```
'***This code places either overall annotations or all annotations

'Declare main objects: sheet, drawing views, annotations and geometry intents

'Get sheet and drawing views
Dim Sheet_1 = ThisDrawing.Sheets.ItemByName("Sheet:1")
Dim FrontView = Sheet_1.DrawingViews.ItemByName("Front")
Dim SideView = Sheet_1.DrawingViews.ItemByName("Side")
Dim SectionView = Sheet_1.DrawingViews.ItemByName("A")
'Get general dimensions and hole thread notes
Dim genDims = Sheet_1.DrawingDimensions.GeneralDimensions
Dim holeThreadNotes = Sheet_1.DrawingNotes.HoleThreadNotes

'[Geometry Intents to be used on the Front View
'Geometry Edge, No Point Intent
Dim LowerEdgeGI = FrontView.GetIntent("LowerEdge")
'Geometry Cylindrical Face, No Point Intent
Dim Hole_FView As GeometryIntent
If Parameter("1-iLogic Attr_Annotations.ipt.HoleInclude") = True
          Hole_FView = FrontView.GetIntent("HoleCylFace")
End If

'GeomInt for WorkPoint3 and 4
Dim WPo2_FV = FrontView.GetIntent("Hole TopQuad_FrontWPo2")
Dim WPo3_FV = FrontView.GetIntent("TopRight_FrontWPo3")
']

'[Geometry Intents to be used on the Side View
'GeomInt for Face
Dim RightFaceInt = SideView.GetIntent("RightFace")

Dim WPo3_SV = SideView.GetIntent("TopRight_FrontWPo3")
Dim WPo4_SV = SideView.GetIntent("TopRight_RearWPo4")
'Geometry Edge, Intent MidPoint of the Edge
Dim FrontRightEdgeInt = SideView.GetIntent("RV_FrontEdge", PointIntentEnum.kMidPointIntent)
Dim RearRightEdgeInt = SideView.GetIntent("RV_RearEdge", PointIntentEnum.kMidPointIntent)
']

'[Geometry Intents to be used on the Section View
Dim oSV_CLineInt1 As GeometryIntent
Dim oSV_CLineInt2 As GeometryIntent
If Parameter("1-iLogic Attr_Annotations.ipt.HoleInclude") = True
          oSV_CLineInt1 = SectionView.GetIntent("FrontCircEdge", PointIntentEnum.kCircularLeftPointIntent)
          oSV_CLineInt2 = SectionView.GetIntent("RearCircEdge", PointIntentEnum.kCircularLeftPointIntent)
End If
']

'Place Annotations
ThisDrawing.BeginManage()

          'Create overall dimensions
          Dim LinearDim1 = genDims.AddLinear("WidthDim", FrontView.SheetPoint(0.5, -0.1), LowerEdgeGI)
          Dim LinearDim2 = genDims.AddLinear("HeightDim", SideView.SheetPoint(-0.25, 0.5),
                              RightFaceInt,,DimensionTypeEnum.kVerticalDimensionType)
          Dim LinearDim3 = genDims.AddLinear("DepthDim", SideView.SheetPoint(0.5, 1.25), WPo3_SV, WPo4_SV)

          'Create centerlines
          If Parameter("1-iLogic Attr_Annotations.ipt.HoleInclude") = True

                              Dim centermark = Sheet_1.Centermarks.Add("Hole Centermark", Hole_FView)
                              Dim centerline = Sheet_1.Centerlines.Add("Hole Centerline", {FrontRightEdgeInt,
                                                                  RearRightEdgeInt} )
                              Dim Section_centerline = Sheet_1.Centerlines.Add("SV_Centerline", {oSV_CLineInt1,
                                                                        oSV_CLineInt2 } )

          End If

          'Place detailed dimensions
          If Dimensions = "All Dimensions"

                    'Angle
                    Dim angDim1 = genDims.AddAngular("AngularDim",
                                                  ThisDrawing.Geometry.Point2d(2.5, 6.5),
                                                  FrontView.GetIntent("InclinedEdge"),
                                                  FrontView.GetIntent("TopEdge"))

                    'Add fillet radius when unsuppressed
                    If Parameter("1-iLogic Attr_Annotations.ipt.FilletInclude") = True

                              Dim filletDim = genDims.AddRadius("Fillet Radius", FrontView.SheetPoint(-0.1, 0.5),
