# How to copy Microchip Assembly code into a Word document

Copying text from the Microchip Assembly editor into a Word document is not as easy as it sounds. You are probably reading this because your efforts to use the following standard methods have failed.

1. Pasting into a Notepad document loses the colour information.

2. Pasting into a Word document also loses the colour information. Even worse, the horizontal and vertical spacing will be corrupted.

3. Pasting a screenshot into a Word document, where it can be cropped and re-sized, will work but the quality will be poor. Using screenshots to transfer anything but a very short program can be tedious.

4. You can use a utility like CutePDF to print the entire Assembly code into a stand-alone PDF file. If that is all you need, then you're all set. Pasting a large PDF file into a word document does have its own drawbacks, however.

I took the bull by the horns and wrote a program in Visual Basic (2015) to do the transfer. A Microchip Assembly source file, with extension ".asm", is simply a text document. The source file does not contain any of the colour information; the colours are added by the Assembly editor.

My Visual Basic program has to go through the same process as the Assembly editor to identify and classify the various fields of text in the code. My programs has several vectors of strings, which are lists of the various fields. For example, vector SysRegNames() is a list of the names of all possible system registers, starting with register "TMR0" and ending with register "EECON2". Vector SysRegBitNames() is a list of all accessible bits in the system registers, starting with bit "irp" in the "STATUS" register and ending with bit "rd" in the "EECON1" register. Vector OpCodeNames() is a list of all possible opcodes. (My Microchip microprocessor of choice is the 16F882. If you use a different chip, then you may have to add to these lists whatever additional register names, bit names and opcodes that your chip uses.) There are also lists for the nouns and verbs which the Microchip Assembly compiler recognizes. (I do not use the compiler aggressively, so you may need to add to the compiler lists any nouns and verbs you use.)

There is one final list, which is populated when the Visual Basic programs starts running. This vector is UserLabelNames(). The program makes an initial pass through the Assembly code to identify all labels (i.e., text fields starting in column 1). Any labels which are not already contained in the system lists are assumed to be User-defined labels and are added to the UserLabelNames() list.

The program assigns colours to the different types of fields. They do not have to be the same as used by the default settings in the Assembly editor, but it may help if they are.

With all this information at hand, the Visual Basic program then runs through the Assembly code a second time. It can now classify every field it encounters and look up the appropriate colour for that type of field. Writing the coloured fields into a new Word document is a straight-forward use of the Office.Interop.Word functions.

The Visual Basic program is a Window Forms application, consisting of a main form and three modules: Variables.vb, ParseAndWrite.vb and Procedures.vb. These blocks are listed in the Appendices.

Jim Hawley
April 2022

(As always, an e-mail describing errors and omissions would be appreciated.)

# Appendix "A"
# Class Form1.vb

```
Option Strict On
Option Explicit On

Imports System
Imports System.Windows.Forms
Imports System.ComponentModel
Imports System.Threading
Imports System.IO.Ports
Imports Word = Microsoft.Office.Interop.Word

Public Class Form1
    Inherits System.Windows.Forms.Form

    Public Sub New()
        ' Set parameters of the screen and the display
        Name = "Main"
        Text = "Print Microchip Assembly code to Word document"
        FormBorderStyle = System.Windows.Forms.FormBorderStyle.Fixed3D
        Size = New Drawing.Size(
            My.Computer.Screen.Bounds.Width, My.Computer.Screen.Bounds.Height)
        CenterToScreen()
        MinimizeBox = True
        MaximizeBox = True
        Me.Refresh()
    End Sub

    Private Sub Main_Load() Handles Me.Load
        ' This subroutine runs automatically when the form is loaded
        Me.Refresh()
    End Sub

    '
********************************************************************************
    ' *** Controls
********************************************************************************

    Private WithEvents buttonExit As New System.Windows.Forms.Button With
        {.Size = New Drawing.Size(75, 25), .Location = New Drawing.Point(5, 5),
        .Text = "Exit", .TextAlign = ContentAlignment.MiddleCenter,
        .Visible = True, .Parent = Me}

    Private WithEvents buttonExecute As New System.Windows.Forms.Button With
        {.Size = New Drawing.Size(75, 25), .Location = New Drawing.Point(5, 35),
        .Text = "Execute", .TextAlign = ContentAlignment.MiddleCenter,
        .Visible = True, .Parent = Me}

    Public labelLineCounter As New System.Windows.Forms.Label With
        {.Size = New Drawing.Size(100, 25), .Location = New Drawing.Point(5, 65),
        .Text = "Line number ", .TextAlign = ContentAlignment.MiddleLeft,
        .Visible = False, .Parent = Me}

    Private textboxContents As New System.Windows.Forms.TextBox With
        {.Size = New Drawing.Size(500, 200), .Location = New Drawing.Point(5, 95),
        .Text = "", .TextAlign = HorizontalAlignment.Left, .Multiline = True,
```

```
        .ScrollBars = ScrollBars.Vertical, .Font = New Drawing.Font("Arial", 12),
        .BorderStyle = BorderStyle.FixedSingle, .Visible = False, .Parent = Me}

    '
'*****************************************************************************************
    ' *** Control handlers
'*********************************************************************

    Public Sub buttonExit_Click() Handles buttonExit.MouseClick
        Application.Exit()
    End Sub

    Public Sub buttonExecute_Click() Handles buttonExecute.MouseClick
        Procedures.DefineAllSystemRegisterNames()
        Procedures.DefineAllSystemRegisterBitNames()
        Procedures.DefineAllOpCodeNames()
        Procedures.DefineAllDestinationNames()
        Procedures.DefineAllCompilerOperatorNames()
        Procedures.DefineAllCompilerNounNames()
        Procedures.SearchForAllUserLabelNames()
        Procedures.DefineAllTabStops()
        Procedures.MainProgram()
    End Sub

End Class
```

# Appendix "B"
# Module Variables.vb

```
Option Strict On
Option Explicit On

Public Module Variables

    ' File names
    Public MainDir As String = "C:\Users\User\Desktop\New_stuff\Print_ASM_to_Word\"
    Public ASMFileName As String = "PICtoPICCommTest.asm"
    Public WordFileName As String = "WordOutputFile.doc"

    ' Page properties for the Word document
    Public PageWidthInch As Double = 8.5
    Public PageHeightInch As Double = 11
    Public LeftMarginInch As Double = 0.5
    Public RightMarginInch As Double = 0.5
    Public TopMarginInch As Double = 0.5
    Public BottomMarginInch As Double = 0.5
    Public IsOrientationPortrait As Boolean = True

    ' Paragraph properties for the Word document
    '   Left-justified
    '   Single spacing, with no additional spacing before or after
    '   No identation of the first line

    ' Default Font properties
    Public FontName As String = "Courier New"
    Public FontSize As Int32 = 10
    Public IsFontBold As Boolean = False
    Public IsFontItalic As Boolean = False
    Public Colour_Default As Microsoft.Office.Interop.Word.WdColor =
        Microsoft.Office.Interop.Word.WdColor.wdColorBlack

    ' Definition of colours
    Public Colour_SysRegName As Microsoft.Office.Interop.Word.WdColor =
        Microsoft.Office.Interop.Word.WdColor.wdColorLightBlue
    Public Colour_SysRegBitName As Microsoft.Office.Interop.Word.WdColor =
        Microsoft.Office.Interop.Word.WdColor.wdColorRed
    Public Colour_OpCodeName As Microsoft.Office.Interop.Word.WdColor =
        Microsoft.Office.Interop.Word.WdColor.wdColorDarkBlue
    Public Colour_DestName As Microsoft.Office.Interop.Word.WdColor =
        Microsoft.Office.Interop.Word.WdColor.wdColorRed
    Public Colour_CompOpName As Microsoft.Office.Interop.Word.WdColor =
        Microsoft.Office.Interop.Word.WdColor.wdColorDarkBlue
    Public Colour_CompNounName As Microsoft.Office.Interop.Word.WdColor =
        Microsoft.Office.Interop.Word.WdColor.wdColorOrange
    Public Colour_UserLabelName As Microsoft.Office.Interop.Word.WdColor =
        Microsoft.Office.Interop.Word.WdColor.wdColorPlum
    Public Colour_Comment As Microsoft.Office.Interop.Word.WdColor =
        Microsoft.Office.Interop.Word.WdColor.wdColorGreen
    Public Colour_Number As Microsoft.Office.Interop.Word.WdColor =
        Microsoft.Office.Interop.Word.WdColor.wdColorOrange
    Public Colour_Black As Microsoft.Office.Interop.Word.WdColor =
        Microsoft.Office.Interop.Word.WdColor.wdColorBlack
```

```
    ' Variables used to detect system register names
    Public NumSysRegNames As Int32
    Public SysRegNames(100) As String

    ' Variables used to detect system registers' bit names
    Public NumSysRegBitNames As Int32
    Public SysRegBitNames(1000) As String

    ' Variables used to detect Assembly OpCodes
    Public NumOpCodeNames As Int32
    Public OpCodeNames(100) As String

    ' Variables used to detect Assembly destinations
    Public NumDestNames As Int32
    Public DestNames(2) As String

    ' Variables used to detect Compiler Operators
    Public NumCompOpNames As Int32
    Public CompOpNames(100) As String

    ' Variables used to detect Compiler Nouns
    Public NumCompNounNames As Int32
    Public CompNounNames(100) As String

    ' Variables used to detect User labels
    Public NumUserLabelNames As Int32
    Public UserLabelNames(1000) As String

    ' Variables used to define Tab stops
    Public NumTabStops As Int32
    Public TabStops(100) As Int32

End Module
```

