

```
Public Class CL2GcodeII
```

```
' *****
'
' Version 6.2.13.83
' George Patterson
' GSS
'
' 03-16-2018
' Revised 07-20-2018
' added support for CL files using MULTAX Off
' added functions to determine R plus or R minus
'
' *****

Public DebugOn As Boolean = False
Public Yes As Boolean = True
Public No As Boolean = False
Public X, XS, XM, XC, XE, Y, YS, YM, YC, YE, Z, ZS, ZM, ZC, ZE, ZD, I, J, K, R, F, Q, Q1, Q2
  As String
Public XCM, YCM As String
Public oldX, oldY, oldZ, oldZD, oldI, oldJ, oldK, oldR, oldF, oldQ, oldQ1, oldQ2 As String
Public S, T, G, DC As Integer
Public oldG, oldS, oldT As Integer
Public RLn As Integer = 0
Public WLn As Integer = 0
Public CC As Integer = 0
Public Currentline As Integer = 1
Public CurrentString As String = ""
Public LineBuffer(8) As String
Public nDigits As Integer = 5
Public Svec, Mvec, Evec, Dvec, CHH, CHX, CHY As Double
Public XSv, YSv, XMv, YMv, XEv, YEv, XCv, YCv As Double
Public Ksign, CHSign As Integer
Public JJ As Integer
Public KK As Integer
Public BP As Integer = 1
Public BMaxP As Integer = 8
Public Direction As Integer
Public Gcode As String
Public WriteCode As Integer
Public inFileName As String = "ncvout.cl"
Public outFileName As String = "Gcode.txt"
Public inPath As String = "C:\Program Files\procad\Verify\"
Public outPath As String = "C:\Users\George\Documents\Procad-CL\"
Public fileReader As System.IO.StreamReader
Public stringReader As String
Public stringWriter As String
Public Cycle As Integer = 0
Public Deep As Integer = 0
Public CycleOn As Integer = 0
Public CycleOff As Integer = 0
Public LookAhead As Integer = 0
Public FirstTool As Integer = 0
Public FirstSpindle As Integer = 0
```

```

Public FirstArc As Integer = 0
Public StartOfProgram As Integer = 0
Public LastOpp As Integer = 0
Public MultiAx As Boolean = False
'modal groups
Public GMove01 As Integer = 0
Public oldGMove01 As Integer = -1
Public GPlane02 As Integer = 17
Public oldGPlane02 As Integer = -1
Public GAbsInc03 As Integer = 90
Public oldGAbsInc03 As Integer = -1
Public GUnits06 As Integer = 20
Public oldGUnits06 As Integer = -1
Public GCRC07 As Integer = 40
Public oldGCRC07 As Integer = -1
Public GTLC08 As Integer = 49
Public oldGTLC08 As Integer = -1
Public GCycle09 As Integer = 80
Public oldGCycle09 As Integer = -1
Public GLevel10 As Integer = 99
Public oldGLevel10 As Integer = -1
Public GWork14 As Integer = 54
Public oldGWork14 As Integer = -1
Public GM(15) As Integer
Public oldGM(15) As Integer
Public fileWriter As System.IO.StreamWriter
Public saveFileDialog1 As New SaveFileDialog()
Public CLLine As String
Public TempGC As String
Public DArea As Double

Public Sub bPost_Click(sender As System.Object, e As System.EventArgs) Handles bPost.Click
    Dim JJ As Integer
    bPost.Enabled = False
    bPost.Text = "wait"
    'Dim fileReader As System.IO.StreamReader
    'Dim fileWriter As System.IO.StreamWriter
    'Dim saveFileDialog1 As New SaveFileDialog()

    Try
        saveFileDialog1.Filter = "NC File|*.nc|Text File|*.txt|CNC File|*.cnc"
        saveFileDialog1.Title = "Save Gcode File"
        saveFileDialog1.AddExtension = True
        saveFileDialog1.DefaultExt = ".nc"
        saveFileDialog1.InitialDirectory = outPath
        saveFileDialog1.ShowDialog()
        If saveFileDialog1.FileName <> "" Then
            outPath = GetPathPart(saveFileDialog1.FileName)
            outFileName = GetFileNamePart(saveFileDialog1.FileName)
        End If
    End Try

```

```

System.IO.File.Delete(outPath & outFileName)
fileReader = My.Computer.FileSystem.OpenTextFileReader(inPath & inFileName)
fileWriter = My.Computer.FileSystem.OpenTextFileWriter(outPath & outFileName, True)
Dim stringReader As String
Dim stringWriter As String
'Read in an 8 line buffer
'if there arent 8 lines BmaxP will be less than 8
Try
    For JJ = 1 To 8
        stringReader = fileReader.ReadLine()
        LineBuffer(JJ) = stringReader
        BMaxP = JJ
    Next

Catch ex As Exception
    MsgBox("Create 8 Line Buffer, something went wrong in Buffer Input " & vbCrLf &
        ex.Message)
End Try
RLn = RLn + 7
While (fileReader.EndOfStream = False)
    stringReader = fileReader.ReadLine()
    'Shift the Buffer down one line
    For JJ = 0 To 7
        LineBuffer(JJ) = LineBuffer(JJ + 1)
    Next
    'set the last buffer to the current line read
    LineBuffer(8) = stringReader
    'process the 1st buffered line
    WriteCode = ParseLine(LineBuffer(0), 4)
    RLn = RLn + 1
    stringWriter = LineBuffer(0)
    If WriteCode = 1 Then
        If Gcode <> "" Then
            fileWriter.WriteLine(Gcode)
            WLn = WLn + 1
        End If
    End If
End While
While LineBuffer(0) <> ""
    'no more lines to read, so process just from the buffer
    For JJ = 0 To 7
        LineBuffer(JJ) = LineBuffer(JJ + 1)
    Next
    LineBuffer(8) = ""
    BMaxP = BMaxP - 1
    WriteCode = ParseLine(LineBuffer(0), 4)
    stringWriter = LineBuffer(0)
    If WriteCode = 1 Then
        If Gcode <> "" Then
            fileWriter.WriteLine(Gcode)
            WLn = WLn + 1
        End If
    End If
End While