                                                  FrontView.GetIntent("FilletFace"))
```

```
                End If

                'Add hole note and dimensions when unsuppressed
                If Parameter("1-iLogic Attr_Annotations.ipt.HoleInclude") = True

                        Dim holeNotePt1 = FrontView.SheetPoint(.7,1.1)
                        Dim holeNote1 = holeThreadNotes.Add("Hole Note 1", holeNotePt1, Hole_FView)
                        'Horizontal location of the hole
                        Dim HoleHorizLoc = genDims.AddLinear("Hole Horiz Loc", FrontView.SheetPoint(0.75,
                                1.16), WPo2_FV, WPo3_FV, DimensionTypeEnum.kHorizontalDimensionType)

                End If

        End If

ThisDrawing.EndManage()
```

```
'***This code creates an attribute on the top face

'Get the part document
Dim oDoc As PartDocument = ThisApplication.ActiveDocument
'Get the first extrusion
Dim oExt As ExtrudeFeature = oDoc.ComponentDefinition.Features(1)
'Get the end faces (There is only one)
Dim oEndFaces As Faces = oExt.EndFaces
Dim oEndFace As Face = oEndFaces(1)

'Define an attribute set
Dim oAttSet As AttributeSet = oEndFace.AttributeSets.Add("General")
'Define an attribute name and value
Dim oAtt As Attribute = oAttSet.Add("Name", kStringType, "Top Face")
```

```
'***This code places a dimension
'***It returns drawing curves by finding geometry entities (face and edge) with specific attributes

'Gets main objects: document, sheet and drawing views
Dim oTG As TransientGeometry = ThisApplication.TransientGeometry
Dim oDoc As DrawingDocument = ThisApplication.ActiveDocument
Dim oSheet As Sheet = oDoc.Sheets(1)
Dim oViews As DrawingViews = oSheet.DrawingViews
Dim oFrontView As DrawingView = oViews(1)
Dim oSideView As DrawingView = oViews(3)

'Get the referenced model and its occurrences
Dim oAssyModel As AssemblyDocument = oFrontView.ReferencedDocumentDescriptor.ReferencedDocument
Dim oAssyOccs As ComponentOccurrences = oAssyModel.ComponentDefinition.Occurrences
Dim oSliderOcc As ComponentOccurrence = oAssyOccs.ItemByName("Slider")
Dim oLowerCapOcc As ComponentOccurrence = oAssyOccs.ItemByName("Lower Cap")
Dim oSliderDoc As PartDocument = oSliderOcc.Definition.Document
Dim oCapDoc As PartDocument = oLowerCapOcc.Definition.Document

'Declares an object collection and drawing curves
Dim oObjs As ObjectCollection
Dim oDrawingCurves As DrawingCurvesEnumerator

'Finds a named face of a part
oObjs = oAssyModel.AttributeManager.FindObjects("General", "Name", "Slider Lower Face")
Dim oSlideFace As Face = oObjs.Item(1)

'Proxy of the Lower Cap Edge to get the representation in the Assembly Level
Dim oSlideFaceProxy As FaceProxy
oSliderOcc.CreateGeometryProxy(oSlideFace, oSlideFaceProxy)
'Gets the drawing curve that represents the face on the side view
oDrawingCurves = oSideView.DrawingCurves(oSlideFaceProxy)
Dim oSliderLowerCurve As DrawingCurve = oDrawingCurves(1)
'Creates a geometry intent object to attach the annotation
Dim oUpperCurveGI As GeometryIntent = oSheet.CreateGeometryIntent(oSliderLowerCurve, 0.5)

'Finds a named edge of a part
oObjs = oCapDoc.AttributeManager.FindObjects("General", "Name", "Cap Lower Edge")
Dim oLCapEdge As Edge = oObjs.Item(1)
'Proxy of the Lower Cap Edge to get the representation in the Assembly Level
Dim oLCapEdgeProxy As EdgeProxy
oLowerCapOcc.CreateGeometryProxy(oLCapEdge, oLCapEdgeProxy)
'Gets the drawing curve that represents the edge on the side view
oDrawingCurves = oSideView.DrawingCurves(oLCapEdgeProxy)
Dim oCapLowerCurve As DrawingCurve = oDrawingCurves(1)
'Creates a geometry intent object to attach the annotation
Dim oLowerCurveGI As GeometryIntent = oSheet.CreateGeometryIntent(oCapLowerCurve)

'Creates the point to place the linear dimension and the linear dimension
