barcode image preprocessing filters in VBScript with ByteScout BarCode Reader SDK

barcode image preprocessing filters in VBScript

Every ByteScout tool contains example VBScript source codes that you can find here or in the folder with installed ByteScout product. ByteScout BarCode Reader SDK was made to help with barcode image preprocessing filters in VBScript. ByteScout BarCode Reader SDK is the barcode decoder with support for code 39, code 128, QR Code, Datamatrix, GS1, PDF417 and all other popular barcodes. Can read barcodes from images, pdf, tiff documents and live web camera. Supports noisy and damaged documents, can split and merge pdf and tiff documents based on barcodes. Can export barcode decoder results to XML, JSON, CSV and into custom data structures.

VBScript code snippet like this for ByteScout BarCode Reader SDK works best when you need to quickly implement barcode image preprocessing filters in your VBScript application. This VBScript sample code should be copied and pasted into your application's code editor. Then just compile and run it to see how it works. Enjoy writing a code with ready-to-use sample VBScript codes to add barcode image preprocessing filters functions using ByteScout BarCode Reader SDK in VBScript.

Trial version can be obtained from our website for free. It includes this and other source code samples for VBScript.

VBScript - TestBarcodeReading.vbs

```
' This exmaple demonstrates the use of image filters to improve the decoding or
speed.

Dim result

Set reader = CreateObject("Bytescout.BarCodeReader.Reader")
reader.RegistrationName = "demo"
reader.RegistrationKey = "demo"

' Set barcode type to find
reader.BarcodeTypesToFind.Code128 = True


' WORKING WITH LOW CONTRAST BARCODE IMAGES

' Add the contrast adjustment for the low contrast image
reader.ImagePreprocessingFilters.AddContrast(40)

result = result & "Image ""low-contrast-barcode.png""" & vbCRLF

reader.ReadFromFile "low-contrast-barcode.png"
```

```
If reader.FoundCount = 0 Then
        result = result & "No barcode found!" & vbCRLF
Else
        For i = 0 To reader.FoundCount - 1
        result = result & "Found barcode with type " &
CStr(reader.GetFoundBarcodeType(i)) & " and value """ &
reader.GetFoundBarcodeValue(i) & """" & vbCRLF
    Next
End If

reader.ImagePreprocessingFilters.Clear()
result = result & vbCRLF


' WORKING WITH NOISY BARCODE IMAGES

' Add the median filter to lower the noise
reader.ImagePreprocessingFilters.AddMedian()

result = result & "Image ""noisy-barcode.png""" & vbCRLF

reader.ReadFromFile "noisy-barcode.png"

If reader.FoundCount = 0 Then
        result = result & "No barcode found!" & vbCRLF
Else
        For i = 0 To reader.FoundCount - 1
        result = result & "Found barcode with type " &
CStr(reader.GetFoundBarcodeType(i)) & " and value """ &
reader.GetFoundBarcodeValue(i) & """" & vbCRLF
    Next
End If

reader.ImagePreprocessingFilters.Clear()
result = result & vbCRLF


' WORKING WITH DENSE AND ILLEGIBLE BARCODES

' Add the scale filter to enlarge the barcode to make gaps between bars more
distinguishable
reader.ImagePreprocessingFilters.AddScale_2(2) ' enlarge twice

result = result & "Image ""too-dense-barcode.png""" & vbCRLF

reader.ReadFromFile "too-dense-barcode.png"

If reader.FoundCount = 0 Then
        result = result & "No barcode found!" & vbCRLF
Else
        For i = 0 To reader.FoundCount - 1
        result = result & "Found barcode with type " &
CStr(reader.GetFoundBarcodeType(i)) & " and value """ &
reader.GetFoundBarcodeValue(i) & """" & vbCRLF
    Next
End If

reader.ImagePreprocessingFilters.Clear()
result = result & vbCRLF
```

```
' REMOVE EMPTY MARGINS FROM IMAGE TO SPEED UP THE PROCESSING

' Add the crop filter to cut off empty margins from the image.
' This will not improve the recognition quality but may speed up the processing
' if you enabled multiple barcode types to search.
reader.ImagePreprocessingFilters.AddCropDark()

result = result & "Image ""barcode-with-large-margins.png""" & vbCRLF

reader.ReadFromFile "barcode-with-large-margins.png"

If reader.FoundCount = 0 Then
        result = result & "No barcode found!" & vbCRLF
Else
        For i = 0 To reader.FoundCount - 1
        result = result & "Found barcode with type " &
CStr(reader.GetFoundBarcodeType(i)) & " and value """ &
reader.GetFoundBarcodeValue(i) & """" & vbCRLF
    Next
End If

reader.ImagePreprocessingFilters.Clear()
result = result & vbCRLF


Msgbox result


Set reader = Nothing
```