```

```

        End If
    End While

    TempGC = ""
    If oldZ <> Z Then
        TempGC = TempGC & " Z" & Z & " M09"
        oldZ = Z
    Else
        TempGC = TempGC & "M09"
    End If
    fileWriter.WriteLine(TempGC)
    fileWriter.WriteLine("G91 G28 Z0. M05")
    fileWriter.WriteLine("G28 Y0.")
    fileWriter.WriteLine("M30")
    fileWriter.WriteLine("(CREATED WITH CL2GCODE BY GSS)")
    fileWriter.WriteLine("( " & Chr(169) & " GSS 2018 " & "Version " &
Application.ProductVersion & " )")
    WLn = WLn + 4
    fileReader.Close()
    fileWriter.Close()
    bQuit.Enabled = True
    bPost.Text = "Done"
    MsgBox(outPath & outFileNames & " Lines read = " & RLn & " Lines write = " & WLn)
Catch ex As Exception
    MsgBox("I ran into an error " & vbCrLf & ex.Message)
End Try
End Sub

Public Function AdvanceBuffer(ByVal Badv As Integer) As Integer
    Dim tLine As String
    Dim CMD As String
    Dim temp As String
    Dim TLS As Integer
    Dim BLLine As String
    Dim KK As Integer
    Dim JJ As Integer
    Try
        'advance the buffer
        For KK = 1 To Badv
            stringReader = fileReader.ReadLine()
            'Shift the Buffer down one line
            For JJ = 0 To 7
                LineBuffer(JJ) = LineBuffer(JJ + 1)
            Next
            'set the last buffer to the current line read
            LineBuffer(8) = stringReader
            RLn = RLn + 1
            '***** New
            BLLine = LineBuffer(0)
            TLS = InStr(BLLine, " ")
            tLine = Microsoft.VisualBasic.Right(BLLine, Len(BLLine) - TLS)
            CMD = UCase(Trim(Microsoft.VisualBasic.Left(tLine, 6)))
            Select Case CMD
                Case "MULTAX"

```

```

        temp = Trim(UCase(Mid(tLine, 8, 4)))
        If temp = "ON" Then
            MultiAx = True
        Else
            MultiAx = False
        End If

    End Select

Next

Catch ex As Exception
    MsgBox("I ran into an error while advancing the buffer " & vbCrLf & ex.Message)
End Try
AdvanceBuffer = 1
End Function
Private Function ParseLine(CLLine As String, nDigits As Integer)
    Dim tLine As String
    Dim CMD As String
    Dim temp As String
    Dim FC As Integer
    Dim TLS As Integer
    Dim JJ As Integer
    'RLn = RLn + 1
    ParseLine = 0
    TLS = InStr(CLLine, " ")
    tLine = Microsoft.VisualBasic.Right(CLLine, Len(CLLine) - TLS)
    CMD = UCase(Trim(Microsoft.VisualBasic.Left(tLine, 6)))
    Select Case CMD
        Case "MULTAX"
            temp = Trim(UCase(Mid(tLine, 8, 4)))
            If temp = "ON" Then
                MultiAx = True
            Else
                MultiAx = False
            End If

        Case "LOADTL"
            T = Val(Mid(tLine, 8, Len(tLine) - 7))
            ParseLine = 1
            If StartOfProgram = 0 Then
                Gcode = MakeGcode()
            End If

            If FirstTool = 0 Then
                'Gcode for 1st tool
                Gcode = "M06 T" & T & "(TOOL CHANGE " & "T" & T & ")"
                fileWriter.WriteLine(Gcode)
                WLn = WLn + 1

            Else
                'G code for Next tool

```

```

Gcode = "M09"
fileWriter.WriteLine(Gcode)
WLn = WLn + 1
Gcode = "G00 G91 G28 Z0. M19"
fileWriter.WriteLine(Gcode)
WLn = WLn + 1
Gcode = "M01"
fileWriter.WriteLine(Gcode)
WLn = WLn + 1
Gcode = "M06 T" & T & "(TOOL CHANGE " & "T" & T & ")"
fileWriter.WriteLine(Gcode)
WLn = WLn + 1

End If
FirstTool = 1
Gcode = ""
oldF = "0."

