

Documentation

INBarcodeOCR.DLL

V1.7.11



Inspirant
Volker Schmid
Roemerstrasse 39
D-78183 Huefingen
Germany

Table Of Content

Introduction.....	3
Company and contact information.....	4
Installation and redistribution.....	4
Licencing.....	5
Evaluation.....	5
Development	5
Integration (runtime-licence).....	5
Integration (runtime-free).....	5
Integration (server-side).....	6
Shareware and demo-versions.....	6
Purchase and pricing.....	6
Changes for your code.....	7
Important changes from V1.6 to V1.6.2.....	7
Important changes from V1.6.2 to V1.7.1.....	7
Important changes from V1.7.x to V1.7.6.....	7
Important changes from V1.7.6 to V1.7.7.....	7
Important changes from V1.7.7 to V1.7.8.....	7
Important changes from V1.7.8 to V1.7.9.....	7
API functions.....	8
Finding and decoding barcodes.....	8
Decoding given single barcodes.....	9
Retrieving results.....	10
PDF related functions.....	11
Recognition enhancements and debug functionality.....	13
Other functions.....	14
Disable features.....	15
Recognition Problems?.....	16
Best Practices.....	18
Supported Charsets.....	19
Supported Code128 / EAN128 charsets.....	19
Supported Code39 charsets.....	20
Supported 2/5, 2/5 interleaved and EAN/UPC charsets.....	20
Declarations and calling examples.....	21

Introduction

Using the INBarcodeOCR you are able to recognize and decode barcodes in an image.

Features:

- recognizes **Code39** (with optional calculation an revision of checksum)
- recognizes **Code128** (including calculation an revision of checksum)
- recognizes **EAN128** (including calculation an revision of checksum)
- recognizes **EAN13** (including calculation an revision of checksum)
- recognizes **EAN8** (including calculation an revision of checksum)
- recognizes **UPC-A** (including calculation an revision of checksum)
- recognizes **2/5** and **2/5 interleaved**
- able to find all supported barcodes inside documents
- supports **BMP**, **TIF**, **JPG** and **PNG** image formats
- supports **PDF** file format
- supports „PDF to image“ and „image to PDF“ conversion and „PDF extraction“
- can **directly use handle** of existing bitmaps in memory (bitmal handle hDC)
- returns position, type, orientation and content of every supported barcode found
- methods to decode single barcode images
- recognizes barcodes rotated by 180°, horizontal and vertical
- runs on Windows 2000/2000 Server/XP/2003 Server/Vista and Windows 7
- ActiveX/COM wrapper available

Restrictions:

- the maximum allowed aberration from horizontal or vertical is +6° / -6°
- every given bitmap must be smaller 8192x8192 pixel
- images must be at least 200DPI
- depending on the size and quality of the barcodes, 200 to 400 DPI is recommended
- if you use multipage TIF files, the library uses only the first page
- The library offers no support for JPG images in JPEG2000 ob JBIG2 format (both directly and embedded in PDF)



Important information about the DEMO version:

The demo version, downloadable from the homepage of inspirant.de, contains not all components needed for the PDF functionalities.

So, the demo version is not able to process PDF files. You will receive those components after registration of at least one developer licence (one single runtime licence).

If you have good reasons to have PDF support and not to buy a single licence, please contact us for individual conditions.

Company and contact information

company

Inspirant is a company of 5 staff members. We have a rich fund of external cooperators which are available for the need of special knowledge or skills. In case of a shortage in development power we have a great assortment in available freelancers. We mainly develop for Windows systems using Visual Basic (5/6 and .NET) and PureBasic (Windows/Linux) to reduce time and cost for our customers. Some routines are also programmed in C, C++ or other languages if there is the need for. In the meantime we are also very familiar with linux-development.

special subjects

Our special subjects are the complete environment of document management and the solution of tricky problems. We are a excellent contact for document management, archieving, migration, workflow or storage as well as computer-aided production data acquisition, billing or warehousing. If you are in need of individual development for your special cases you are on the right homepage.