# Appendix "C"
# Module ParseAndWrite.vb

```
'Option Strict On
Option Explicit On

' List of subroutines:-
'   ParseAndWriteOneAssemblyLine()
'   ReplaceTabsWithBlanks()
'   WriteOneFieldToWordDocument()

Public Module ParseAndWrite

    Private FirstCharacter As String
    Private RemainStr As String
    Private LenRemainStr As Int32
    Private Loc As Int32
    Private Label As String
    Private Verb As String
    Private Noun As String
    Private NounContainsComma As Boolean
    Private Argument As String

    Public Sub ParseAndWriteOneAssemblyLine(ByVal lLineIn As String)
        ' If the input line is nil, then return
        If (lLineIn = "") Then
            WriteOneFieldToWordDocument(vbCrLf, Colour_Black)
            Exit Sub
        End If
        ' Use variable RemainStr when parsing; it gets chopped up.
        ' Replace all Tabs with blanks
        RemainStr = ReplaceTabsWithBlanks(lLineIn)
        ' Trim any trailing blanks from RemainStr
        RemainStr = RTrim(RemainStr)
        ' Test if first character of the input line is a semi-colon
        FirstCharacter = Strings.Left(RemainStr, 1)
        If (FirstCharacter = ";") Then
            ' The entire line is a comment
            WriteOneFieldToWordDocument(RemainStr, Colour_Comment)
            WriteOneFieldToWordDocument(vbCrLf, Colour_Black)
            Exit Sub
        End If
        ' If the input line starts with a Label, then process it
        If (FirstCharacter <> " ") Then
            ' Find the end of the Label
            Loc = InStr(RemainStr, " ")
            ' Extract the Label
            If (Loc > 0) Then
                Label = Strings.Left(RemainStr, Loc - 1)
                RemainStr = Strings.Right(RemainStr, Len(RemainStr) - Loc + 1)
            Else
                Label = RemainStr
                RemainStr = ""
            End If
            Do
                ' Test if the Label is a system register name
                For I As Int32 = 1 To NumSysRegNames Step 1
```

```
            If (Label = SysRegNames(I)) Then
                ' The Label is a system register name
                WriteOneFieldToWordDocument(Label, Colour_SysRegName)
                Exit Do
            End If
        Next I
        ' Test if the Label is a system register's bit name
        For I As Int32 = 1 To NumSysRegBitNames Step 1
            If (Label = SysRegBitNames(I)) Then
                ' The Label is a system register's bit name
                WriteOneFieldToWordDocument(Label, Colour_SysRegBitName)
                Exit Do
            End If
        Next I
        ' Test if the Label is an Assembly destination name
        For I As Int32 = 1 To NumDestNames Step 1
            If (Label = DestNames(I)) Then
                ' The Label is an Assembly destination name
                WriteOneFieldToWordDocument(Label, Colour_DestName)
                Exit Do
            End If
        Next I
        ' Test if the Label is a User label name
        For I As Int32 = 1 To NumUserLabelNames Step 1
            If (Label = UserLabelNames(I)) Then
                ' The Label is a User label name
                WriteOneFieldToWordDocument(Label, Colour_UserLabelName)
                Exit Do
            End If
        Next I
        ' The Label cannot be classified
        MsgBox(
            "In line:" & vbCrLf & lLineIn & vbCrLf &
            "Error:" & vbCrLf & "The Label is not recognized.")
        ' Close the Assembly file
        FileReader.Close()
        FileReader.Dispose()
        ' Close the Word application
        objWordDoc.Close()
        objWordApp.Quit()
        System.Runtime.InteropServices.Marshal.ReleaseComObject(objWordApp)
        Application.Exit()
        Exit Sub
    Loop
End If
' If there is nothing else on this line, then return
LenRemainStr = Len(RemainStr)
If (LenRemainStr = 0) Then
    WriteOneFieldToWordDocument(vbCrLf, Colour_Black)
    Exit Sub
End If
' Copy all blanks which come before the Verb
For I As Int32 = 1 To LenRemainStr Step 1
    If (Strings.Mid(RemainStr, I, 1) = " ") Then
        WriteOneFieldToWordDocument(" ", Colour_Black)
    Else
        Exit For
    End If
```

```
Next I
' Now remove the leading blanks from RemainStr
RemainStr = LTrim(RemainStr)
' If there is nothing else on this line, then return
LenRemainStr = Len(RemainStr)
If (LenRemainStr = 0) Then
    WriteOneFieldToWordDocument(vbCrLf, Colour_Black)
    Exit Sub
End If
' Test if first character of RemainStr is a semi-colon
FirstCharacter = Strings.Left(RemainStr, 1)
If (FirstCharacter = ";") Then
    ' The rest of the line is a comment
    WriteOneFieldToWordDocument(RemainStr, Colour_Comment)
    WriteOneFieldToWordDocument(vbCrLf, Colour_Black)
    Exit Sub
End If
' Find the end of the Verb
Loc = InStr(RemainStr, " ")
' Extract the Verb
If (Loc > 0) Then
    Verb = Strings.Left(RemainStr, Loc - 1)
    RemainStr = Strings.Right(RemainStr, Len(RemainStr) - Loc + 1)
Else
    Verb = RemainStr
    RemainStr = ""
End If
Do
    ' Test if the Verb is an Assembly OpCode
    For I As Int32 = 1 To NumOpCodeNames Step 1
        If (Verb = OpCodeNames(I)) Then
            ' The Verb is an Assembly OpCode
            WriteOneFieldToWordDocument(Verb, Colour_OpCodeName)
            Exit Do
        End If
    Next I
    ' Test if the Verb is a Compiler Operator
    For I As Int32 = 1 To NumCompOpNames Step 1
        If (Verb = CompOpNames(I)) Then
            ' The Verb is a Compiler Operator
            WriteOneFieldToWordDocument(Verb, Colour_CompOpName)
            Exit Do
        End If
    Next I
    ' The Verb cannot be classified
    MsgBox(
        "In line:" & vbCrLf & lLineIn & vbCrLf &
        "Error:" & vbCrLf & "The Verb is not recognized.")
    ' Close the Assembly file
    FileReader.Close()
    FileReader.Dispose()
    ' Close the Word application
    objWordDoc.Close()
    objWordApp.Quit()
    System.Runtime.InteropServices.Marshal.ReleaseComObject(objWordApp)
    Application.Exit()
    Exit Sub
Loop
```

```
' If there is nothing else on this line, then return
LenRemainStr = Len(RemainStr)
If (LenRemainStr = 0) Then
    WriteOneFieldToWordDocument(vbCrLf, Colour_Black)
    Exit Sub
End If
' Copy all blanks which come after the Verb
For I As Int32 = 1 To LenRemainStr Step 1
    If (Strings.Mid(RemainStr, I, 1) = " ") Then
        WriteOneFieldToWordDocument(" ", Colour_Black)
    Else
        Exit For
    End If
Next I
' Now remove the leading blanks from RemainStr
RemainStr = Trim(RemainStr)
' If there is nothing else on this line, then return
LenRemainStr = Len(RemainStr)
If (LenRemainStr = 0) Then
    WriteOneFieldToWordDocument(vbCrLf, Colour_Black)
    Exit Sub
End If
' Test if first character of RemainStr is a semi-colon
FirstCharacter = Strings.Left(RemainStr, 1)
If (FirstCharacter = ";") Then
    ' The rest of the line is a comment
    WriteOneFieldToWordDocument(RemainStr, Colour_Comment)
    WriteOneFieldToWordDocument(vbCrLf, Colour_Black)
    Exit Sub
End If
' Find the end of the Noun
Loc = InStr(RemainStr, " ")
' Extract the Noun
If (Loc > 0) Then
    Noun = Strings.Left(RemainStr, Loc - 1)
    RemainStr = Strings.Right(RemainStr, Len(RemainStr) - Loc + 1)
Else
    Noun = RemainStr
    RemainStr = ""
End If
' If the Noun contains a comma, then parse out the Argument
Loc = InStr(Noun, ",")
If (Loc > 0) Then
    ' The comma cannot be at the end of the Noun
    If (Loc = Len(RemainStr)) Then
        MsgBox(
            "In line:" & vbCrLf & lLineIn & vbCrLf &
            "Error:" & vbCrLf & "The Noun ends with a comma.")
        ' Close the Assembly file
        FileReader.Close()
        FileReader.Dispose()
        ' Close the Word application
        objWordDoc.Close()
        objWordApp.Quit()
        System.Runtime.InteropServices.Marshal.ReleaseComObject(objWordApp)
        Application.Exit()
        Exit Sub
    End If
```

```
        NounContainsComma = True
        Argument = Strings.Right(Noun, Len(Noun) - Loc)
        Noun = Strings.Left(Noun, Loc - 1)
    Else
        NounContainsComma = False
        Argument = ""
    End If
    Do
        ' Test if the Noun is a system register name
        For I As Int32 = 1 To NumSysRegNames Step 1
            If (Noun = SysRegNames(I)) Then
                ' The Noun is a system register name
                WriteOneFieldToWordDocument(Noun, Colour_SysRegName)
                Exit Do
            End If
        Next I
        ' Test if the Noun is a Compiler Noun name
        For I As Int32 = 1 To NumCompNounNames Step 1
            If (Noun = CompNounNames(I)) Then
                ' The Noun is a CompilerNoun name
                WriteOneFieldToWordDocument(Noun, Colour_CompNounName)
                Exit Do
            End If
        Next I
        ' Test if the Noun is a User label name
        For I As Int32 = 1 To NumUserLabelNames Step 1
            If (Noun = UserLabelNames(I)) Then
                ' The Noun is a User label name
                WriteOneFieldToWordDocument(Noun, Colour_UserLabelName)
                Exit Do
            End If
        Next I
        ' Test if the Noun is an integer from zero to seven
        If (Len(Noun) = 1) Then
            If ((CInt(Noun) >= 0) And (CInt(Noun) <= 7)) Then
                ' The Noun is an integer from zero to seven