```

```

Case "SPINDL"
    temp = Trim(UCase(Mid(tLine, 8, 4)))
    If temp = "CLW" Then
        Direction = 3
    Else
        Direction = 4
    End If
    TLS = InStr(tLine, ",")
    S = Val(UCase(Mid(tLine, TLS + 1, Len(tLine) - (TLS - 1))))
    ParseLine = 1
    If FirstSpindle = 0 Then

        CC = InStr(LineBuffer(3), "CYCLE")
        If CC = 0 Then
            AdvanceBuffer(4)
            CLLine = LineBuffer(0)
        Else
            CLLine = LineBuffer(2)
        End If
        'AdvanceBuffer(4)
        'CLLine = LineBuffer(0)
    Else
        CC = InStr(LineBuffer(3), "CYCLE")
        If CC = 0 Then
            If MultiAx = True Then
                AdvanceBuffer(3)
            Else
                AdvanceBuffer(4)
            End If
            CLLine = LineBuffer(0)
        Else
            CLLine = LineBuffer(2)
        End If
    End If

```

```

End If
End If

'CLLine = LineBuffer(0)
TLS = InStr(CLLine, " ")
tLine = Microsoft.VisualBasic.Right(CLLine, Len(CLLine) - TLS)
'Parse X Y Z from Goto
'trim off the un-need left most 8 characters
temp = Microsoft.VisualBasic.Right(tLine, Len(tLine) - 8)
'Find X
'Find where the first "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
X = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find Y
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
Y = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find Z
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
If MultiAx = True Then
    Z = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
Else
    Z = Round(Val(temp), nDigits)

End If

'Create gcode based on 1st tool or not 1st tool

If FirstSpindle = 0 Then
    'G cod for 1st Spindle start of program
    Gcode = "G00 G90 G54" & " X" & X & " Y" & Y & " S" & S & " M" & Direction
    oldF = "0."
    fileWriter.WriteLine(Gcode)
    WLn = WLn + 1
    Gcode = "G43 H" & T & " Z" & Z & " M08"
    fileWriter.WriteLine(Gcode)
    WLn = WLn + 1
    'Gcode = "Z" & Z
    'fileWriter.WriteLine(Gcode)
    'WLn = WLn + 1
Else

```

```

'Gcode for next spindle
Gcode = "G00 G90 G54" & " X" & X & " Y" & Y & " S" & S & " M" & Direction
oldF = "0."
fileWriter.WriteLine(Gcode)
WLn = WLn + 1
Gcode = "G43 H" & T & " Z" & Z & " M08"
fileWriter.WriteLine(Gcode)
WLn = WLn + 1
End If
oldF = "0."
FirstSpindle = 1
Gcode = ""

Case "RAPID"
    G = 0
    ParseLine = 0
    FirstArc = 0

Case "FEDRAT"
    G = 1
    F = Round(Val(UCase(Mid(tLine, 13, Len(tLine) - 12))), nDigits)
    ParseLine = 0
    FirstArc = 0

Case "GOTO"

    If MultiAx = True Then
        ParseLine = 0
        'look ahead 5 lines. If it is Cycle Off, don't output rapid.
        LookAhead = InStr(LineBuffer(5), "CYCLE / OFF")
        If LookAhead = 0 Then
            If ParseGoto(CLLine, nDigits) = 1 Then
                ParseLine = 1
                If Cycle = 0 Then
                    'not a cycle output G0 or G1
                    Gcode = MakeGcode()
                Else
                    'output the cycle Gcode instead
                    If Deep = 0 Then 'normal Cycle
                        Gcode = MakeGcode()
                    Else ' Deep Drill Cycle
                        Gcode = MakeGcode()
                    End If
                End If
            End If
        End If ' end look ahead
    End If 'MultiAx = true

    If MultiAx = False Then
        ParseLine = 0
    
```



```

'look ahead 5 lines. If it is Cycle Off, don't output rapid.
LookAhead = InStr(LineBuffer(5), "CYCLE / OFF")
If LookAhead = 0 Then
    If ParseGoto(CLLine, nDigits) = 1 Then
        ParseLine = 1
        If Cycle = 0 Then
            'not a cycle output G0 or G1
            Gcode = MakeGcode()
        Else
            'output the cycle Gcode instead
            If Deep = 0 Then 'normal Cycle
                Gcode = MakeGcode()
            Else ' Deep Drill Cycle
                Gcode = MakeGcode()
            End If
        End If
    End If
End If ' end look ahead
End If 'MultiAx = False

```

Case "CIRCLE"

```

'trime off the un-need left most 8 characters
temp = Microsoft.VisualBasic.Right(tLine, Len(tLine) - 8)
'Find X
'Find where the first "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
XC = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find Y
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
YC = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find Z
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
ZC = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find I
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","

```

```

I = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find J
'Find where the ", " is
FC = InStr(temp, ",")
'Get the Value Of X up to the ", "
J = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find K
'Find where the ", " is
FC = InStr(temp, ",")
'Get the Value Of X up to the ", "
K = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find R
'Find where the ", " is
FC = InStr(temp, ",")
'Get the Value Of X up to the ", "
'K = Val(Microsoft.VisualBasic.Left(temp, FC - 1))
R = Round(Val(temp), nDigits)

'*****
'Need to re-write for Buffer use
'get next GOTO for End of Circle
For JJ = 0 To 7
    LineBuffer(JJ) = LineBuffer(JJ + 1)
Next
'Read in One more line to the end of the buffer
'Can the Get a Line be made into a Function or Sub???
Try
    LineBuffer(8) = fileReader.ReadLine()
Catch ex As Exception
    MsgBox("I ran into an error" & vbCrLf & ex.Message)
End Try
CLLine = LineBuffer(0)
RLn = RLn + 1
'*****
TLS = InStr(CLLine, " ")
tLine = Microsoft.VisualBasic.Right(CLLine, Len(CLLine) - TLS)
CMD = UCase(Trim(Microsoft.VisualBasic.Left(tLine, 6)))
'trime off the un-need left most 8 characters
temp = Microsoft.VisualBasic.Right(tLine, Len(tLine) - 8)
'Find X
'Find where the first ", " is
FC = InStr(temp, ",")
'Get the Value Of X up to the ", "
X = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)