We are always trying to keep costs as low as possible. We also looking for open source and linux solutions whenever it is possible and better.

contact information

Inspirant EDV-Software-Consulting
Volker Schmid
Roemerstrasse 39
78173 Huefingen
Germany

Phone: 0049 (0) 771 89784 235
e-mail: info@inspirant.de
web: www.inspirant.de

Installation and redistribution

To **install or redistribute the library**, you need to copy the INBarcodeOCR.dll to the system-directory. In most cases this is `c:\winnt\system32\` or `c:\windows\system32\` directory. At least, the same directory than your executable may work, too.

To **add the PDF support**, the INBarcodePDF.dll has to be installed into the same directory than the INBarcodeOCR.dll. If you don't need the PDF support, you can omit the INBarcodePDF.dll. Please use the `PDFEnabled()` function to check PDF availability.



Information: The INBarcodePDF.dll is not a part of the demo version downloadable at the inspirant.de homepage. You need at least one licence to receive the DLL.

Licencing

This library is sold by runtime licences. There is also an option to buy a runtime-free licence. Choose this model if you decide to sell your product in higher numbers or you want to sell a server-side application with no user-oriented licence.

Evaluation

You can use this library without calling the RegisterBarcodeDLL() function. In this case you are free to test the capabilities of the library for your needs. You will see an evaluation-reminder at some times calling the GetBarcodesResult() function until you called RegisterBarcodeDLL() with the correct name and password.

Development

You need only one single runtime-licence for development inside your company. This licence applies for all computers inside your company which are used for development but not for productivity.

Integration (runtime-licence)

For each computer your customer likes to install this library, you have to buy one runtime-licence. So you have to buy one licence for each licence your customer buys.

example: You sell 8 program- or user-licences. You have to buy 8 runtime-licences of INBarcodeOCR.

special case server-side applications

Is your product meant for server-side installation, you may have to calculate the number of licences needed with the average amount of accessing users.

example 1: You sell a server-side application but charge your customer for 8 user licences, you have to buy 8 licences of INBarcodeOCR.

example 2: You sell or use the application server-side without consideration of the amount of users using this product (for example, for automated importing or data extraction). Here you have to buy the runtime-free licence of INBarcodeOCR for each server.

Integration (runtime-free)

The runtime-free licence allows you to integrate INBarcodeOCR into your product and sell it without the need of buying further runtime-licences of INBarcodeOCR. No matter if you sell 10 or 10000 licences of your software, the runtime-free licence covers all.

Important: This does not apply to server-side usage without user-counting (like batch-jobs). In this case, the runtime-free licence needs to get bought for each server which uses INBarcodeOCR in such a manner.

Integration (server-side)

For server-side integration of INBarcodeOCR, you need to buy a runtime-free licence for each server that will run INBarcodeOCR.

example 1: You sell a product that gets installed on 10 servers of your customers without counting the number of users. You need to buy 10 runtime-free licences.

example 2: You like to install an application that uses INBarcodeOCR locally on two servers of your own company without counting the number of users. You need to buy two runtime-free licences.

In case of larger amounts or educational use, please feel free to contact us for an individual offer.

Shareware and demo-versions

If you distribute your product as limited shareware or demo-version (restricted functionality or time-limited), you don't need runtime-licences for this versions. In the moment the customer buys one ore more licences of your software, you have to order the corresponding runtime-licenses for INBarcodeOCR, too.

Purchase and pricing

The actual prices are available on the homepage you downloaded this library. There you can find the purchasing information, too.

Changes for your code

This chapter describes new features and changes that may affect your application after an upgrade. Please revisit this chapter after each update to a newer version of this library.

Important changes from V1.6 to V1.6.2

There is a new function available: [UseFineSearch\(\)](#). Simply add the declaration to be able to use this feature.

Important changes from V1.6.2 to V1.7.1

Recognition speed is up to 25% faster than V1.6.x.

