Algorithm for Computing the Sheet Code (XY → code) (pseudocode, VB6-friendly)

```
Input: X, Y (Double), targetScale (Long) ∈ {10000, 5000, 2000, 1000, 500}

1) If point outside PL-2000 domain → "Point outside PL-2000"
2) X ← X + ε : Y ← Y + ε
3) A = CLng(Y) \ 1000000
   BBB = Int((X/1000# - 4920#) / 5#)
   CC  = Int(((CLng(Y) Mod 1000000) / 1000# - 332#) / 8#)
   code = CStr(A) & "." & CStr(BBB) & "." & Format$(CC, "00")
4) If targetScale = 5000 Then code = code & "." & Index2x2(X, Y, 5000)
5) If targetScale ≤ 2000 Then code = code & "." & Index5x5(X, Y)
   If targetScale ≤ 1000 Then code = code & "." & Index2x2(X, Y, 1000)
   If targetScale = 500  Then code = code & "." & Index2x2(X, Y, 500)
6) Return code
```

Algorithm for Reconstructing the Sheet Extent (code → min–max) (pseudocode, VB6-friendly)

```
1) If invalid pattern/length → error/False
2) G, scale ← ParseSheetCode(sheetCode)           ' digits G(1..n) and scale
3) A = G(1): BBB = G(2)*100 + G(3)*10 + G(4): CC = G(5)*10 + G(6)
4) X_top  = (BBB*5 + 4925) * 1000
   Y_left = (CC*8 + 332) * 1000 + A * 1000000
5) If scale = 5000 Then
       Apply2x2(X_top, Y_left, G(7), 5000)
6) If scale ≤ 2000 Then
       SS    = G(7)*10 + G(8)
       X_top = X_top - ((SS - 1) \ 5) * 1000
       Y_left= Y_left + ((SS - 1) Mod 5) * 1600
       If scale ≤ 1000 Then Apply2x2(X_top, Y_left, G(9), 1000)
       If scale =  500 Then Apply2x2(X_top, Y_left, G(10), 500)
7) PminX = X_top - scale*0.5 : PminY = Y_left
   PmaxX = X_top : PmaxY = Y_left + scale*0.8
```

Algorithm ready-to-implement in the MicroStation environment

Implementation note (MicroStation axes).

The VB6/VBA listings are direct, working copies from MicroStation CONNECT, where the Point3d axes follow CAD screen orientation: X = easting (horizontal), Y = northing (vertical), Z = height. Throughout the paper we adopt the PL-2000 convention X—northing, Y—easting. Accordingly, in the listings the variables map as paper X ↔ listing Y and paper Y ↔ listing X. The arithmetic, boundary rules (including ε-perturbation), and scale logic are identical—only the axis labels differ. This is intentional and not an error.