                WriteOneFieldToWordDocument(Noun, Colour_Number)
                Exit Do
            End If
        End If
        ' Test if the Noun is a hex code
        If (Len(Noun) >= 2) Then
            If (Strings.Left(Noun, 2) = "0x") Then
                ' The Noun is a hex code
                WriteOneFieldToWordDocument(Noun, Colour_Number)
                Exit Do
            End If
        End If
        ' Test if the Noun is a binary number
        If (Len(Noun) >= 2) Then
            If (Strings.Left(Noun, 2) = "B'") Then
                ' The Noun is a binary number
                WriteOneFieldToWordDocument(Noun, Colour_Number)
                Exit Do
            End If
        End If
        ' Test if the Noun is a decimal number
        If (Len(Noun) >= 2) Then
```

```
            If (Strings.Left(Noun, 2) = "D'") Then
                ' The Noun is a decimal number
                WriteOneFieldToWordDocument(Noun, Colour_Number)
                Exit Do
            End If
        End If
        ' The Noun cannot be classified
        MsgBox(
            "In line:" & vbCrLf & lLineIn & vbCrLf &
            "Error:" & vbCrLf & "The Noun is not recognized.")
        ' Close the Assembly file
        FileReader.Close()
        FileReader.Dispose()
        ' Close the Word application
        objWordDoc.Close()
        objWordApp.Quit()
        System.Runtime.InteropServices.Marshal.ReleaseComObject(objWordApp)
        Application.Exit()
        Exit Sub
    Loop
    ' If the Noun has an Argument, then copy the comma
    If (NounContainsComma = True) Then
        ' The full Noun contains a comma
        WriteOneFieldToWordDocument(",", Colour_Black)
    End If
    ' If the Noun has an argument, then process it
    If (NounContainsComma = True) Then
        Do
            If (Argument <> "") Then
                ' Test if the Argument is a system register's bit name
                For I As Int32 = 1 To NumSysRegBitNames Step 1
                    If (Argument = SysRegBitNames(I)) Then
                        ' The Argument is a system register's bit name
                        WriteOneFieldToWordDocument(Argument, Colour_SysRegBitName)
                        Exit Do
                    End If
                Next I
                ' Test if the Argument is an Assembly destination name
                For I As Int32 = 1 To NumDestNames Step 1
                    If (Argument = DestNames(I)) Then
                        ' The Argument is an Assembly destination name
                        WriteOneFieldToWordDocument(Argument, Colour_DestName)
                        Exit Do
                    End If
                Next I
                ' Test if the Argument is a Compiler Noun name
                For I As Int32 = 1 To NumCompNounNames Step 1
                    If (Argument = CompNounNames(I)) Then
                        ' The argument is a Compiler Noun name
                        WriteOneFieldToWordDocument(Argument, Colour_CompNounName)
                        Exit Do
                    End If
                Next I
                ' Test if the Argument is a User label name
                For I As Int32 = 1 To NumUserLabelNames Step 1
                    If (Argument = UserLabelNames(I)) Then
                        ' The argument is a User label name
                        WriteOneFieldToWordDocument(Argument, Colour_UserLabelName)
```

```
                    Exit Do
                End If
            Next I
            ' Test if the Argument is an integer from zero to seven
            If (Len(Argument) = 1) Then
                If ((CInt(Argument) >= 0) And (CInt(Argument) <= 7)) Then
                    ' The argument is an integer from zero to seven
                    WriteOneFieldToWordDocument(Argument, Colour_Number)
                    Exit Do
                End If
            End If
            ' Test if the Argument is a hex code
            If (Len(Argument) >= 2) Then
                If (Strings.Left(Argument, 2) = "0x") Then
                    ' The Argument is a hex code
                    WriteOneFieldToWordDocument(Argument, Colour_Number)
                    Exit Do
                End If
            End If
            ' Test if the Argument is a binary number
            If (Len(Argument) >= 2) Then
                If (Strings.Left(Argument, 2) = "B'") Then
                    ' The Argument is a binary number
                    WriteOneFieldToWordDocument(Argument, Colour_Number)
                    Exit Do
                End If
            End If
            ' Test if the Argument is a decimal number
            If (Len(Argument) >= 2) Then
                If (Strings.Left(Argument, 2) = "D'") Then
                    ' The Argument is a decimal number
                    WriteOneFieldToWordDocument(Argument, Colour_Number)
                    Exit Do
                End If
            End If
            ' The Argument cannot be classified
            MsgBox(
            "In line:" & vbCrLf & lLineIn & vbCrLf &
            "Error:" & vbCrLf & "The Argument is not recognized.")
            ' Close the Assembly file
            FileReader.Close()
            FileReader.Dispose()
            ' Close the Word application
            objWordDoc.Close()
            objWordApp.Quit()
            System.Runtime.InteropServices.Marshal.ReleaseComObject(objWordApp)
            Application.Exit()
            Exit Sub
        End If
    Loop
End If
' If there is nothing else on this line, then return
LenRemainStr = Len(RemainStr)
If (LenRemainStr = 0) Then
    WriteOneFieldToWordDocument(vbCrLf, Colour_Black)
    Exit Sub
End If
' Copy all blanks which come after the Noun/Argument
```

```
    For I As Int32 = 1 To LenRemainStr Step 1
        If (Strings.Mid(RemainStr, I, 1) = " ") Then
            WriteOneFieldToWordDocument(" ", Colour_Black)
        Else
            Exit For
        End If
    Next I
    ' Now remove the leading blanks from RemainStr
    RemainStr = Trim(RemainStr)
    ' If there is nothing else on this line, then return
    LenRemainStr = Len(RemainStr)
    If (LenRemainStr = 0) Then
        WriteOneFieldToWordDocument(vbCrLf, Colour_Black)
        Exit Sub
    End If
    ' Test if first character of RemainStr is a semi-colon
    FirstCharacter = Strings.Left(RemainStr, 1)
    If (FirstCharacter = ";") Then
        ' The rest of the line is a comment
        WriteOneFieldToWordDocument(RemainStr, Colour_Comment)
        WriteOneFieldToWordDocument(vbCrLf, Colour_Black)
        Exit Sub
    Else
        ' This is unexpected conclusion to the input line
        MsgBox(
            "In line:" & vbCrLf & lLineIn & vbCrLf &
            "Error:" & vbCrLf & "The line has an unexpected conclusion.")
        ' Close the Assembly file
        FileReader.Close()
        FileReader.Dispose()
        ' Close the Word application
        objWordDoc.Close()
        objWordApp.Quit()
        System.Runtime.InteropServices.Marshal.ReleaseComObject(objWordApp)
        Application.Exit()
        Exit Sub
    End If
End Sub

Public Function ReplaceTabsWithBlanks(ByVal lLineIn As String) As String
    Dim lLenLineIn As Int32
    Dim lLineOut As String = ""
    Dim lLenLineOut As Int32 = 0
    Dim lCharStr As String = ""
    Dim lCharInt As Int32
    Dim lIndexOfTab As Int32
    Dim lNumOfBlanks As Int32
    ' Return immediately if the input line is nil
    lLenLineIn = Len(lLineIn)
    If (lLenLineIn = 0) Then
        Return ""
    End If
    ' Process each character in the input line
    For I As Int32 = 1 To lLenLineIn Step 1
        lCharStr = Strings.Mid(lLineIn, I, 1)
        lCharInt = Asc(lCharStr)
        ' Test if the character is a Tab
        If (lCharInt <> 9) Then
```

```
                ' The character is not a Tab, so copy it to the output line
                lLineOut = lLineOut & lCharStr
                lLenLineOut = lLenLineOut + 1
            Else
                ' The character is a Tab, so search for the next greater Tab stop
                lIndexOfTab = -1
                For J As Int32 = 1 To NumTabStops Step 1
                    If (TabStops(J) > lLenLineOut) Then
                        lIndexOfTab = J
                        Exit For
                    End If
                Next J
                ' Calculate the number of blanks needed
                lNumOfBlanks = TabStops(lIndexOfTab) - lLenLineOut
                ' Add the required number of blanks to the output line
                lLineOut = lLineOut & Strings.Space(lNumOfBlanks)
                ' Update the length of the output line
                lLenLineOut = lLenLineOut + lNumOfBlanks
            End If
        Next I
        Return lLineOut
    End Function

    Public Sub WriteOneFieldToWordDocument(
        ByVal lField As String, ByVal lColour As Microsoft.Office.Interop.Word.WdColor)
        ' Set the insertion point to the end of the document
        objWordRange.Collapse(
            Microsoft.Office.Interop.Word.WdCollapseDirection.wdCollapseEnd)
        ' Write the field
        objWordRange.InsertAfter(lField)
        ' The range now includes the field.  Change its colour.
        objWordRange.Font.Color = lColour
    End Sub

End Module
```