```

```

'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find Y
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
Y = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find Z
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
'Z = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
Z = Round(Val(temp), nDigits)
'Trim off the Left uo to the ","
'temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)
'*****
R = GetRDirection(oldX, oldY, X, Y, XC, YC, R)

ParseLine = 1
If K = 1 Then G = 3
If K = -1 Then G = 2
Gcode = MakeGcode()
FirstArc = 1
'Gcode = "G" & G & " X" & X & " Y" & Y & " Z" & Z & " R" & R
'If oldF <> F Then
'    Gcode = Gcode & " F" & F
'    oldF = F
'End If

Case "CALL"
'trime off the un-need left most 8 characters

' get start points of Arc
AdvanceBuffer(2)
tLine = LineBuffer(0)
temp = Microsoft.VisualBasic.Right(tLine, Len(tLine) - 20)
'Find X START
'Find where the first "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
XS = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - (FC + 9))

'Find Y START
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
YS = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)

```

```

'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - (FC + 9))

'Find Z START
'Find where the ", " is
FC = InStr(temp, ",")
'Get the Value Of X up to the ", "
ZS = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Get Mid points of arc
AdvanceBuffer(1)
tLine = LineBuffer(0)
temp = Microsoft.VisualBasic.Right(tLine, Len(tLine) - 20)
'Find X Mid point
'Find where the first ", " is
FC = InStr(temp, ",")
'Get the Value Of X up to the ", "
XM = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - (FC + 9))

'Find Y MID
'Find where the ", " is
FC = InStr(temp, ",")
'Get the Value Of X up to the ", "
YM = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - (FC + 9))

'Find Z MID
'Find where the ", " is
FC = InStr(temp, ",")
'Get the Value Of X up to the ", "
ZM = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'get End points of arc
AdvanceBuffer(1)
tLine = LineBuffer(0)
temp = Microsoft.VisualBasic.Right(tLine, Len(tLine) - 20)
'Find X END
'Find where the first ", " is
FC = InStr(temp, ",")
'Get the Value Of X up to the ", "
XE = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - (FC + 9))

```

```

'Find Y END
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
YE = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - (FC + 9))

'Find Z END
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
ZE = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Get Center point of Arc
AdvanceBuffer(1)
tLine = LineBuffer(0)
temp = Microsoft.VisualBasic.Right(tLine, Len(tLine) - 20)

'Find X Center
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
XC = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - (FC + 9))

'Find Y Center
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
YC = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - (FC + 9))

'Find Z Center
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
ZC = Round(Val(temp), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Need to calculate R
'R = length from start to center
R = Math.Sqrt(((XC - XS) * (XC - XS)) + ((YC - YS) * (YC - YS)))
R = Round(R, nDigits)
'SET X,Y,Z to the end points
X = XE
Y = YE

```

```

Z = ZE
'FindForm the Length of the Chord from Start to End
DC = Math.Sqrt(((XE - XS) * (XE - XS)) + ((YE - YS) * (YE - YS)))
K = (((XS - XC) * (YE - YC)) - ((YS - YC) * (XE - XC)))
If K > 0 Then K = 1 Else K = -1
K = GetDirection(XS, YS, XM, YM, XE, YE, XC, YC)
'IF THE CHORD IS

ParseLine = 1
If K = 1 Then G = 3
If K = -1 Then G = 2
Gcode = MakeGcode()
If DebugOn = True Then DoDebug1()
FirstArc = 1
'Gcode = "G" & G & " X" & X & " Y" & Y & " Z" & Z & " R" & R
'If oldF <> F Then
'    Gcode = Gcode & " F" & F
'    oldF = F
'End If

Case Else
    ParseLine = 0

End Select
'to debug a long cl file where you find an error
'set the Write Line Value to the line before the error happens
'un comment these lines, and set a break point
If WLn > 7 Then
    WLn = WLn
End If
End Function

Public Function ParseGoto(ByVal CLLine As String, ByVal nDigits As Integer)
    Dim tLine As String
    Dim CMD As String
    Dim temp As String
    Dim FC As Integer
    Dim TLS As Integer
    Cycle = 0
    Deep = 0
    CycleOff = 0
    ParseGoto = 0
    'Parse the the command
    TLS = InStr(CLLine, " ")
    tLine = Microsoft.VisualBasic.Right(CLLine, Len(CLLine) - TLS)
    CMD = UCase(Trim(Microsoft.VisualBasic.Left(tLine, 6)))
    Select Case CMD

        Case "GOTO"
            If G <> 0 Then G = 1
            'trime off the un-need left most 8 characters
            temp = Microsoft.VisualBasic.Right(tLine, Len(tLine) - 8)
            'Find X