Listing S1. VB6/VBA module: PL-2000 forward (XY->code).

```
Option Explicit
' --- Only for systems where there is no definition of the Point3d
structure.
   ' Type Point3d
     ' x as double
     ' y as double
     ' z as double
```

```
    ' End Type
' MicroStation environment note:
' Point3d axes follow CAD screen orientation (X = easting, Y = northing).
' NOTE: In this validator, the northing range is checked against point.X
' and the easting (zone million) against point.Y, by design.
' This matches the listings' axis usage and boundary rules.
' --- Module for converting XY coordinates to sheet code---
' --- Auxiliary Function: 1:10,000 Map Sheet Code ---
Function SheetPL2000Code10k(point As Point3d) As String
    Dim millionZone As Integer, rowIndex As Integer, columnIndex As Integer

    ' Determine the PL-2000 million zone (e.g., 5, 6, 7, 8),
corresponding to full millions in X
    millionZone = point.x \ 1000000

    ' Calculate the row index: from 4920 km northward, divided every 5 km
    rowIndex = Int((point.y * 0.001 - 4920) / 5)

    ' Calculate the column index: from 332 km eastward, divided every 8 km
    columnIndex = Int(((point.x Mod 1000000) * 0.001 - 332) / 8)

    ' Compose the final code: A.BBB.CC, where C is two-digit
    SheetPL2000Code10k = CStr(millionZone) & "." & CStr(rowIndex) & "."
& Format$(columnIndex, "00")
End Function

Function SheetPL2000Code5x5(point As Point3d) As String
    Dim basePoint As Point3d
    Dim rowIndex As Integer, columnIndex As Integer, indexNumber As Integer

    ' Determine base point X (division every 8000 m, shifted by 4000 m)
    basePoint.x = Int(point.x / 8000) * 8000
    If point.x - basePoint.x < 4000 Then
        basePoint.x = basePoint.x - 4000
    Else
        basePoint.x = basePoint.x + 4000
    End If

    ' Determine base point Y as the upper edge of the 5 km section
    basePoint.y = (Int(point.y / 5000) + 1) * 5000

    ' Calculate row index (section height = 1 km)
    rowIndex = Int((basePoint.y - point.y) / 1000)

    ' Calculate column index (section width = 1.6 km)
    columnIndex = Int((point.x - basePoint.x) / 1600)

    ' Compute section index in the 5x5 grid (01 to 25)
```

```
    indexNumber = rowIndex * 5 + columnIndex + 1

    ' Format index as two-digit string
    SheetPL2000Code5x5 = Format$(indexNumber, "00")
End Function

Function SheetPL2000Code2x2(scaleS As Long, point As Point3d) As String
    Dim basePoint As Point3d
    Dim widthX As Double, offsetX As Double
    Dim rowIndex As Integer, columnIndex As Integer, indexNumber As Integer

    ' Set X offset depending on scale
    Select Case scaleS
        Case 1000: offsetX = 800
        Case 5000: offsetX = 4000
        Case Else: offsetX = 0
    End Select

    ' Set sheet width in X as 1.6 * scale
    widthX = scaleS * 1.6

    ' Determine base point X in the grid with possible offset
    basePoint.x = Int(point.x / widthX) * widthX
    If point.x - basePoint.x < offsetX Then
        basePoint.x = basePoint.x - offsetX
    Else
        basePoint.x = basePoint.x + offsetX
    End If

    ' Set upper boundary Y of the sheet
    basePoint.y = (Int(point.y / scaleS) + 1) * scaleS

    ' Calculate row number (0 or 1) - section height = 0.5 * scale
    rowIndex = Int((basePoint.y - point.y) / (scaleS * 0.5))

    ' Calculate column number (0 or 1) - section width = 0.8 * scale
    columnIndex = Int((point.x - basePoint.x) / (scaleS * 0.8))

    ' Calculate sheet number: 1..4
    indexNumber = rowIndex * 2 + columnIndex + 1

    ' Return index as string
    SheetPL2000Code2x2 = CStr(indexNumber)
End Function

' --- Helper function: Check if the point lies within the PL-2000
coordinate system ---
' --- MicroStation environment note: ---
```

```
' --- Point3d axes follow CAD screen orientation (X = easting, Y =
northing).---
' --- The range checks below are written accordingly.---
Function IsPointInPl2000(point As Point3d) As Boolean
    Dim zone As Integer
    If point.y > 5420000 And point.y < 6100000 Then
        For zone = 5 To 8
            If point.x > 364000 + (zone * 1000000) And point.x < 636000
+ (zone * 1000000) Then
                IsPointInPl2000 = True
                Exit Function
            End If
        Next zone
    End If
    IsPointInPl2000 = False
End Function