Dim oDWGDims As DrawingDimensions = oSheet.DrawingDimensions
Dim oGenDims As GeneralDimensions = oDWGDims.GeneralDimensions

Dim oSliderPos As Point2d = oTG.CreatePoint2d(oSideView.Position.X + oSideView.Width / 2 + 2, _
(oSideView.Position.Y - oSideView.Height/2 + (oSliderLowerCurve.EndPoint.Y - oCapLowerCurve.EndPoint.Y) / 2))

Call oGenDims.AddLinear(oSliderPos, oUpperCurveGI, oLowerCurveGI, kVerticalDimensionType)
```

```
'***This code places a dimension using workpoints
'***It creates centermarks using workpoints from models

'Gets main objects: document, sheet and drawing views
Dim oTG As TransientGeometry = ThisApplication.TransientGeometry
Dim oDoc As DrawingDocument = ThisApplication.ActiveDocument
Dim oSheet As Sheet = oDoc.Sheets(1)
Dim oViews As DrawingViews = oSheet.DrawingViews
Dim oFrontView As DrawingView = oViews(1)
Dim oSideView As DrawingView = oViews(3)

'Gets/Creates the centermark styles
Dim oStylesManager As DrawingStylesManager
oStylesManager=ThisApplication.ActiveDocument.StylesManager
Dim oCMStyleForDims As CentermarkStyle
Try
          oCMStyleForDims = oStylesManager.CentermarkStyles.Item("Center Mark (for AutoDims)")
Catch
          oRefCMStyle = oStylesManager.CentermarkStyles(1)
          oCMStyleForDims = oRefCMStyle.Copy("Center Mark (for AutoDims)")
          oCMStyleForDims.MarkSize = 0.001
          oCMStyleForDims.GapDistance = 0.001
          oCMStyleForDims.OvershootDistance = 0.001
          oCMStyleForDims.ExtensionLength = 0.001
          oCMStyleForDims.DefaultRadius = 0.001
End Try

'Get the referenced model and its occurrences
Dim oAssyModel As AssemblyDocument = oFrontView.ReferencedDocumentDescriptor.ReferencedDocument
Dim oAssyOccs As ComponentOccurrences = oAssyModel.ComponentDefinition.Occurrences
Dim oUpperCapOcc As ComponentOccurrence = oAssyOccs.ItemByName("Upper Cap")
Dim oCapDoc As PartDocument = oUpperCapOcc.Definition.Document

'Declares dimensions
Dim oDWGDims As DrawingDimensions = oSheet.DrawingDimensions
Dim oGenDims As GeneralDimensions = oDWGDims.GeneralDimensions

'Gets the second workpoint of the assembly
Dim oAssyWorkPoints As WorkPoints = oAssyModel.ComponentDefinition.WorkPoints
Dim oArmTopWPo As WorkPoint = oAssyWorkPoints.Item(2)
'Gets the second workpoint of an occurrence
Dim oCapWorkPoints As WorkPoints = oCapDoc.ComponentDefinition.WorkPoints
Dim oCapMidWPo As WorkPoint = oCapWorkPoints(2)

Dim oCapMidWPoProxy As WorkPointProxy
oUpperCapOcc.CreateGeometryProxy(oCapMidWPo, oCapMidWPoProxy)

'Creates centermarks using workpoints
Dim oCenterMarks = oSheet.Centermarks

Dim oCMark1 As Centermark = oCenterMarks.AddByWorkFeature(oArmTopWPo, oSideView)
Dim oCMark2 As Centermark = oCenterMarks.AddByWorkFeature(oCapMidWPoProxy, oSideView)

oCMark1.Style = oCMStyleForDims
oCMark2.Style = oCMStyleForDims

oCMark1.Visible = False
oCMark2.Visible = False
'Creates geometry intent objects to attach the linear dimension
Dim oGI1 As GeometryIntent = oSheet.CreateGeometryIntent(oCMark1)
Dim oGI2 As GeometryIntent = oSheet.CreateGeometryIntent(oCMark2)
'Creates the point to place the linear dimension
Dim oDimPos As Point2d = oTG.CreatePoint2d((oCMark1.Position.X+oCMark2.Position.X)/2 , oCMark2.Position.Y + 1)
'Creates a linear dimension
Call oGenDims.AddLinear(oDimPos, oGI1, oGI2, kHorizontalDimensionType)
```

```
'***This code places a linear dimension using points obtained from the 3D model and transferred to the sheet
'***Note: This code uses Sheet.FindUsingPoint(Point2D, [Proximity Tolerance]) to find points from 3D to 2D sheet
'and find its related object (e.g. drawing curve segment)
'***FindUsingPoint has a limitation. It uses underlying selection functionality that Inventor uses when doing
selections interactively.