## Appendix "D"
## Module Procedures.vb

```
' Option Strict On
Option Explicit On

' List of subroutines:-
'   MainProgram()
'   DefineAllSystemRegisterNames()
'   DefineAllSystemRegisterBitNames()
'   DefineAllOpCodeNames()
'   DefineAllDestinationNames()
'   DefineAllCompilerOperatorNames()
'   DefineAllCompilerNounNames()
'   SearchForAllUserLabelNames()
'   DefineAllTabStops()

Public Module Procedures

    Public FileReader As System.IO.StreamReader
    Public LineAsRead As String = ""
    Public objWordApp As Microsoft.Office.Interop.Word.Application
    Public objWordDoc As Microsoft.Office.Interop.Word.Document
    Public objWordPar As Microsoft.Office.Interop.Word.Paragraph
    Public objWordRange As Microsoft.Office.Interop.Word.Range
    Public DefaultStyle As Microsoft.Office.Interop.Word.Style

    Public Sub MainProgram()
        ' Open the Assembly file
        FileReader = My.Computer.FileSystem.OpenTextFileReader(MainDir & ASMFileName)
        ' Delete the Word file if it already exists
        If System.IO.File.Exists(MainDir & WordFileName) Then
            System.IO.File.Decrypt(MainDir & WordFileName)
        End If
        ' Create a new Word document
        objWordApp = CreateObject("Word.Application")
        objWordApp.Visible = False
        objWordDoc = objWordApp.Documents.Add
        ' Set the page properties
        objWordDoc.PageSetup.PageWidth = CInt(PageWidthInch * 72)
        objWordDoc.PageSetup.PageHeight = CInt(PageHeightInch * 72)
        If (IsOrientationPortrait = True) Then
            objWordDoc.PageSetup.Orientation =
                Microsoft.Office.Interop.Word.WdOrientation.wdOrientPortrait
        Else
            objWordDoc.PageSetup.Orientation =
                Microsoft.Office.Interop.Word.WdOrientation.wdOrientLandscape
        End If
        objWordDoc.PageSetup.LeftMargin = CInt(LeftMarginInch * 72)
        objWordDoc.PageSetup.RightMargin = CInt(RightMarginInch * 72)
        objWordDoc.PageSetup.TopMargin = CInt(TopMarginInch * 72)
        objWordDoc.PageSetup.BottomMargin = CInt(BottomMarginInch)
        ' Set the default paragraph properties
        objWordDoc.Paragraphs.Alignment =
            Microsoft.Office.Interop.Word.WdParagraphAlignment.wdAlignParagraphLeft
        objWordDoc.Paragraphs.FirstLineIndent = 0
        objWordDoc.Paragraphs.LineSpacingRule =
```

```
            Microsoft.Office.Interop.Word.WdLineSpacing.wdLineSpaceSingle
        objWordDoc.Paragraphs.LineUnitBefore = 0
        objWordDoc.Paragraphs.LineUnitAfter = 0
        objWordDoc.Paragraphs.SpaceBefore = 0
        objWordDoc.Paragraphs.SpaceAfter = 0
        ' Set the default font
        DefaultStyle = objWordDoc.Styles(
            Microsoft.Office.Interop.Word.WdBuiltinStyle.wdStyleNormal)
        With DefaultStyle.Font
            .Name = FontName
            .Size = FontSize
            .Bold = IsFontBold
            .Italic = IsFontItalic
            .ColorIndex = Colour_Default
        End With
        ' Instantiate the objWordRange
        objWordRange = objWordDoc.Range
        ' Expose the label showing the line counter
        Dim LineCounter As Int32 = 0
        Form1.labelLineCounter.Visible = True
        Form1.labelLineCounter.Text = "Line number " & Trim(Str(LineCounter))
        Form1.labelLineCounter.Refresh()
        ' Main loop
        Do While (Not FileReader.EndOfStream)
            ' Update the line counter in the display
            LineCounter = LineCounter + 1
            Form1.labelLineCounter.Text = "Line number " & Trim(Str(LineCounter))
            Form1.labelLineCounter.Refresh()
            ' Read the next line in the Assembly file
            LineAsRead = FileReader.ReadLine
            ' Parse and write this line
            ParseAndWrite.ParseAndWriteOneAssemblyLine(LineAsRead)
            ' Give the User a chance to stop the program
            Application.DoEvents()
        Loop
        ' Save the Word file
        objWordDoc.SaveAs(MainDir & WordFileName)
        ' Close the Assembly file
        FileReader.Close()
        FileReader.Dispose()
        ' Close the Word application
        objWordDoc.Close()
        objWordApp.Quit()
        System.Runtime.InteropServices.Marshal.ReleaseComObject(objWordApp)
        ' Alert the User
        MsgBox("All finished")
    End Sub

    Public Sub DefineAllSystemRegisterNames()
        SysRegNames(1) = "TMR0"
        SysRegNames(2) = "PCL"
        SysRegNames(3) = "STATUS"
        SysRegNames(4) = "FSR"
        SysRegNames(5) = "portA"
        SysRegNames(6) = "portB"
        SysRegNames(7) = "portC"
        SysRegNames(8) = "portE"
        SysRegNames(9) = "PCLATH"
```

```
SysRegNames(10) = "INTCON"
SysRegNames(11) = "PIR1"
SysRegNames(12) = "PIR2"
SysRegNames(13) = "TMR1L"
SysRegNames(14) = "TMR1H"
SysRegNames(15) = "T1CON"
SysRegNames(16) = "TMR2"
SysRegNames(17) = "T2CON"
SysRegNames(18) = "SSPBUF"
SysRegNames(19) = "SSPCON"
SysRegNames(20) = "CCPR1L"
SysRegNames(21) = "CCPR1H"
SysRegNames(22) = "CCP1CON"
SysRegNames(23) = "RCSTA"
SysRegNames(24) = "TXREG"
SysRegNames(25) = "RCREG"
SysRegNames(26) = "CCPR2L"
SysRegNames(27) = "CCPR2H"
SysRegNames(28) = "CCP2CON"
SysRegNames(29) = "ADRESH"
SysRegNames(30) = "ADCON0"
SysRegNames(31) = "OPTION_REG"
SysRegNames(32) = "TRISA"
SysRegNames(33) = "TRISB"
SysRegNames(34) = "TRISC"
SysRegNames(35) = "TRISE"
SysRegNames(36) = "PIE1"
SysRegNames(37) = "PIE2"
SysRegNames(38) = "PCON"
SysRegNames(39) = "OSCCON"
SysRegNames(40) = "OSCTUNE"
SysRegNames(41) = "SSPCON2"
SysRegNames(42) = "PR2"
SysRegNames(43) = "SSPADD"
SysRegNames(44) = "SSPSTAT"
SysRegNames(45) = "WPUB"
SysRegNames(46) = "IOCB"
SysRegNames(47) = "VRCON"
SysRegNames(48) = "TXSTA"
SysRegNames(49) = "SPBRG"
SysRegNames(50) = "SPBRGH"
SysRegNames(51) = "PWM1CON"
SysRegNames(52) = "ECCPAS"
SysRegNames(53) = "PSTRCON"
SysRegNames(54) = "ADRESL"
SysRegNames(55) = "ADCON1"
SysRegNames(56) = "WDTCON"
SysRegNames(57) = "CM1CON0"
SysRegNames(58) = "CM2CON0"
SysRegNames(59) = "CM2CON1"
SysRegNames(60) = "EEDAT"
SysRegNames(61) = "EEADR"
SysRegNames(62) = "EEDATH"
SysRegNames(63) = "EEADRH"
SysRegNames(64) = "SRCON"
SysRegNames(65) = "BAUDCTL"
SysRegNames(66) = "ANSEL"
SysRegNames(67) = "ANSELH"
```

```
    SysRegNames(68) = "EECON1"
    SysRegNames(69) = "EECON2"