[GetBarcodesResult\(\)](#) now additionally returns the PageNumber. Using normal bitmaps, the page-number is always 1. Using PDF images, the number corresponds to the page-number the barcode was found.

The new option [ReturnAllCandidates\(\)](#) allows you to return barcodes that even have not been recognized correctly, but hardly seem to be a barcode (for example if checksum tests failed). Missed characters are displayed as „?“.

New function [PDFExtract\(\)](#) to extract/split portions of a PDF file into a new generated PDF file.

Important changes from V1.7.x to V1.7.6

Recognition speed for PDF files is up to 30% faster than in previous versions.

Important changes from V1.7.6 to V1.7.7

New function [PDFGetPageCount\(\)](#) to retrieve the number of pages inside a PDF document.

Important changes from V1.7.7 to V1.7.8

PDF-functions do need much less memory.
Reduced number of „false positives“ for 2/5 and 2/5 interleaved.

Important changes from V1.7.8 to V1.7.9

New option [UseDistortionCheck\(\)](#). If this option is activated, the library try's to read distorted codes, too. This will reduce processing speed, but may enhance recognition quality on heavy distorted barcodes.

API functions

Finding and decoding barcodes

This routines are meant to find and recognize barcodes in a document image. Use this functions if you don't have an isolated barcode image. The complete image is scanned for possible barcodes and then every one of this candidates is checked against the possible barcode-formats.



FindBarcodesFile(strFilename)

Retrieves the barcodes in a given image-file (BMP, JPG, PNG, TIF, PDF). If a PDF image is used, every page will get extracted and scanned for available barcodes.

Return values (signed long):

- Returns the number of barcodes found.
- Returns -1 for an error (file not found)

FindBarcodesClipboard()

Retrieves the barcodes in a given image in clipboard.

Return values (signed long):

- Returns the number of barcodes found.
- Returns -1 for an error (no image in clipboard) (long)



Info: This is not working for PDF-files inside the clipboard! Use FindBarcodesFile() instead.

FindBarcodesHDC (lngHandle)

new since V1.5

Retrieves the barcodes in a given image. Provide an existing bitmap-handle (hDC) as parameter to define the image to use. This may be a very good way to process images that are already loaded by your application. Don't forget to release your hDC after processing inside your application.

Return values (signed long):

- Returns the number of barcodes found.
- Returns -1 for an error (long)

Decoding given single barcodes

Use this routines if you have images of isolated barcodes and you don't have the need for finding barcodes in a document. Isolated barcodes are not allowed to contain other pixels than the ones of the barcode (especially to the left and the right).



Use `GetBarcodesResult()` to retrieve the result of this functions.

DecodeBarcodeFile (strFilename)

Decodes the barcode in a given image-file (BMP, JPG, PNG, TIF).

Return values (signed long):

- Returns the number of barcodes found (always 0 or 1).
- Returns -1 for an error (file not found) (long)

DecodeBarcodeClipboard ()

Decodes the barcode of the image in the clipboard.

Return values (signed long):

- Returns the number of barcodes found (always 0 or 1).
- Returns -1 for an error (no image in clipboard) (long)



Info: This is not working for PDF-files inside the clipboard! Use `FindBarcodesFile()` instead.



Important: `DecodeBarcodeFile()` and `DecodeBarcodeClipboard()` do not return 2 or 3 for orientation. The only orientation information available is 0 (horizontal) or 1 (vertical).

Retrieving results

This functions are for retrieving the results of a foregoing barcode-recognition. Please call this functions to receive the results of GetBarcodesFile(), GetBarcodesClipboard(), DecodeBarcodeFile() and DecodeBarcodeClipboard().

GetBarcodesResult ()

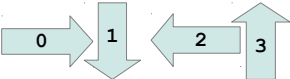
Returns the result of the barcode OCR.

Returns the following string (long pointer to string):

X TAB Y TAB Width TAB Height TAB Orientation TAB Codetype TAB Code TAB Page CR
X TAB Y TAB Width TAB Height TAB Orientation TAB Codetype TAB Code TAB Page CR

...