'***This relies on the graphics being available because the initial selection is made to the graphics and then maps
back to the associated entity

Sub Main
	'Gets main objects: document, sheet and drawing views
	Dim oTG As TransientGeometry = ThisApplication.TransientGeometry
	Dim oDoc As DrawingDocument = ThisApplication.ActiveDocument
	Dim oSheet As Sheet = oDoc.Sheets(1)
	Dim oViews As DrawingViews = oSheet.DrawingViews
	Dim oFrontView As DrawingView = oViews(1)
	Dim oSideView As DrawingView = oViews(3)
	'Gets the referenced model
	Dim oAssyModel As AssemblyDocument = oFrontView.ReferencedDocumentDescriptor.ReferencedDocument
	'Gets the max and min point of the 3D model volume
	Dim oAssyMin As Point = oAssyModel.ComponentDefinition.RangeBox.MinPoint
	Dim oAssyMax As Point = oAssyModel.ComponentDefinition.RangeBox.MaxPoint
	'Clean existing dimensions
	Dim oDWGDims As DrawingDimensions = oSheet.DrawingDimensions
	Dim oGenDims As GeneralDimensions = oDWGDims.GeneralDimensions
	Dim oGenDim As GeneralDimension
	For Each oGenDim In oGenDims
		oGenDim.Delete
	Next

	'Front View - Top Left Corner
	Dim oTLeftEdgeX_3DPo As Point = oTG.CreatePoint(oAssyMin.X, oAssyMax.Y, oAssyMin.Z)
	Dim oTLeftEdgeX_2DPo As Point2d
	Dim oTLeftGInt As GeometryIntent
	GetSheetPo_n_GeomInt(oSheet, oFrontView, _
			oTLeftEdgeX_3DPo, oTLeftEdgeX_2DPo, oTLeftGInt)

	'Front View - Top Right Corner
	Dim oTRightEdgeX_3DPo As Point = oTG.CreatePoint(oAssyMax.X, oAssyMax.Y, oAssyMin.Z)
	Dim oTRightEdgeX_2DPo As Point2d
	Dim oTRightGInt As GeometryIntent
	GetSheetPo_n_GeomInt(oSheet, oFrontView, _
			oTRightEdgeX_3DPo, oTRightEdgeX_2DPo, oTRightGInt)

	'Front View - Lower Left Corner
	Dim oBLeftEdgeX_3DPo As Point = oTG.CreatePoint(oAssyMax.X, oAssyMin.Y, oAssyMin.Z)
	Dim oBLeftEdgeX_2DPo As Point2d
	Dim oBLeftGInt As GeometryIntent
	GetSheetPo_n_GeomInt(oSheet, oFrontView, _
			oBLeftEdgeX_3DPo, oBLeftEdgeX_2DPo, oBLeftGInt)

	'***SIDE VIEW
	'SideView - Top Right Corner
	Dim oTR_SV_EdgeX_3DPo As Point = oTG.CreatePoint(oAssyMax.X, oAssyMax.Y, oAssyMin.Z)
	Dim oTR_SV_EdgeX_2DPo As Point2d
	Dim oTR_SV_GInt As GeometryIntent
	GetSheetPo_n_GeomInt(oSheet, oSideView, _
			oTR_SV_EdgeX_3DPo, oTR_SV_EdgeX_2DPo, oTR_SV_GInt)

	'Side View - Left Edge
	Dim oFront_SV_EdgeX_3DPo As Point = oTG.CreatePoint(oAssyMax.X, 53.8, 35.65)
	Dim oFront_SV_EdgeX_2DPo As Point2d
	Dim oFront_SV_GInt As GeometryIntent
	GetSheetPo_n_GeomInt(oSheet, oSideView, _
			oFront_SV_EdgeX_3DPo, oFront_SV_EdgeX_2DPo, oFront_SV_GInt)

	'Width
	Dim DimWidthPo As Point2d = oTG.CreatePoint2d((oTLeftEdgeX_2DPo.X + (oTRightEdgeX_2DPo.X -
oTLeftEdgeX_2DPo.X)/2), oTRightEdgeX_2DPo.Y + 2)
	oGenDims.AddLinear(DimWidthPo, oTLeftGInt, oTRightGInt, kHorizontalDimensionType)

	'Height
	Dim DimHeightPo As Point2d = oTG.CreatePoint2d(oTLeftEdgeX_2DPo.X - 2, (oTLeftEdgeX_2DPo.Y  -
(oTLeftEdgeX_2DPo.Y - oBLeftEdgeX_2DPo.Y)/2))
	oGenDims.AddLinear(DimHeightPo, oTLeftGInt, oBLeftGInt, kVerticalDimensionType)

	'Depth
	Dim DimDepthPo As Point2d = oTG.CreatePoint2d((oFront_SV_EdgeX_2DPo.X + (oTR_SV_EdgeX_2DPo.X -
oFront_SV_EdgeX_2DPo.X)/2), oTR_SV_EdgeX_2DPo.Y + 2)
	oGenDims.AddLinear(DimDepthPo,oFront_SV_GInt,oTR_SV_GInt, kHorizontalDimensionType)
```

```
End Sub

'Subprocedure to sheet point and the geometry intent object
Sub GetSheetPo_n_GeomInt(oSheet As Sheet, oView As DrawingView, _
                    ModelPo As Point, ByRef SheetPo As Point2d, ByRef oGI As GeometryIntent)

          'Get a sheet point using a 3D model point
          SheetPo = oView.ModelToSheetSpace(ModelPo)

          'Method that finds drawing curve segments, entities on a sheet sketch, centerlines and centermarks that
          the given point lies on.