    NumSysRegNames = 69
End Sub

Public Sub DefineAllSystemRegisterBitNames()
    NumSysRegBitNames = 0
    ' STATUS 0x03
    SysRegBitNames(NumSysRegBitNames + 1) = "irp"
    SysRegBitNames(NumSysRegBitNames + 2) = "page1"
    SysRegBitNames(NumSysRegBitNames + 3) = "page0"
    SysRegBitNames(NumSysRegBitNames + 4) = "t0"
    SysRegBitNames(NumSysRegBitNames + 5) = "pd"
    SysRegBitNames(NumSysRegBitNames + 6) = "zero"
    SysRegBitNames(NumSysRegBitNames + 7) = "dc"
    SysRegBitNames(NumSysRegBitNames + 8) = "carry"
    NumSysRegBitNames = NumSysRegBitNames + 8
    ' portA 0x05
    SysRegBitNames(NumSysRegBitNames + 1) = "ra7"
    SysRegBitNames(NumSysRegBitNames + 2) = "ra6"
    SysRegBitNames(NumSysRegBitNames + 3) = "ra5"
    SysRegBitNames(NumSysRegBitNames + 4) = "ra4"
    SysRegBitNames(NumSysRegBitNames + 5) = "ra3"
    SysRegBitNames(NumSysRegBitNames + 6) = "ra2"
    SysRegBitNames(NumSysRegBitNames + 7) = "ra1"
    SysRegBitNames(NumSysRegBitNames + 8) = "ra0"
    NumSysRegBitNames = NumSysRegBitNames + 8
    ' portB 0x06
    SysRegBitNames(NumSysRegBitNames + 1) = "rb7"
    SysRegBitNames(NumSysRegBitNames + 2) = "rb6"
    SysRegBitNames(NumSysRegBitNames + 3) = "rb5"
    SysRegBitNames(NumSysRegBitNames + 4) = "rb4"
    SysRegBitNames(NumSysRegBitNames + 5) = "rb3"
    SysRegBitNames(NumSysRegBitNames + 6) = "rb2"
    SysRegBitNames(NumSysRegBitNames + 7) = "rb1"
    SysRegBitNames(NumSysRegBitNames + 8) = "rb0"
    NumSysRegBitNames = NumSysRegBitNames + 8
    ' portC 0x07
    SysRegBitNames(NumSysRegBitNames + 1) = "rc7"
    SysRegBitNames(NumSysRegBitNames + 2) = "rc6"
    SysRegBitNames(NumSysRegBitNames + 3) = "rc5"
    SysRegBitNames(NumSysRegBitNames + 4) = "rc4"
    SysRegBitNames(NumSysRegBitNames + 5) = "rc3"
    SysRegBitNames(NumSysRegBitNames + 6) = "rc2"
    SysRegBitNames(NumSysRegBitNames + 7) = "rc1"
    SysRegBitNames(NumSysRegBitNames + 8) = "rc0"
    NumSysRegBitNames = NumSysRegBitNames + 8
    ' portE 0x09
    SysRegBitNames(NumSysRegBitNames + 1) = "re3"
    SysRegBitNames(NumSysRegBitNames + 2) = "re2"
    SysRegBitNames(NumSysRegBitNames + 3) = "re1"
    SysRegBitNames(NumSysRegBitNames + 4) = "re0"
    NumSysRegBitNames = NumSysRegBitNames + 4
    ' INTCON 0x0B
    SysRegBitNames(NumSysRegBitNames + 1) = "gie"
    SysRegBitNames(NumSysRegBitNames + 2) = "peie"
    SysRegBitNames(NumSysRegBitNames + 3) = "tmr0ie"
    SysRegBitNames(NumSysRegBitNames + 4) = "inte"
```

```
SysRegBitNames(NumSysRegBitNames + 5) = "rbie"
SysRegBitNames(NumSysRegBitNames + 6) = "tmr0if"
SysRegBitNames(NumSysRegBitNames + 7) = "intf"
SysRegBitNames(NumSysRegBitNames + 8) = "rbif"
NumSysRegBitNames = NumSysRegBitNames + 8
' PIR1 0x0C
SysRegBitNames(NumSysRegBitNames + 1) = "adif"
SysRegBitNames(NumSysRegBitNames + 2) = "rcif"
SysRegBitNames(NumSysRegBitNames + 3) = "txif"
SysRegBitNames(NumSysRegBitNames + 4) = "sspif"
SysRegBitNames(NumSysRegBitNames + 5) = "ccp1if"
SysRegBitNames(NumSysRegBitNames + 6) = "tmr2if"
SysRegBitNames(NumSysRegBitNames + 7) = "tmr1if"
NumSysRegBitNames = NumSysRegBitNames + 7
' PIR2 0x0D
SysRegBitNames(NumSysRegBitNames + 1) = "osfif"
SysRegBitNames(NumSysRegBitNames + 2) = "c2if"
SysRegBitNames(NumSysRegBitNames + 3) = "c1if"
SysRegBitNames(NumSysRegBitNames + 4) = "eeif"
SysRegBitNames(NumSysRegBitNames + 5) = "bclif"
SysRegBitNames(NumSysRegBitNames + 6) = "ulpwuif"
SysRegBitNames(NumSysRegBitNames + 7) = "ccp2if"
NumSysRegBitNames = NumSysRegBitNames + 7
' T1CON 0x10
SysRegBitNames(NumSysRegBitNames + 1) = "t1ginv"
SysRegBitNames(NumSysRegBitNames + 2) = "tmr1ge"
SysRegBitNames(NumSysRegBitNames + 3) = "t1ckps1"
SysRegBitNames(NumSysRegBitNames + 4) = "t1ckps0"
SysRegBitNames(NumSysRegBitNames + 5) = "t1oscen"
SysRegBitNames(NumSysRegBitNames + 6) = "t1sync"
SysRegBitNames(NumSysRegBitNames + 7) = "tmr1cs"
SysRegBitNames(NumSysRegBitNames + 8) = "tmr1on"
NumSysRegBitNames = NumSysRegBitNames + 8
' T2CON 0x12
SysRegBitNames(NumSysRegBitNames + 1) = "toutps3"
SysRegBitNames(NumSysRegBitNames + 2) = "toups2"
SysRegBitNames(NumSysRegBitNames + 3) = "toutps1"
SysRegBitNames(NumSysRegBitNames + 4) = "toutps0"
SysRegBitNames(NumSysRegBitNames + 5) = "tmr2on"
SysRegBitNames(NumSysRegBitNames + 6) = "t2ckps1"
SysRegBitNames(NumSysRegBitNames + 7) = "t2ckps0"
NumSysRegBitNames = NumSysRegBitNames + 7
' SSPCON 0x14
SysRegBitNames(NumSysRegBitNames + 1) = "wcol"
SysRegBitNames(NumSysRegBitNames + 2) = "sspov"
SysRegBitNames(NumSysRegBitNames + 3) = "sspen"
SysRegBitNames(NumSysRegBitNames + 4) = "ckp"
SysRegBitNames(NumSysRegBitNames + 5) = "sspm3"
SysRegBitNames(NumSysRegBitNames + 6) = "sspm2"
SysRegBitNames(NumSysRegBitNames + 7) = "sspm1"
SysRegBitNames(NumSysRegBitNames + 8) = "sspm0"
NumSysRegBitNames = NumSysRegBitNames + 8
' CCP1CON 0x17
SysRegBitNames(NumSysRegBitNames + 1) = "p1m1"
SysRegBitNames(NumSysRegBitNames + 2) = "p1m0"
SysRegBitNames(NumSysRegBitNames + 3) = "dc1b1"
SysRegBitNames(NumSysRegBitNames + 4) = "dc1b0"
SysRegBitNames(NumSysRegBitNames + 5) = "ccp1m3"
```

```
SysRegBitNames(NumSysRegBitNames + 6) = "ccp1m2"
SysRegBitNames(NumSysRegBitNames + 7) = "ccp1m1"
SysRegBitNames(NumSysRegBitNames + 8) = "ccp1m0"
NumSysRegBitNames = NumSysRegBitNames + 8
' RCSTA 0x18
SysRegBitNames(NumSysRegBitNames + 1) = "spen"
SysRegBitNames(NumSysRegBitNames + 2) = "rx9"
SysRegBitNames(NumSysRegBitNames + 3) = "sren"
SysRegBitNames(NumSysRegBitNames + 4) = "cren"
SysRegBitNames(NumSysRegBitNames + 5) = "adden"
SysRegBitNames(NumSysRegBitNames + 6) = "ferr"
SysRegBitNames(NumSysRegBitNames + 7) = "oerr"
SysRegBitNames(NumSysRegBitNames + 8) = "rx9d"
NumSysRegBitNames = NumSysRegBitNames + 8
' CCP2CON 0x1D
SysRegBitNames(NumSysRegBitNames + 1) = "dc2b1"
SysRegBitNames(NumSysRegBitNames + 2) = "dc2b0"
SysRegBitNames(NumSysRegBitNames + 3) = "ccp2m3"
SysRegBitNames(NumSysRegBitNames + 4) = "ccp2m2"
SysRegBitNames(NumSysRegBitNames + 5) = "ccp2m1"
SysRegBitNames(NumSysRegBitNames + 6) = "ccp2m0"
NumSysRegBitNames = NumSysRegBitNames + 6
' ADRESH 0x1E
SysRegBitNames(NumSysRegBitNames + 1) = "adres9"
SysRegBitNames(NumSysRegBitNames + 2) = "adres8"
SysRegBitNames(NumSysRegBitNames + 3) = "adres7"
SysRegBitNames(NumSysRegBitNames + 4) = "adres6"
SysRegBitNames(NumSysRegBitNames + 5) = "adres5"
SysRegBitNames(NumSysRegBitNames + 6) = "adres4"