(TAB = ASCII 9, CR = ASCII 13)

Value	Position	Description
X	1	The upper left position of the rectangle used to identify this barcode.
Y	2	The upper left position of the rectangle used to identify this barcode.
Width	3	The width of the rectangle used to identify this barcode.
Height	4	The height of the rectangle used to identify this barcode.
Orientation	5	The orientation of the barcode. 0 = horizontal 2 = horizontal/rotated reading directions:  1 = vertical 3 = vertical/rotated
Codetype	6	The type of this recognized barcode. Possible values are: CODE39 CODE128 EAN13 EAN8 EAN128 UPC-A 2/5i 2/5
Code	7	The recognized barcode-value.
Page	8	The page number, on which page the barcode has been found (currently only PDF)



NOTE 1:

Retrieving a single barcode using DecodeBarcodeFile() and DecodeBarcodeClipboard() results always in one single line without the CR character or an empty string (in case of no recognized barcode).



NOTE 2:

The coordinates returned by this function (X, Y, Width, Height) are the coordinates of the image used to recognize the barcode. There is no guarantee that this rectangle covers the complete barcode! In many cases this rectangle covers only a part of the complete barcode. This is meant only for information to the developer in case he wishes to decide something based on the position.



NOTE 3:

The orientation is clockwise enumerated. This result depends on the direction the program recognized the code correctly.

PDF related functions

PDFSetResolution (lngDPI)

For retrieving barcodes in PDF images, the first side of the PDF image is rendered into a bitmap. The default resolution of this rendering process is 240 DPI. If you know the resolution of the barcodes inside the PDF file, you can set this value for this function, too.

The correct value may increase the recognition quality. Higher resolutions will increase the time needed for the recognition functions.

PDFEnabled()

Returns information about the PDF availability.

Return values (long):

Returns 0 if PDF support is not given

Returns 1 if PDF support is available

If PDF support is not available, please ensure that the INBarcodePDF.dll is located in the same directory than the INBarcodeOCR.dll.

PDFGetImage (strPDFFilename, lngDPI, strFormat, lngStartPage, lngEndPage)

changed since V1.5

Extract the image of the first page of a PDF file.

The DPI value defines the rendering resolution. Please use values between 50 and 800 DPI. If you set to 0, the current resolution is used (default is 240 DPI but may be changed using SetPDFResolution() before).

Using strFormat you can define the output format. Use "BMP", "JPG" or "PNG" as parameters.

If the PDF file has more than one page, you can use lngStartPage and lngEndPage to define which pages you like to extract. For extracting only the first page use 1 for lngStartPage and 1 for lngEndPage.

The page number will be appended to the end of the filename before the extension. For example, if strPDFFileName is "c:\output.jpg" and page 10 is rendered the image will be stored in a file called "output10.jpg".

Return values (long pointer to string):

The function returns the complete filenames of the rendered images divided using CR (CarriageReturn or ASCII 13).

The output path will be in the current temp directory.

The function returns an empty string if an error occurred.

PDFImageToPDF(strImage, strDestination, strSize)

Converts the given image(s) (strImage) into a PDF file (strDestination). This function is able to generate multipage PDF files if the source image is a multipage TIF file or more than one image is given.

The desired document size and format is defined using the strSize parameter. The size of the paper can be one of the following:

A0 to A10	B0 to B10	ISOB0 to ISOB10	C0 to C7
DL	Letter	Legal	Statement
Tabloid	Ledger	Executive	Folio.

You can make a landscape page by adding the word „landscape“ after the format, for example „A4 landscape“ or „Letter landscape“.

You can merge multiple images into one pdf, if you join multiple filenames inside strImage. Please use tabulator char (ASCII 9) to divide multiple image filenames.

