in fraction
    Dim frac As String      'string to hold the fraction

    denom = 8                           'must be 2, 4, 8, 16, 32, 64, etc.
```

```
    nbrft = Int(dDim)                   'whole number of feet
    inchin = 12 * (dDim - nbrft)        'fractional number of inches
    nbrin = Int(inchin)                 'whole number of inches
    fracin = denom * (inchin - nbrin)
    numer = Round(fracin)
    If numer = 0 Then                   'if numerator is 0, no fraction
        frac = ""
    ElseIf numer = denom Then           'if numerator = denominator, it is 1 inch
        nbrin = nbrin + 1
        frac = ""
    Else
        Do                              'loop to reduce fractions while numerator
is even
            If numer / 2 = Int(numer / 2) Then
                numer = numer / 2
                denom = denom / 2
            Else
                frac = " " & Format(numer) & "/" & Format(denom)
                Exit Do
            End If
        Loop
    End If
    strDim = Format(nbrft) & "'-"
    If nbrft = 0 Then strDim = ""
    strDim = strDim & Format(nbrin) & frac & """"

End Function

Function CellExists(cellName As String) As Boolean

  CellExists = False
  If IsCellLibraryAttached Then
    Dim celEnum As CellInformationEnumerator

    ' Get list of cells
    Set celEnum = GetCellInformationEnumerator(True, False)
    ' Loop through cells
    Do While celEnum.MoveNext
      If celEnum.Current.Name = cellName Then
        ' Found the cell so return True
        CellExists = True
        Exit Function
      End If
    Loop
    ' Cell not found
    MsgBox "Cell " & cellName & " Not Found"
  Else
    ' No library attached
    MsgBox "No Cell Library Attached"
  End If

End Function

Function strAngle(angle As Double) As String

Dim Degree As Double    'whole number of degrees
Dim Minute As Double    'whole number of minutes
Dim Second As Double    'decimal number of seconds
Dim dAngle As Double    'angle in decimal degrees

dAngle = Degrees(angle + 0.00000001)
'Set degree to Integer of Argument Passed
Degree = Int(dAngle)
```

```
'Set minutes to 60 times the number to the right
'of the decimal for the variable Decimal_Deg
Minute = (dAngle - Degree) * 60

'Set seconds to 60 times the number to the right of the
'decimal for the variable Minute
Second = 60 * (Minute - Int(Minute))

'Returns the Result of degree conversion
'(for example, 10.46 = 10~ 27  ' 36")
strAngle = Format(Degree, "0") & "^-" & Format(Int(Minute), "00") _
    & "'-" & Format(Second, "00.0") + Chr(34)

End Function


Function BrLength() As Double

BrLength = Val(txtSpan1.Value)

If Val(txtNumSpans.Value) > 1 Then BrLength = BrLength + Val(txtSpan2.Value)
If Val(txtNumSpans.Value) > 2 Then BrLength = BrLength + Val(txtSpan3.Value)
If Val(txtNumSpans.Value) > 3 Then BrLength = BrLength + Val(txtSpan4.Value)
If Val(txtNumSpans.Value) > 4 Then BrLength = BrLength + Val(txtSpan5.Value)
If Val(txtNumSpans.Value) > 5 Then BrLength = BrLength + Val(txtSpan6.Value)
If Val(txtNumSpans.Value) > 6 Then BrLength = BrLength + Val(txtSpan7.Value)
If Val(txtNumSpans.Value) > 7 Then BrLength = BrLength + Val(txtSpan8.Value)

End Function

Function strScale() As String

Dim dScale As Double

dScale = Round(Val(txtScale.Value), 0)

Select Case dScale
    Case 16
        strScale = "3/4 " & """ & " = 1'-0" & """
    Case 24
        strScale = "1/2 " & """ & " = 1'-0" & """
    Case 32
        strScale = "3/8 " & """ & " = 1'-0" & """
    Case 48
        strScale = "1/4 " & """ & " = 1'-0" & """
    Case 64
        strScale = "3/16 " & """ & " = 1'-0" & """
    Case 96
        strScale = "1/8 " & """ & " = 1'-0" & """
    Case 120
        strScale = "1" & """ & " = 10'-0" & """
    Case 240
        strScale = "1" & """ & " = 20'-0" & """
    Case 360
        strScale = "1" & """ & " = 30'-0" & """
    Case 480
        strScale = "1" & """ & " = 40'-0" & """
    Case 600
        strScale = "1" & """ & " = 50'-0" & """
    Case 720
        strScale = "1" & """ & " = 60'-0" & """
    Case 120
        strScale = "1" & """ & " = 10'-0" & """
    Case Else
```

```
        strScale = "1:" & dScale
    End Select
strScale = "SCALE:  " & strScale

End Function
```