SysRegBitNames(NumSysRegBitNames + 7) = "adres3"
SysRegBitNames(NumSysRegBitNames + 8) = "adres2"
NumSysRegBitNames = NumSysRegBitNames + 8
' ADCON0 0x1F
SysRegBitNames(NumSysRegBitNames + 1) = "adcs1"
SysRegBitNames(NumSysRegBitNames + 2) = "adcs0"
SysRegBitNames(NumSysRegBitNames + 3) = "chs3"
SysRegBitNames(NumSysRegBitNames + 4) = "chs2"
SysRegBitNames(NumSysRegBitNames + 5) = "chs1"
SysRegBitNames(NumSysRegBitNames + 6) = "chs0"
SysRegBitNames(NumSysRegBitNames + 7) = "go_done"
SysRegBitNames(NumSysRegBitNames + 8) = "adon"
NumSysRegBitNames = NumSysRegBitNames + 8
' OPTION_REG 0x81
SysRegBitNames(NumSysRegBitNames + 1) = "rpbu"
SysRegBitNames(NumSysRegBitNames + 2) = "intedg"
SysRegBitNames(NumSysRegBitNames + 3) = "t0cs"
SysRegBitNames(NumSysRegBitNames + 4) = "t0se"
SysRegBitNames(NumSysRegBitNames + 5) = "psa"
SysRegBitNames(NumSysRegBitNames + 6) = "ps2"
SysRegBitNames(NumSysRegBitNames + 7) = "ps1"
SysRegBitNames(NumSysRegBitNames + 8) = "ps0"
NumSysRegBitNames = NumSysRegBitNames + 8
' TRISA 0x85
SysRegBitNames(NumSysRegBitNames + 1) = "trisa7"
SysRegBitNames(NumSysRegBitNames + 2) = "trisa6"
SysRegBitNames(NumSysRegBitNames + 3) = "trisa5"
SysRegBitNames(NumSysRegBitNames + 4) = "trisa4"
SysRegBitNames(NumSysRegBitNames + 5) = "trisa3"
```

```
SysRegBitNames(NumSysRegBitNames + 6) = "trisa2"
SysRegBitNames(NumSysRegBitNames + 7) = "trisa1"
SysRegBitNames(NumSysRegBitNames + 8) = "trisa0"
NumSysRegBitNames = NumSysRegBitNames + 8
' TRISB 0x86
SysRegBitNames(NumSysRegBitNames + 1) = "trisb7"
SysRegBitNames(NumSysRegBitNames + 2) = "trisb6"
SysRegBitNames(NumSysRegBitNames + 3) = "trisb5"
SysRegBitNames(NumSysRegBitNames + 4) = "trisb4"
SysRegBitNames(NumSysRegBitNames + 5) = "trisb3"
SysRegBitNames(NumSysRegBitNames + 6) = "trisb2"
SysRegBitNames(NumSysRegBitNames + 7) = "trisb1"
SysRegBitNames(NumSysRegBitNames + 8) = "trisb0"
NumSysRegBitNames = NumSysRegBitNames + 8
' TRISC 0x87
SysRegBitNames(NumSysRegBitNames + 1) = "trisc7"
SysRegBitNames(NumSysRegBitNames + 2) = "trisc6"
SysRegBitNames(NumSysRegBitNames + 3) = "trisc5"
SysRegBitNames(NumSysRegBitNames + 4) = "trisc4"
SysRegBitNames(NumSysRegBitNames + 5) = "trisc3"
SysRegBitNames(NumSysRegBitNames + 6) = "trisc2"
SysRegBitNames(NumSysRegBitNames + 7) = "trisc1"
SysRegBitNames(NumSysRegBitNames + 8) = "trisc0"
NumSysRegBitNames = NumSysRegBitNames + 8
' TRISE 0x89
SysRegBitNames(NumSysRegBitNames + 1) = "trise3"
SysRegBitNames(NumSysRegBitNames + 2) = "trise2"
SysRegBitNames(NumSysRegBitNames + 3) = "trise1"
SysRegBitNames(NumSysRegBitNames + 4) = "trise0"
NumSysRegBitNames = NumSysRegBitNames + 4
' PIE1 0x8C
SysRegBitNames(NumSysRegBitNames + 1) = "adie"
SysRegBitNames(NumSysRegBitNames + 2) = "rcie"
SysRegBitNames(NumSysRegBitNames + 3) = "txie"
SysRegBitNames(NumSysRegBitNames + 4) = "sspi"
SysRegBitNames(NumSysRegBitNames + 5) = "ccp1ie"
SysRegBitNames(NumSysRegBitNames + 6) = "tmr2ie"
SysRegBitNames(NumSysRegBitNames + 7) = "tmr1ie"
NumSysRegBitNames = NumSysRegBitNames + 7
' PIE2 0x8D
SysRegBitNames(NumSysRegBitNames + 1) = "osfie"
SysRegBitNames(NumSysRegBitNames + 2) = "c2ie"
SysRegBitNames(NumSysRegBitNames + 3) = "c1ie"
SysRegBitNames(NumSysRegBitNames + 4) = "eeie"
SysRegBitNames(NumSysRegBitNames + 5) = "bclie"
SysRegBitNames(NumSysRegBitNames + 6) = "ulpwuie"
SysRegBitNames(NumSysRegBitNames + 7) = "ccp2ie"
NumSysRegBitNames = NumSysRegBitNames + 7
' PCON 0x8E
SysRegBitNames(NumSysRegBitNames + 1) = "ulpwue"
SysRegBitNames(NumSysRegBitNames + 2) = "sboren"
SysRegBitNames(NumSysRegBitNames + 3) = "por"
SysRegBitNames(NumSysRegBitNames + 4) = "bor"
NumSysRegBitNames = NumSysRegBitNames + 4
' OSCCON 0x8F
SysRegBitNames(NumSysRegBitNames + 1) = "ircf2"
SysRegBitNames(NumSysRegBitNames + 2) = "ircf1"
SysRegBitNames(NumSysRegBitNames + 3) = "ircf0"
```

```
SysRegBitNames(NumSysRegBitNames + 4) = "osts"
SysRegBitNames(NumSysRegBitNames + 5) = "hts"
SysRegBitNames(NumSysRegBitNames + 6) = "lts"
SysRegBitNames(NumSysRegBitNames + 7) = "scs"
NumSysRegBitNames = NumSysRegBitNames + 7
' OSCTUNE 0x90
SysRegBitNames(NumSysRegBitNames + 1) = "tun4"
SysRegBitNames(NumSysRegBitNames + 2) = "tun3"
SysRegBitNames(NumSysRegBitNames + 3) = "tun2"
SysRegBitNames(NumSysRegBitNames + 4) = "tun1"
SysRegBitNames(NumSysRegBitNames + 5) = "tun0"
NumSysRegBitNames = NumSysRegBitNames + 5
' SSPCON2 0x91
SysRegBitNames(NumSysRegBitNames + 1) = "gcen"
SysRegBitNames(NumSysRegBitNames + 2) = "ackstat"
SysRegBitNames(NumSysRegBitNames + 3) = "ackdt"
SysRegBitNames(NumSysRegBitNames + 4) = "acken"
SysRegBitNames(NumSysRegBitNames + 5) = "rcen"
SysRegBitNames(NumSysRegBitNames + 6) = "pen"
SysRegBitNames(NumSysRegBitNames + 7) = "rsen"
SysRegBitNames(NumSysRegBitNames + 8) = "sen"
NumSysRegBitNames = NumSysRegBitNames + 8
' SSPSTAT 0x94
SysRegBitNames(NumSysRegBitNames + 1) = "smp"
SysRegBitNames(NumSysRegBitNames + 2) = "cke"
SysRegBitNames(NumSysRegBitNames + 3) = "d_a"
SysRegBitNames(NumSysRegBitNames + 4) = "p"
SysRegBitNames(NumSysRegBitNames + 5) = "s"
SysRegBitNames(NumSysRegBitNames + 6) = "r_w"
SysRegBitNames(NumSysRegBitNames + 7) = "ua"
SysRegBitNames(NumSysRegBitNames + 8) = "bf"
NumSysRegBitNames = NumSysRegBitNames + 8
' WPUB 0x95
SysRegBitNames(NumSysRegBitNames + 1) = "wpub7"
SysRegBitNames(NumSysRegBitNames + 2) = "wpub6"
SysRegBitNames(NumSysRegBitNames + 3) = "wpub5"
SysRegBitNames(NumSysRegBitNames + 4) = "wpub4"
SysRegBitNames(NumSysRegBitNames + 5) = "wpub3"
SysRegBitNames(NumSysRegBitNames + 6) = "wpub2"
SysRegBitNames(NumSysRegBitNames + 7) = "wpub1"
SysRegBitNames(NumSysRegBitNames + 8) = "wpub0"
NumSysRegBitNames = NumSysRegBitNames + 8
' IOCB 0x96
SysRegBitNames(NumSysRegBitNames + 1) = "iocb7"
SysRegBitNames(NumSysRegBitNames + 2) = "iocb6"
SysRegBitNames(NumSysRegBitNames + 3) = "iocb5"
SysRegBitNames(NumSysRegBitNames + 4) = "iocb4"
SysRegBitNames(NumSysRegBitNames + 5) = "iocb3"
SysRegBitNames(NumSysRegBitNames + 6) = "iocb2"
SysRegBitNames(NumSysRegBitNames + 7) = "iocb1"
SysRegBitNames(NumSysRegBitNames + 8) = "iocb0"
NumSysRegBitNames = NumSysRegBitNames + 8
' VRCON 0x97
SysRegBitNames(NumSysRegBitNames + 1) = "vren"
SysRegBitNames(NumSysRegBitNames + 2) = "vroe"
SysRegBitNames(NumSysRegBitNames + 3) = "vrr"
SysRegBitNames(NumSysRegBitNames + 4) = "vrss"
SysRegBitNames(NumSysRegBitNames + 5) = "vr3"
```

```