Example to join three picture files into one PDF image:

```
strImage = "c:\header.jpg" + chr(9) + "c:\image2.tif" + chr(9) + "c:\footer.bmp"
strDestination = "c:\output.pdf"
PDFImageToPDF(strImage, strDestination, "A4")
```

Return values (signed long):

The function returns the number of generated pages.

Negative numbers are reserved for errors. The following error-codes are supported:

- 1 = error while setting the size (check strSize)
- 2 = error while opening the image(s) (check strImage)
- 3 = internal error while creating the PDF document
- 4 = error while saving the generated PDF document (check strDestination. Destination locked? Already open?)

PDFExtract(strOriginal, strDestination, strRange)

Extracts portions of a given PDF file (strOriginal) into a new PDF file (strDestination). The strRange parameter defines the pages to extract. This function is often used to split PDF images into multiple separate documents.

strRange example: **"10,15,18-20,25-35"** extracts the pages 10, 15, 18 to 20 and 25 to 35.

Return values (signed long):

- Returns 0 in case of success
- Returns -1 in case of failure

PDFGetPageCount(strPDFFilename)

new since V1.7.7

Returns the number of pages inside the given PDF document.

Return values (long):

- Returns 0 in case of failure
- Returns number of pages in case of success

Recognition enhancements and debug functionality

This functions are needed for special cases with recognition problems.

UseIncreasedSensitivity (lngFlag)

new since V1.6.1

Use increased sensitivity for reading barcodes in colored or grayscale images. On B/W images this option is automatically ignored. Set this flag to 1 to enable and 0 to disable this feature.

Important: Enabling the UseIncreasedSensitivity-option will slowdown recognition noticeable, but can produce much more accurate results in color- or gray-scanned images with more than 4 bit color-depth.

UseFineSearch (lngFlag)

new since V1.6.2

Use this option for reading barcodes in small images with 150DPI and lower. Set this flag to 1 to enable and 0 to disable this option.

Important: Enabling the UseFineSearch-option will slowdown recognition noticeable, but can produce much more accurate results in small images with less than 150DPI. Those images need to be in excelent quality (like barcodes which are inserted directly into images instead of scanning). It has negative effects on images with more than 150 DPI!

UseDistortionCheck (lngFlag)

new since V1.7.9

Use this option for reading barcodes that are distorted or having many scattered pixels. Set this flag to 1 to enable and 0 to disable this option.

Important: Enabling the UseDistortionCheck-option will slowdown recognition noticeable.

ReturnAllCandidates (lngFlag)

new since V1.7.0

This option allows you to return barcode candidates, too. Set this flag to 1 to enable and 0 to disable this option.

Important: Use this option only for debugging purposes, as there may be „?“ in the returned code. A ? represents a char that has not been recognized.
This option disables all checksum-tests, too!

Other functions

GetBarcodeVersionInfo()

Returns the version of this DLL.

Return values (long pointer to string):

Returns version information (string)

RegisterBarcodeDLL(strName, strPassword)

Registers this DLL using a name and a password you got after purchase. The *name* parameter is case-sensitive!

Return values (signed long):

Returns 0 for success (long)

Returns -1 for failure (long)



Important: You have to register this DLL each time you initialize it to be able to use it. You will receive the name and password after successfully purchasing a licence. Without registering, this library will show an evaluation-reminder at some times calling the GetBarcodesResult() function.

UseCode39Checksum(lngFlag)

Decide if the OCR should calculate and revise a CheckSum for code39 barcodes. Code39 defines no checksum by default but there is an option to calculate and encode this checksum. If your barcodes are Code39 and having a checksum you can enable this checkup with this function.

0 = don't calculate and revise Code39 checksum (default)

1 = calculate and revise Code39 checksum

Disable features

This section describes functions for disabling some features of the library. The reason may be speed improvements. Another reason may be the wish to receive only one barcode-type even though other barcodes exist on the documents. Especially the abandonment of checking for rotated barcodes may speed up things dramatically.