```

```

'Find where the first "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
X = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find Y
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
Y = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

'Find Z
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of X up to the ","
Z = Round(Val(temp), nDigits)

'Lookup(Of Ahead to Next line. If Cycle, don't process
LookAhead = InStr(LineBuffer(1), "CYCLE")
'is this a deep cycle 0=no
Deep = InStr(LineBuffer(1), "DEEP")
CycleOff = InStr(LineBuffer(1), "OFF")
'if not cycle off ok to output
If CycleOff = 0 Then
    If LookAhead = 0 Then
        ParseGoto = 1
    Else
        CycleOn = 1
        FirstArc = 0
        'determine which cycle and parse R, Q, F, Z depth
        ' set cycle to a 1 so the line output will be a cycle.
        TLS = InStr(LineBuffer(1), " ")
        tLine = Microsoft.VisualBasic.Right(LineBuffer(1), Len(LineBuffer(1)) -
        TLS)
        'trime off the un-need left most 8 characters
        temp = Microsoft.VisualBasic.Right(tLine, Len(tLine) - 8)

        'this is the Cycle but skip it for now
        'Find the first ,
        FC = InStr(temp, ",")
        'Trim off the Left uo to the ","
        temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)

    If Deep = 0 Then
        'this is for Drill and Face, Tap, Bore, Ream

```

```

'find the next , and get Z depth
FC = InStr(temp, ",")
'Get the Value Of ZD up to the ", "
ZD = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)
'trim off up to the next , Skip the IPM IPR
FC = InStr(temp, ",")
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)
'Find F
'Find where the ", " is
FC = InStr(temp, ",")
'Get the Value Of F up to the ", "
F = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)
'Find R
'Find where the ", " is
FC = InStr(temp, ",")
'Get the Value Of F up to the ", "
R = Round(Val(temp), nDigits)

Cycle = 1
'determine which G code Cycle it is
G = 81
If InStr(LineBuffer(1), "DRILL") <> 0 Then G = 81
If InStr(LineBuffer(1), "FACE") <> 0 Then G = 82
'Tap G code is Based on Spindle Direction
If InStr(LineBuffer(1), "TAP") <> 0 Then
    If Direction = 3 Then
        G = 84
    Else
        G = 74
    End If
End If
If InStr(LineBuffer(1), "BORE") <> 0 Then G = 85
If InStr(LineBuffer(1), "REAM") <> 0 Then G = 85

ParseGoto = 1

Else

'this is for DEEP
'find the next , and get Z depth
FC = InStr(temp, ",")
'Get the Value Of ZD up to the ", "
ZD = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left uo to the ", "
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)
'trim off up to the next , Skip the INCR
FC = InStr(temp, ",")
'Trim off the Left uo to the ", "

```



```

temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)
'Find Q1
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of Q1 up to the ","
Q1 = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left up to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)
'Find Q2
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of Q2 up to the ","
Q2 = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left up to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)
'trim off up to the next , Skip the IPM IPR
FC = InStr(temp, ",")
'Trim off the Left up to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)
'Find F
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of F up to the ","
F = Round(Val(Microsoft.VisualBasic.Left(temp, FC - 1)), nDigits)
'Trim off the Left up to the ","
temp = Microsoft.VisualBasic.Right(temp, Len(temp) - FC)
'Find R
'Find where the "," is
FC = InStr(temp, ",")
'Get the Value Of F up to the ","
R = Round(Val(temp), nDigits)

Cycle = 1
G = 83
ParseGoto = 1
End If 'end of type of cycle

```

```

End If 'end of look ahead cycle
End If 'end of cycle off test
'End Goto

```

```
End Select
```

```
End Function
```

```

Public Function MakeGcode() As String
Dim TempGC As String = ""
TempGC = ""
If StartOfProgram = 0 Then
TempGC = "O1234"
fileWriter.WriteLine(TempGC)
WLn = WLn + 1

TempGC = "G00 G17 G20 G40 G49 G80 G90 G94 G98"

```

```

        fileWriter.WriteLine(TempGC)
        WLn = WLn + 1

        TempGC = "G52 X0. Y0."
        fileWriter.WriteLine(TempGC)
        WLn = WLn + 1

        TempGC = "G00 G28 G91 Z0."
        fileWriter.WriteLine(TempGC)
        WLn = WLn + 1

        TempGC = ""
        StartOfProgram = 1
    End If

    If G = 2 Or G = 3 Then
        'Gcode = "G" & G & " X" & X & " Y" & Y & " Z" & Z & " R" & R
        If oldG <> 0 Then
            TempGC = TempGC & " G" & G
            oldG = G
        End If
        If oldX <> X Then
            TempGC = TempGC & " X" & X
            oldX = X
        End If
        If oldY <> Y Then
            TempGC = TempGC & " Y" & Y
            oldY = Y
        End If
        If oldZ <> Z Then
            TempGC = TempGC & " Z" & Z
            oldZ = Z
        End If
        'If FirstArc = 0 Then
        '    TempGC = TempGC & " R" & R
        '    oldR = R
        '    'TempGC = TempGC & " F" & F
        '    'oldF = F
        'End If
        If R = R Then
            TempGC = TempGC & " R" & R
            oldR = R
        End If
        If oldF <> F Then
            TempGC = TempGC & " F" & F
            oldF = F
        End If
    End If

    If Cycle = 0 Then
        'not a cycle output G0 or G1
        'Gcode = "G" & G & " X" & X & " Y" & Y & " Z" & Z
    End If

```

```

If oldG <> G Then
    TempGC = TempGC & " G" & G
    oldG = G
End If
If oldX <> X Then
    TempGC = TempGC & " X" & X
    oldX = X
End If
If oldY <> Y Then
    TempGC = TempGC & " Y" & Y
    oldY = Y
End If
If oldZ <> Z Then
    TempGC = TempGC & " Z" & Z
    oldZ = Z
End If
If G <> 0 Then
    If oldF <> F Then
        TempGC = TempGC & " F" & F
        oldF = F
    End If
End If