SysRegBitNames(NumSysRegBitNames + 6) = "vr2"
SysRegBitNames(NumSysRegBitNames + 7) = "vr1"
SysRegBitNames(NumSysRegBitNames + 8) = "vr0"
NumSysRegBitNames = NumSysRegBitNames + 8
' TXSTA 0x98
SysRegBitNames(NumSysRegBitNames + 1) = "csrc"
SysRegBitNames(NumSysRegBitNames + 2) = "tx9"
SysRegBitNames(NumSysRegBitNames + 3) = "txen"
SysRegBitNames(NumSysRegBitNames + 4) = "sync"
SysRegBitNames(NumSysRegBitNames + 5) = "sendb"
SysRegBitNames(NumSysRegBitNames + 6) = "brgh"
SysRegBitNames(NumSysRegBitNames + 7) = "trmt"
SysRegBitNames(NumSysRegBitNames + 8) = "tx9d"
NumSysRegBitNames = NumSysRegBitNames + 8
' PWM1CON 0x9B
SysRegBitNames(NumSysRegBitNames + 1) = "prsen"
SysRegBitNames(NumSysRegBitNames + 2) = "pdc6"
SysRegBitNames(NumSysRegBitNames + 3) = "pdc5"
SysRegBitNames(NumSysRegBitNames + 4) = "pdc4"
SysRegBitNames(NumSysRegBitNames + 5) = "pdc3"
SysRegBitNames(NumSysRegBitNames + 6) = "pdc2"
SysRegBitNames(NumSysRegBitNames + 7) = "pdc1"
SysRegBitNames(NumSysRegBitNames + 8) = "pdc0"
NumSysRegBitNames = NumSysRegBitNames + 8
' ECCPAS 0x9C
SysRegBitNames(NumSysRegBitNames + 1) = "eccpase"
SysRegBitNames(NumSysRegBitNames + 2) = "eccpas2"
SysRegBitNames(NumSysRegBitNames + 3) = "eccpas1"
SysRegBitNames(NumSysRegBitNames + 4) = "eccpas0"
SysRegBitNames(NumSysRegBitNames + 5) = "pssac1"
SysRegBitNames(NumSysRegBitNames + 6) = "pssac0"
SysRegBitNames(NumSysRegBitNames + 7) = "pssbd1"
SysRegBitNames(NumSysRegBitNames + 8) = "pssbd0"
NumSysRegBitNames = NumSysRegBitNames + 8
' PSTRCON 0x9D
SysRegBitNames(NumSysRegBitNames + 1) = "strsync"
SysRegBitNames(NumSysRegBitNames + 2) = "strd"
SysRegBitNames(NumSysRegBitNames + 3) = "strc"
SysRegBitNames(NumSysRegBitNames + 4) = "strb"
SysRegBitNames(NumSysRegBitNames + 5) = "stra"
NumSysRegBitNames = NumSysRegBitNames + 5
' ADRESL 0x9E
SysRegBitNames(NumSysRegBitNames + 1) = "adres1"
SysRegBitNames(NumSysRegBitNames + 2) = "adres0"
NumSysRegBitNames = NumSysRegBitNames + 2
' ADCON1 0x9F
SysRegBitNames(NumSysRegBitNames + 1) = "adfm"
SysRegBitNames(NumSysRegBitNames + 2) = "vcfg1"
SysRegBitNames(NumSysRegBitNames + 3) = "vcfg0"
NumSysRegBitNames = NumSysRegBitNames + 3
' WDTCON 0x105
SysRegBitNames(NumSysRegBitNames + 1) = "wdtps3"
SysRegBitNames(NumSysRegBitNames + 2) = "wdtps2"
SysRegBitNames(NumSysRegBitNames + 3) = "wdtps1"
SysRegBitNames(NumSysRegBitNames + 4) = "wdtps0"
SysRegBitNames(NumSysRegBitNames + 5) = "swdten"
NumSysRegBitNames = NumSysRegBitNames + 5
' CM1CON0 0x107
```

```
SysRegBitNames(NumSysRegBitNames + 1) = "c1On"
SysRegBitNames(NumSysRegBitNames + 2) = "c1out"
SysRegBitNames(NumSysRegBitNames + 3) = "c1oe"
SysRegBitNames(NumSysRegBitNames + 4) = "c1pol"
SysRegBitNames(NumSysRegBitNames + 5) = "c1R"
SysRegBitNames(NumSysRegBitNames + 6) = "c1ch1"
SysRegBitNames(NumSysRegBitNames + 7) = "c1ch0"
NumSysRegBitNames = NumSysRegBitNames + 7
' CM2CON0 0x108
SysRegBitNames(NumSysRegBitNames + 1) = "c2On"
SysRegBitNames(NumSysRegBitNames + 2) = "c2out"
SysRegBitNames(NumSysRegBitNames + 3) = "c2oe"
SysRegBitNames(NumSysRegBitNames + 4) = "c2pol"
SysRegBitNames(NumSysRegBitNames + 5) = "c2R"
SysRegBitNames(NumSysRegBitNames + 6) = "c2ch1"
SysRegBitNames(NumSysRegBitNames + 7) = "c2ch0"
NumSysRegBitNames = NumSysRegBitNames + 7
' CM2CON1 0x109
SysRegBitNames(NumSysRegBitNames + 1) = "mc1out"
SysRegBitNames(NumSysRegBitNames + 2) = "mc2out"
SysRegBitNames(NumSysRegBitNames + 3) = "c1Rsel"
SysRegBitNames(NumSysRegBitNames + 4) = "c2Rsel"
SysRegBitNames(NumSysRegBitNames + 5) = "t1gss"
SysRegBitNames(NumSysRegBitNames + 6) = "c2Sync"
NumSysRegBitNames = NumSysRegBitNames + 6
' EEDAT 0x10C
SysRegBitNames(NumSysRegBitNames + 1) = "eedat7"
SysRegBitNames(NumSysRegBitNames + 2) = "eedat6"
SysRegBitNames(NumSysRegBitNames + 3) = "eedat5"
SysRegBitNames(NumSysRegBitNames + 4) = "eedat4"
SysRegBitNames(NumSysRegBitNames + 5) = "eedat3"
SysRegBitNames(NumSysRegBitNames + 6) = "eedat2"
SysRegBitNames(NumSysRegBitNames + 7) = "eedat1"
SysRegBitNames(NumSysRegBitNames + 8) = "eedat0"
NumSysRegBitNames = NumSysRegBitNames + 8
' EEADR 0x10D
SysRegBitNames(NumSysRegBitNames + 1) = "eeadr7"
SysRegBitNames(NumSysRegBitNames + 2) = "eeadr6"
SysRegBitNames(NumSysRegBitNames + 3) = "eeadr5"
SysRegBitNames(NumSysRegBitNames + 4) = "eeadr4"
SysRegBitNames(NumSysRegBitNames + 5) = "eeadr3"
SysRegBitNames(NumSysRegBitNames + 6) = "eeadr2"
SysRegBitNames(NumSysRegBitNames + 7) = "eeadr1"
SysRegBitNames(NumSysRegBitNames + 8) = "eeadr0"
NumSysRegBitNames = NumSysRegBitNames + 8
' EEDATH 0x10E
SysRegBitNames(NumSysRegBitNames + 1) = "eedath5"
SysRegBitNames(NumSysRegBitNames + 2) = "eedath4"
SysRegBitNames(NumSysRegBitNames + 3) = "eedath3"
SysRegBitNames(NumSysRegBitNames + 4) = "eedath2"
SysRegBitNames(NumSysRegBitNames + 5) = "eedath1"
SysRegBitNames(NumSysRegBitNames + 6) = "eedath0"
NumSysRegBitNames = NumSysRegBitNames + 6
' EEADRH 0x10F
SysRegBitNames(NumSysRegBitNames + 1) = "eeadrh4"
SysRegBitNames(NumSysRegBitNames + 2) = "eeadrh3"
SysRegBitNames(NumSysRegBitNames + 3) = "eeadrh2"
SysRegBitNames(NumSysRegBitNames + 4) = "eeadrh1"
```

```
    SysRegBitNames(NumSysRegBitNames + 5) = "eeadrh0"
    NumSysRegBitNames = NumSysRegBitNames + 5
    ' SRCON 0x185
    SysRegBitNames(NumSysRegBitNames + 1) = "sr1"
    SysRegBitNames(NumSysRegBitNames + 2) = "sr0"
    SysRegBitNames(NumSysRegBitNames + 3) = "c1Sen"
    SysRegBitNames(NumSysRegBitNames + 4) = "c2Ren"
    SysRegBitNames(NumSysRegBitNames + 5) = "pulss"
    SysRegBitNames(NumSysRegBitNames + 6) = "pulsr"
    SysRegBitNames(NumSysRegBitNames + 7) = "fvren"
    NumSysRegBitNames = NumSysRegBitNames + 7
    ' BAUDCTL 0x187
    SysRegBitNames(NumSysRegBitNames + 1) = "abdovf"
    SysRegBitNames(NumSysRegBitNames + 2) = "rcidl"
    SysRegBitNames(NumSysRegBitNames + 3) = "sckp"
    SysRegBitNames(NumSysRegBitNames + 4) = "brg16"
    SysRegBitNames(NumSysRegBitNames + 5) = "wue"
    SysRegBitNames(NumSysRegBitNames + 6) = "adben"
    NumSysRegBitNames = NumSysRegBitNames + 6
    ' ANSEL 0x188
    SysRegBitNames(NumSysRegBitNames + 1) = "ans7"
    SysRegBitNames(NumSysRegBitNames + 2) = "ans6"
    SysRegBitNames(NumSysRegBitNames + 3) = "ans5"
    SysRegBitNames(NumSysRegBitNames + 4) = "ans4"
    SysRegBitNames(NumSysRegBitNames + 5) = "ans3"
    SysRegBitNames(NumSysRegBitNames + 6) = "ans2"