          Dim oItemsFoundAtPoint As ObjectsEnumerator = oSheet.FindUsingPoint(SheetPo, 0.001)

          Dim oCurvSegAtPo As DrawingCurveSegment
          Dim oCurvAtPo As DrawingCurve

          Dim oItemNoFAPo As Integer

          For oItemNoFAPo = 1 To oItemsFoundAtPoint.Count

                    If oItemsFoundAtPoint.Item(oItemNoFAPo).Type = kDrawingCurveSegmentObject
                              oCurvSegAtPo = oItemsFoundAtPoint.Item(oItemNoFAPo)
                              If Round(oCurvSegAtPo.StartPoint.X,2) = Round(oCurvSegAtPo.EndPoint.X,2)
                                        oCurvAtPo = oCurvSegAtPo.Parent
                                        oGI = oSheet.CreateGeometryIntent(oCurvAtPo, SheetPo)
                                        Exit For
                              End If
                    End If

          Next

End Sub
```

```
'PLACE AUTOMATED CENTERLINES

'Gets the active document - drawing
Dim oDoc As DrawingDocument = ThisApplication.ActiveDocument

'Gets drawing settings
Dim oDrawingSettings As DrawingSettings
oDrawingSettings = oDoc.DrawingSettings

'Get the sheet
Dim oSheet As Sheet = oDoc.Sheets(1)

'Clean Centerlines
Dim oCenterlines As Centerlines = oSheet.Centerlines
Dim oCenterline As Centerline
For Each oCenterline In oCenterlines
          oCenterline.Delete
Next
'Clean Centermarks
Dim oCentermarks As Centermarks = oSheet.Centermarks
Dim oCentermark As Centermark
For Each oCentermark In oCentermarks
          oCentermark.Delete
Next

'Get the drawing views collection
Dim oDrawingViews As DrawingViews = oSheet.DrawingViews

Dim oFrontView As DrawingView = oDrawingViews.Item(1)
Dim oSecView As DrawingView = oDrawingViews.Item(2)

'Get the automated centerline settings object
Dim oAutoCenterlines As AutomatedCenterlineSettings
oAutoCenterlines = oDrawingSettings.AutomatedCenterlineSettings

'Select the automated centerline settings to place in the views
oAutoCenterlines.ApplyToRevolutions = True
oAutoCenterlines.ApplyToCylinders = True
oAutoCenterlines.ApplyToHoles = True
oAutoCenterlines.ApplyToCircularPatterns = True
oAutoCenterlines.ProjectionNormalAxis = True

'Front View - Sets the Automated Centerlines
oFrontView.SetAutomatedCenterlineSettings(oAutoCenterlines)

'Side View - Sets the Automated Centerlines
oAutoCenterlines.ProjectionNormalAxis = False
oAutoCenterlines.ProjectionParallelAxis = True

oSecView.SetAutomatedCenterlineSettings(oAutoCenterlines)
```

```
'Retrieve all dimensions
Dim oSheet As Sheet = ThisApplication.ActiveDocument.Sheets(1)
Dim oFrontView As DrawingView = oSheet.DrawingViews(1)
Dim oSideView As DrawingView = oSheet.DrawingViews(2)
Dim oGenDims As GeneralDimensions = oSheet.DrawingDimensions.GeneralDimensions

oGenDims.Retrieve(oFrontView)
oGenDims.Retrieve(oSideView)
```