' --- Main function: Generate map sheet code based on coordinates and
map scale ---
Function SheetPL2000CodeFromXY(targetScale As Integer, point As Point3d)
As String
    Const EPSILON As Double = 0.00000001
    Dim sheetCode As String

    ' Check if the point lies within the PL-2000 coordinate system
    If Not IsPointInPl2000(point) Then
        SheetPL2000CodeFromXY = "Point outside PL-2000"
        Exit Function
    End If

    ' Adjust coordinates for section boundaries (eliminates
rounding/boundary errors)
    point.x = point.x + EPSILON
    point.y = point.y + EPSILON

    ' Calculate the main part of the map sheet code (A.BBB.CC) in the
1:10,000 grid
    sheetCode = SheetPL2000Code10k(point)

    ' Expand the code according to the hierarchical map scale structure:
    ' Depending on the target scale, the code is extended with
additional components.
    ' This is a cascading structure: each finer scale provides more
detailed localization.
        If targetScale <= 2000 Then
            sheetCode = sheetCode & "." & SheetPL2000Code5x5(point)
        ' divide 1:10,000 into 25 sections
```

```
        If targetScale <= 1000 Then sheetCode = sheetCode & "." &
SheetPL2000Code2x2(1000, point) ' divide 1:2000 into 4
        If targetScale = 500 Then sheetCode = sheetCode & "." &
SheetPL2000Code2x2(500, point)   ' divide 1:1000 into 4
      ElseIf targetScale = 5000 Then
        sheetCode = sheetCode & "." & SheetPL2000Code2x2(5000,
point)      ' divide 1:10,000 into 4 sections
      End If

   ' Return the final hierarchical code (e.g., 6.122.15.03.2.1)
   SheetPL2000CodeFromXY = sheetCode
End Function



'Listing S2. VB6/VBA module: inverse (code->bounds) algorithms.

' --- Module for converting a map sheet code (godło) into coordinate
bounds ---
Function ParseSheetPL2000Code(sheetCode As String) As Long()
   Dim cleanCode As String
   Dim i As Integer, g() As Long

   ' Remove dots to get a sequence of digits
   cleanCode = Replace(sheetCode, ".", "")

   ' Allocate memory for array g(): one for each digit + one for scale
(g(0))
   ReDim g(Len(cleanCode))

   ' Fill array with digits from the code
   For i = 1 To UBound(g())
      g(i) = CInt(Mid$(cleanCode, i, 1))
   Next i

   ' Determine scale based on the number of digits
   i = UBound(g())
   Select Case i
      Case 6: g(0) = 10000
      Case 7: g(0) = 5000
      Case 8: g(0) = 2000
      Case 9: g(0) = 1000
      Case 10: g(0) = 500
      Case Else: g(0) = -1
   End Select

   ParseSheetPL2000Code = g
End Function
```

```
' --- Applies a 2x2 subdivision based on index and scale ---
Sub ApplySubdivisionPL20002x2(ByRef point As Point3d, ByVal index As
Long, ByVal mapScale As Long)
    If index > 2 Then point.y = point.y - mapScale * 0.5
    If index = 2 Or index = 4 Then point.x = point.x + mapScale * 0.8
End Sub

' --- Main: Calculate bounding box based on hierarchical code ---
Sub GetSheetPL2000BoundsFromCode(sheetCode As String, Pmin As Point3d,
Pmax As Point3d)
    Dim g() As Long
    Dim mapScale As Long
    Dim basePoint As Point3d

    'checking the correctness of the emblem components, we do not
include the code
    'If Not IsValidSheetCode(sheetCode) Then Exit Sub

    g = ParseSheetPL2000Code(sheetCode)
    mapScale = g(0)
    If mapScale = -1 Then Exit Sub

    ' Calculate base point (top-left corner) for 1:10,000 sheet
    basePoint.y = ((g(2) * 100 + g(3) * 10 + g(4)) * 5 + 4925) * 1000
    basePoint.x = ((g(5) * 10 + g(6)) * 8 + 332) * 1000 + g(1) * 1000000

Debug.Print basePoint.x; basePoint.y

        If mapScale <= 2000 Then
           Dim sectionIndex25 As Long
           sectionIndex25 = g(7) * 10 + g(8)
           basePoint.y = basePoint.y - ((sectionIndex25 - 1) \ 5) * 1000
           basePoint.x = basePoint.x + ((sectionIndex25 - 1) Mod 5) * 1600
           If mapScale <= 1000 Then ApplySubdivisionPL20002x2
basePoint, g(9), 1000
           If mapScale = 500 Then ApplySubdivisionPL20002x2 basePoint,
g(10), 500
        ElseIf mapScale = 5000 Then
          ApplySubdivisionPL20002x2 basePoint, g(7), 5000
        End If

    ' Compute sheet bounds (lower-left and upper-right corners)
    Pmin.x = basePoint.x
    Pmin.y = basePoint.y - mapScale * 0.5
    Pmax.x = basePoint.x + mapScale * 0.8
    Pmax.y = basePoint.y
End Sub
```