```

```

If CycleOn = 1 Then
    'add cycle end to Gcode
    TempGC = "G80" & TempGC
    CycleOn = 0
    If InStr(TempGC, "Z") = 0 Then
        TempGC = TempGC & " Z" & Z
        oldZ = Z
    End If
End If

```

Else

```

'output the cycle Gcode instead
If Deep = 0 Then 'normal Cycle
    'Gcode = "G" & G & " G99 X" & X & " Y" & Y & " Z-" & ZD & " R" & R
    If oldG <> G Then
        TempGC = TempGC & " G" & G & " G99"
        oldG = G
    End If
    If oldX <> X Then
        TempGC = TempGC & " X" & X
        oldX = X
    End If
    If oldY <> Y Then
        TempGC = TempGC & " Y" & Y
        oldY = Y
    End If
    If oldZD <> ZD Then
        TempGC = TempGC & " Z-" & ZD
        oldZD = ZD
    End If

```

```

    If oldR <> R Then
        TempGC = TempGC & " R" & R
        oldR = R
    End If

Else ' Deep Drill Cycle
    'Gcode = "G" & G & " G99 X" & X & " Y" & Y & " Z-" & ZD & " Q1 " & Q1 & " Q2 " &
    Q2 & " R" & R
    'Gcode = "G" & G & " G99 X" & X & " Y" & Y & " Z-" & ZD & " Q" & Q2 & " R" & R
    If oldG <> G Then
        TempGC = TempGC & " G" & G & " G99"
        oldG = G
    End If
    If oldX <> X Then
        TempGC = TempGC & " X" & X
        oldX = X
    End If
    If oldY <> Y Then
        TempGC = TempGC & " Y" & Y
        oldY = Y
    End If
    If oldZD <> ZD Then
        TempGC = TempGC & " Z-" & ZD
        oldZD = ZD
    End If
    If oldQ2 <> Q2 Then
        TempGC = TempGC & " Q" & Q2
        oldQ2 = Q2
    End If
    If oldR <> R Then
        TempGC = TempGC & " R" & R
        oldR = R
    End If

End If
If G <> 0 Then
    If oldF <> F Then
        TempGC = TempGC & " F" & F
        oldF = F
    End If
End If
End If

MakeGcode = LTrim(TempGC)
End Function

Public Function Round(nn As Double, nDigits As Integer) As String
    Round = Int(nn * 10 ^ nDigits + 0.5) / (10 ^ nDigits)
    'edit 3-9-2018 - no need to test value. just check for missing decimal point
    'If ((Val(Round) >= 0) Or (Val(Round) < 0)) Then
    If InStr(Round, ".") = 0 Then
        Round = Round & "."
    End If

```

```

    'End If
End Function

Private Sub bQuit_Click(sender As System.Object, e As System.EventArgs) Handles bQuit.Click
    Me.Close()
End Sub

Private Sub Button1_Click(sender As System.Object, e As System.EventArgs) Handles
Button1.Click

    Dim saveFileDialog1 As New SaveFileDialog()
    saveFileDialog1.Filter = "NC File|*.nc|Text File|*.txt|CNC File|*.cnc"
    saveFileDialog1.Title = "Save Gcode File"
    saveFileDialog1.AddExtension = True
    saveFileDialog1.DefaultExt = ".nc"
    saveFileDialog1.InitialDirectory = outputPath
    saveFileDialog1.ShowDialog()

    ' If the file name is not an empty string open it for saving.
    If saveFileDialog1.FileName <> "" Then
        outputPath = GetPathPart(saveFileDialog1.FileName)
        outFileName = GetFileNamePart(saveFileDialog1.FileName)
    End If

    'Dim BrowseFolder As New FolderBrowserDialog
    'BrowseFolder.SelectedPath = outputPath
    'BrowseFolder.ShowDialog()
    'outputPath = BrowseFolder.SelectedPath
    '
    'Dim message, title, defaultValue As String
    ' Dim myValue As Object
    ' Set prompt.
    'message = "Enter a New File Name"
    ' Set title.
    'title = "CL2Gcode File Name"
    'defaultValue = outFileName ' Set default value.

    ' Display message, title, and default value.
    'myValue = InputBox(message, title, defaultValue)
    ' If user has clicked Cancel, set myValue to defaultValue
    'If myValue <> "" Then outFileName = myValue

End Sub

Private Function GetFileNamePart(ByRef aFilename As String)
    Dim LF As Integer
    Dim FTemp As String
    Dim LSlash As Integer
    FTemp = aFilename
    LF = Len(FTemp)
    LSlash = InStrRev(FTemp, "\")
    If LSlash <> 0 Then