DisableEAN13 (lngFlag)

Set if the OCR should search and decode EAN13 and UPC-A barcodes

0 = Enable this barcode-type (default)

1 = Disable this barcode-type

DisableEAN8 (lngFlag)

Set if the OCR should search and decode EAN8 barcodes

0 = Enable this barcode-type (default)

1 = Disable this barcode-type

DisableCode39 (lngFlag)

Set if the OCR should search and decode Code39 barcodes

0 = Enable this barcode-type (default)

1 = Disable this barcode-type

DisableCode128 (lngFlag)

Set if the OCR should search and decode Code128 barcodes

0 = Enable this barcode-type (default)

1 = Disable this barcode-type

Disable2of5 (lngFlag)

Set if the OCR should search and decode 2/5 interleaved barcodes. This is very handy if you have a lot of false positives. The 2/5 code is very vulnerable to false positives and even ordinary text may be recognized as a correct barcode sometimes. If you don't need 2/5 in your application, it is good to deactivate this by default.

0 = Enable this barcode-type (default)

1 = Disable this barcode-type

DisableRotation (lngFlag)

changed since V1.5

Set if the OCR should search and decode rotated barcodes (by 90°, 180°, 270°)

0 = Enable this feature (default)

1 = Disable this feature (only left to right barcodes can be found)

Recognition Problems?

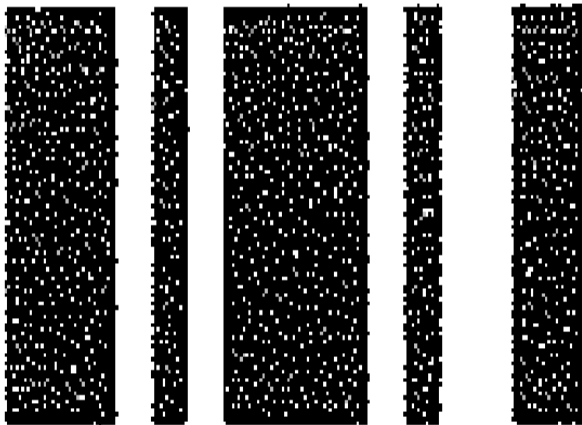
Most problems occur when

- barcode-type is not supported
- image quality is bad
- image resolution is too low (<200 DPI)
- less contrast (background colour?)
- barcodes are rotated more than 6°
- barcode is distorted
- the left or right whitespace is too small

Tips:

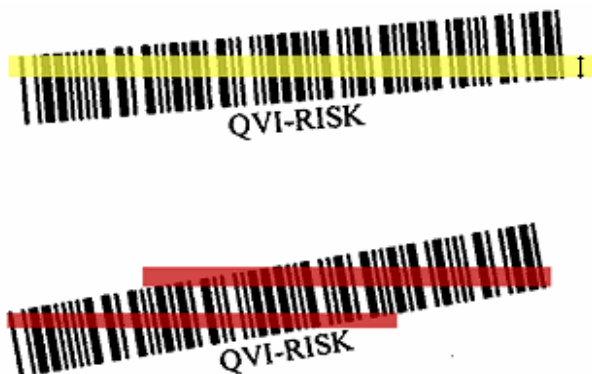
- try scanning using 300 DPI or even more
- experiment using brightness and contrast of the images (scanner interface)
- try B/W scanning with various thresholds
- remember, that only the first page is scanned inside of multipage TIF documents
- try `UseFineSearch()` or `UseDistortionCheck()`

Examples:



This is a section of a scanned barcode in high resolution. As you can see, the bars are containing very much noise.

As the library checks a barcode pixel by pixel, it is not really clear if the bar ends or if it is just a part of the noise. So, this barcode has only been readable using the activated `UseDistortionCheck`-option. To avoid such noisy barcodes, try to change the black/white threshold values of your scanning device and, maybe, reduce the scanning resolution.



The `INBarcodeOCR`-library works horizontal and vertical to find and read barcodes. As the complete barcode is needed to verify a checksum or find the end-code, rotation is a critical problem.