    SysRegBitNames(NumSysRegBitNames + 7) = "ans1"
    SysRegBitNames(NumSysRegBitNames + 8) = "ans0"
    NumSysRegBitNames = NumSysRegBitNames + 8
    ' ANSELH 0x189
    SysRegBitNames(NumSysRegBitNames + 1) = "ans13"
    SysRegBitNames(NumSysRegBitNames + 2) = "ans12"
    SysRegBitNames(NumSysRegBitNames + 3) = "ans11"
    SysRegBitNames(NumSysRegBitNames + 4) = "ans10"
    SysRegBitNames(NumSysRegBitNames + 5) = "ans9"
    SysRegBitNames(NumSysRegBitNames + 6) = "ans8"
    NumSysRegBitNames = NumSysRegBitNames + 6
    ' EECON1 0x18C
    SysRegBitNames(NumSysRegBitNames + 1) = "eepgd"
    SysRegBitNames(NumSysRegBitNames + 2) = "wrerr"
    SysRegBitNames(NumSysRegBitNames + 3) = "wren"
    SysRegBitNames(NumSysRegBitNames + 4) = "wr"
    SysRegBitNames(NumSysRegBitNames + 5) = "rd"
    NumSysRegBitNames = NumSysRegBitNames + 5
End Sub

Public Sub DefineAllOpCodeNames()
    OpCodeNames(1) = "addwf"
    OpCodeNames(2) = "andwf"
    OpCodeNames(3) = "clrf"
    OpCodeNames(4) = "clrw"
    OpCodeNames(5) = "comf"
    OpCodeNames(6) = "decf"
    OpCodeNames(7) = "decfsz"
    OpCodeNames(8) = "incf"
    OpCodeNames(9) = "incfsz"
    OpCodeNames(10) = "iorwf"
    OpCodeNames(11) = "movf"
```

```
    OpCodeNames(12) = "movwf"
    OpCodeNames(13) = "nop"
    OpCodeNames(14) = "rlf"
    OpCodeNames(15) = "rrf"
    OpCodeNames(16) = "subwf"
    OpCodeNames(17) = "swapf"
    OpCodeNames(18) = "xorwf"
    OpCodeNames(19) = "bcf"
    OpCodeNames(20) = "bsf"
    OpCodeNames(21) = "btfsc"
    OpCodeNames(22) = "btfss"
    OpCodeNames(23) = "addlw"
    OpCodeNames(24) = "andlw"
    OpCodeNames(25) = "call"
    OpCodeNames(26) = "clrwdt"
    OpCodeNames(27) = "goto"
    OpCodeNames(28) = "iorlw"
    OpCodeNames(29) = "movlw"
    OpCodeNames(30) = "retfie"
    OpCodeNames(31) = "retlw"
    OpCodeNames(32) = "return"
    OpCodeNames(33) = "sleep"
    OpCodeNames(34) = "sublw"
    OpCodeNames(35) = "xorlw"
    NumOpCodeNames = 35
End Sub

Public Sub DefineAllDestinationNames()
    DestNames(1) = "f"
    DestNames(2) = "w"
    NumDestNames = 2
End Sub

Public Sub DefineAllCompilerOperatorNames()
    CompOpNames(1) = "#include"
    CompOpNames(2) = "processor"
    CompOpNames(3) = "__CONFIG"
    CompOpNames(4) = "equ"
    CompOpNames(5) = "org"
    CompOpNames(6) = "END"
    NumCompOpNames = 6
End Sub

Public Sub DefineAllCompilerNounNames()
    CompNounNames(1) = """p16F882.inc"""
    CompNounNames(2) = "16F882"
    CompNounNames(3) = "_CONFIG1"
    CompNounNames(4) = "_CONFIG2"
    NumCompNounNames = 4
End Sub

Public Sub SearchForAllUserLabelNames()
    Dim lLabel As String
    ' Initialize the number of names
    NumUserLabelNames = 0
    ' Open the Assembly file
    FileReader = My.Computer.FileSystem.OpenTextFileReader(MainDir & ASMFileName)
    ' Main loop
```

```
Do While (Not FileReader.EndOfStream)
    ' Read the next line in the Assembly file
    LineAsRead = FileReader.ReadLine
    ' Process the line
    Do
        ' Check if the line is null
        If (Len(LineAsRead) = 0) Then
            ' The line is null, so ignore it
            Exit Do
        End If
        ' Check to see if the first character in the line is a blank
        If (Strings.Left(LineAsRead, 1) = " ") Then
            ' The first character is a blank, so ignore this line
            Exit Do
        End If
        ' Check to see if the first charcater is a semi-colon
        If (Strings.Left(LineAsRead, 1) = ";") Then
            ' The first character is a semi-colon, so ignore this line
            Exit Do
        End If
        ' Check to see if the first character in the line is ASCII 9 (Tab)
        If (Asc(Strings.Left(LineAsRead, 1)) = 9) Then
            ' The first character is a Tab, so ignore this line
            Exit Do
        End If
        ' Parse out the field to the first blank character
        lLabel = ""
        For I As Int32 = 1 To Len(LineAsRead) Step 1
            ' Check if the next character is a blank
            If (Strings.Mid(LineAsRead, I, 1) = " ") Then
                ' A blank has been encountered
                lLabel = Strings.Left(LineAsRead, I - 1)
                Exit For
            End If
            ' Check if the next character is a Tab
            If (Asc(Strings.Mid(LineAsRead, I, 1)) = 9) Then
                ' A Tab has been encountered
                lLabel = Strings.Left(LineAsRead, I - 1)
                Exit For
            End If
        Next I
        ' In some cases, the entire line will be a label
        If (lLabel = "") Then
            lLabel = LineAsRead
        End If
        ' Now that a label has been extracted, let's check it
        ' Test #1: Is the label a system register?
        For I As Int32 = 1 To NumSysRegNames Step 1
            If (lLabel = SysRegNames(I)) Then
                ' The label is a system register, so ignore it
                Exit Do
            End If
        Next I
        ' Test #2: Is the label a system register's bit name
        For I As Int32 = 1 To NumSysRegBitNames Step 1
            If (lLabel = SysRegBitNames(I)) Then
                ' The label is a system register's bit name, so ignore it
                Exit Do
```

```
                    End If
                Next I
                ' Test #3: Is the label a destination name
                For I As Int32 = 1 To NumDestNames Step 1
                    If (lLabel = DestNames(I)) Then
                        ' The label is a destination name, so ignore it
                        Exit Do
                    End If
                Next I
                ' The label must be a User label, so add it to the list
                NumUserLabelNames = NumUserLabelNames + 1
                UserLabelNames(NumUserLabelNames) = lLabel
                Exit Do
            Loop
        Loop
        ' Close the Assembly file
        FileReader.Close()
        FileReader.Dispose()
    End Sub

    Public Sub DefineAllTabStops()
        TabStops(1) = 8
        TabStops(2) = 16
        TabStops(3) = 24
        TabStops(4) = 32
        TabStops(5) = 40
        TabStops(6) = 48
        TabStops(7) = 56
        TabStops(8) = 64
        TabStops(9) = 72
        TabStops(10) = 80
        TabStops(11) = 88
        TabStops(12) = 96
        TabStops(13) = 104
        NumTabStops = 13
    End Sub

End Module
```