```

```

        FTemp = Mid(FTemp, LSlash + 1, LF - LSlash)
    End If
    Return FTemp
End Function

Private Function GetPathPart(ByRef aFilename As String)
    Dim LF As Integer
    Dim FTemp As String
    Dim LSlash As Integer
    FTemp = aFilename
    LF = Len(FTemp)
    LSlash = InStrRev(FTemp, "\")
    If LSlash <> 0 Then
        FTemp = Mid(FTemp, 1, LSlash)
    End If
    Return FTemp
End Function

Public Function GetSign(ByVal Number As Integer) As Integer
    GetSign = 1
    If Number >= 0 Then GetSign = 1 Else GetSign = -1
End Function

Public Function GetChordHeight(ByVal XSv, YSv, XMv, YMv, XEv, YEv, XCv, YCv, R) As Double
    'get the Sagitta
    'Dim Svec, Mvec, Evec, Dvec, Ksign, CHH, CHSign
    'Dim XSv, YSv, XMv, YMv, XEv, YEv, XCv, YCv As Double
    'Dim CHX, CHY, CHH As Double
    GetChordHeight = 0
    'find X mid point of chord
    CHX = (XSv + XEv) / 2
    'find Y mid point of chord
    CHY = (YSv + YEv) / 2
    'get the Sagitta
    CHH = Math.Sqrt((((CHX - XMv) * (CHX - XMv)) + ((CHY - YMv) * (CHY - YMv))))
    CHH = Round(CHH, 6)
    'invert CHH if it is Greater than the Radius
    If CHH > R Then CHH = CHH * -1
    GetChordHeight = CHH
End Function

Public Function GetDirection(ByRef XS As String, YS As String, XM As String, YM As String,
XE As String, YE As String, XC As String, YC As String) As Integer
    ' Get the Swing Direction of the Arc from Start Point, to Mid Point on Arc, to the End
    Point, Back to the Start Point
    ' Area of the triangles will be clockwise if < 0
    ' Area of triangles will be CounterClockwise if > 0
    ' If Area = 0 then it is Not an Arc
    '-1 = CW
    '1 = CCW
    GetDirection = 1
    'get center values
    XCv = Val(XC)
    YCv = Val(YC)

```

```

'normalize the points based on the center
'subtract the center from each point
XSv = Val(XS) - XCv
YSv = Val(YS) - YCv
XMv = Val(XM) - XCv
YMv = Val(YM) - YCv
XEv = Val(XE) - XCv
YEv = Val(YE) - YCv
' set center points to Zero
XCv = 0
YCv = 0
'Now all the points are normalized around Xc,Yc = 0,0

'
' I use the Sagitta to determine if it's an arc > 180 angle
'get the Sagitta
CHH = GetChordHeight(XSv, YSv, XMv, YMv, XEv, YEv, XCv, YCv, R)
CHSign = GetSign(CHH)
'-1 = CW
'1 = CCW
'' GetDirection = Ksign * CHSign

' Now Use the Surveyors Area method.
' I only need to know positive or negative area
'
' need at least three vertices of the area.
' the Observer or Surveyor is at the last point
' the Observer see's the vertices in order and enters them in this order
' if they are CW order the area is Negative
' if they are CCW order the Area is Positive
' With Three vertices we list them in Order, with the Observer
' on the Last vertex.
' P1x, P1y P2x, P2y, Ox, Oy
' shoelace method (((P1x*P2y) + (P2x*Oy) + (Ox*P1y)) - ((P1y*P2x) + (P2y*Ox) + (Oy*P1x)
)) / 2
' Pass the vertices in the order Start Point, Mid Point, End point - Observer location
DArea = Area(XSv, YSv, XMv, YMv, XEv, YEv)
'1= CCW -1= CW
'If K = 1 Then G = 3
'If K = -1 Then G = 2
If DArea < 0 Then Ksign = -1 'K =1 CCW
If DArea > 0 Then Ksign = 1 'K = -1 CW
' find the Mid point of the Chord of the arc
' Start point to End point = Chord
I = GetMidPoint(XSv, YSv, XEv, YEv)
' Find the Sagitta ( length from Mid point of Chord to Mid point on Arc
' round it to 2 decimal places more than the radius
I = Round(GetSagitta(XCM, YCM, XMv, YMv), 6)
' If the Sagitta is > than the radius, the Swing of the Arc is > 180
' So R must be Negative
If Val(I) > Val(R) Then
    R = R * -1
End If

```

```

    GetDirection = Ksign
End Function

Public Function GetRDirection(ByRef XS As String, YS As String, XE As String, YE As String,
XC As String, YC As String, R As String) As String
    ' compare the Arc Swing Direction to the Triangle Direction of the SP, CP, EP, SP
    Direction
    ' Area of the triangles will be clockwise if < 0
    ' Area of triangles will be CounterClockwise if > 0
    ' If Area = 0 then it is Not an Arc
    '-1 = CW
    '1 = CCW
    GetRDirection = R
    'get center values
    XCv = Val(XC)
    YCv = Val(YC)
    'normalize the points based on the center
    'subtract the center from each point
    XSv = Val(XS) - XCv
    YSv = Val(YS) - YCv
    XMv = Val(XM) - XCv
    YMv = Val(YM) - YCv
    XEv = Val(XE) - XCv
    YEv = Val(YE) - YCv
    ' set center points to Zero
    XCv = 0
    YCv = 0
    '
    ' Now use the Surveyors Area method.
    '
    ' need at least three vertices of the area.
    ' the Observer or Surveyor is at the last point
    ' the Observer see's the vertices in order and enters them in this order
    ' if they are CW order the area is Negative
    ' if they are CCW order the Area is Positive
    ' With Three vertices we list them in Order, with the Observer
    ' on the Last vertex.
    ' Plx, Ply P2x, P2y, Ox, Oy
    ' shoelace method (((Plx*P2y) + (P2x*Oy) + (Ox*Ply)) - ((Ply*P2x) + (P2y*Ox) + (Oy*Plx)
    )) / 2
    ' Pass the vertices in the order Center Point, End point, Start Point - observer
    location )
    ' Standing at the Start point, look at the Center Point, then the End point, then back
    to the start point
    ' What Direction order do the points go in?
    DArea = Area(XCv, YCv, XEv, YEv, XSv, YSv)
    '1= CCW -1= CW
    'If K = 1 Then arc is G = 3 CCW
    'If K = -1 Then arc is G = 2 CW
    'if Swing of Arc is <=180 then R direction Opposite of Swing Direction and R is Plus
    'if Swing of Arc is > 180 then R direction is the Same as the Swing Direction and R is
    Negative

```



```

    Ksign = 0
    If DArea < 0 Then Ksign = -1 'K =1 CCW
    If DArea > 0 Then Ksign = 1 'K = -1 CW
    If Ksign = Val(K) Then GetRDirection = R * -1
    If Ksign <> Val(K) Then GetRDirection = R
End Function

Public Function Area(P1x, P1y, P2x, P2y, Ox, Oy) As Double
    ' Start Point, Mid point, End Point in correct order
    Area = 0
    ' shoelace method (((P1x*P2y) + (P2x*Oy) + (Ox*P1y)) - ((P1y*P2x) + (P2y*Ox) + (Oy*P1x)))
    / 2
    Area = (((P1x * P2y) + (P2x * Oy) + (Ox * P1y)) - ((P1y * P2x) + (P2y * Ox) + (Oy * P1x)
    )) / 2

End Function

Public Function GetSagitta(XCM, YCM, XM, YM)
    GetSagitta = Math.Sqrt((XM - XCM) ^ 2 + (YM - YCM) ^ 2)
End Function

Function GetMidPoint(ByRef XS, YS, XE, YE)
    XCM = (XS + XE) / 2
    YCM = (YS + YE) / 2
    GetMidPoint = 1
End Function

Public Sub DoDebug1()
    'XS, YS, XM, YM, XE, YE, XC, YC, R
    'Dvec = Svec - Mvec - Evec
    'Ksign = GetSign(Dvec) * -1
    ' AddText( "Text to Add", CR Yes or No )
    'Gcode
    AddText("Real Values", Yes)
    AddText("X Values S,M,E,C ", Yes)
    AddText(XS, Yes)
    AddText(XM, Yes)
    AddText(XE, Yes)
    AddText(XC, Yes)
    AddText("Y Values S,M,E,C ", Yes)
    AddText(YS, Yes)
    AddText(YM, Yes)
    AddText(YE, Yes)
    AddText(YC, Yes)
    AddText("", Yes)
    AddText("Normalized Values", Yes)
    AddText("X Values S,M,E,C ", Yes)
    AddText(XSv, Yes)
    AddText(XMv, Yes)
    AddText(XEv, Yes)
    AddText(XCv, Yes)
    AddText("Y Values S,M,E,C ", Yes)
    AddText(YSv, Yes)
    AddText(YMv, Yes)
    AddText(YEv, Yes)

```

```

AddText(YCv, Yes)
AddText("", Yes)
AddText("DArea, CHMx, CHMy ", Yes)
AddText(DArea & ", " & XCM & ", " & YCM, Yes)
AddText("Sagita, I, KSign, K-Direction ", Yes)
AddText(CHH & ", " & I & ", " & Ksign & ", " & K, Yes)
AddText(Gcode, Yes)
AddText("", Yes)
AddText("", Yes)
If Not Debug1.Visible Then Debug1.Visible = True
End Sub
Public Function AddText(TS As String, Optional CR As Boolean = True)
    If CR = True Then
        Debug1.tb_Debug1.Text = Debug1.tb_Debug1.Text & TS & vbCrLf
    Else
        Debug1.tb_Debug1.Text = Debug1.tb_Debug1.Text & TS
    End If
    AddText = 1
End Function
Public Function GetQuad(ByRef ArcPoint)
    'X+ Y+ = Q1
    'X- Y+ = Q2
    'X- Y- = Q3
    'X+ Y- = Q4
    GetQuad = 1
    If ArcPoint.x >= 0 And ArcPoint.Y >= 0 Then
        ArcPoint.Q = 1
        GetQuad = 1
    End If
    If ArcPoint.x < 0 And ArcPoint.Y >= 0 Then
        ArcPoint.Q = 2
        GetQuad = 2
    End If
    If ArcPoint.x < 0 And ArcPoint.Y < 0 Then
        ArcPoint.Q = 3
        GetQuad = 3
    End If
    If ArcPoint.x >= 0 And ArcPoint.Y < 0 Then
        ArcPoint.Q = 4
        GetQuad = 4
    End If
End Function

Private Sub CL2GcodeII_Load(sender As Object, e As System.EventArgs) Handles Me.Load
    Dim JJ As Integer
    Me.Label11.Text = Chr(169) & " GSS 2018 " & "Version " & Application.ProductVersion
    'set modal default values
    GM(0) = 52 'G04 G09 G10 G28 G52
    GM(1) = 0 'Motion Group 1 G00 G01
    GM(2) = 17 'Plane select G17 G18 G19
    GM(3) = 90 'AbsInc G90 G91
    GM(6) = 20 'Units G20 INCH G21 METRIC
    GM(7) = 40 'CRC Cutter Radius Comp G40 G41 G42

```

```
GM(8) = 49 'TLC Tool Length Comp G43 G49
GM(9) = 80 'Canned Cycles
GM(10) = 99 'Level G99=R G98=Initial
GM(14) = 54 'Work G54 G55 G56 G57 G58 G59
GM(15) = 64 'CUTTING MODE G64=Profiling G61=exact stop
For JJ = 0 To 15
    oldGM(JJ) = -1
Next

End Sub
End Class
```