In this example you can see, how the usable part of a barcode decreases with rotation. The yellow portion is the only usable part. With even more rotation, the barcode is no longer readable (red).



Here you can see a barcode printed with an old or defect printer. There is some 'jitter' in the bars. If you look closely at the first bar, you will see that the width is varying. This example has been readable by `INBarcodeOCR` but more of this problems may cause failure of recognition.



Here is the contrast of the scanned image not very good. It has been readable, but a higher contrast would increase the liability very much.



This barcode has been processed by some image enhancing algorithms. It seems, that all lines are stronger. But the barcode itself does not profit of such enhancement as the bars no longer appear as filled. It has not been readable.

False Positives:

It sometimes may happen that INBarcodeOCR finds a barcode in documents which is not really a barcode. Especially 2/5 and 2/5 interleaved are very vulnerable of false positives, because some charsets and words are really like a barcode in some special cases. We recommend to use `Disable2of5()` in case you are shure you don't need to recognise this type of barcode.

If it is not possible to disable 2/5 recognition, please do a validity check of the recognized values. Mostly, the false positives are not longer than two or three characters (like 65 or 12).

Best Practices

To get best possible results, please take notice about this:

- **Use a barcode encoding that fits your needs.** Do not use EAN or Code128 if you don't need the chars available here. The best recognized barcode is 2/5i followed by Code39, Code128 and EAN codes.
- **Disable barcode-types that you are sure you do not need to recognize.** If you need to find only Code39 barcodes, use `DisableCode128()`, `Disable2of5()`, `DisableEAN13`, `DisableEAN8()` to enhance speed and reduce the risk of false positives.
- **Disable vertical barcode recognition, if you do not need to find vertical barcodes** (`DisableRotation()` function).
- Best results have been achieved using **scanning resolution from 240 to 400 DPI**. If your image has been generated by software (not scanned), please try to use the `UseFineSearch()` feature to advise the engine about lower resolution).
- If you are **processing colour-images** or grayscale-images with varying lighting conditions, please try the `UseIncreasedSensitivity()` function to activate a more adaptive threshold.
- If you are scanning B/W or grayscale, please **try to vary the scanner threshold** settings.
- Some scanners offering **image enhancements**. Some of them may be helpful (denoise, automatic threshold), others may create problems (too much sharpening, automatic deskew).
- If an image has not been able to get recognized, sometimes a simple rescan may help.
- If you are processing PDF images and you know the DPI settings of the containing barcode images, it is strongly recommended to use `SetPDFResolution()` with the same resolution. This will speed up PDF conversion and enhance the recognition quality.

Supported Charsets

Supported Code128 / EAN128 charsets

To use this kind of codes, a scan-resolution of 300DPI or higher is recommended.

Value	Code A	Code B	Code C
0	[Space]	[Space]	00
1	!	!	01
2	"	"	02
3	#	#	03
4	\$	\$	04
5	%	%	05
6	&	&	06
7	'	'	07
8	((08
9))	09
10	*	*	10
11	+	+	11
12	,	,	12
13	-	-	13
14	.	.	14
15	/	/	15
16	0	0	16
17	1	1	17
18	2	2	18
19	3	3	19
20	4	4	20
21	5	5	21
22	6	6	22
23	7	7	23
24	8	8	24
25	9	9	25
26	:	:	26
27	;	;	27
28	<	<	28
29	=	=	29
30	>	>	30
31	?	?	31
32	@	@	32
33	A	A	33
34	B	B	34
35	C	C	35
36	D	D	36
37	E	E	37
38	F	F	38
39	G	G	39
40	H	H	40
41	I	I	41
42	J	J	42
43	K	K	43
44	L	L	44
45	M	M	45
46	N	N	46
47	O	O	47
48	P	P	48
49	Q	Q	49
50	R	R	50

Value	Code A	Code B	Code C
51	S	S	51
52	T	T	52
53	U	U	53
54	V	v	54
55	W	W	55
56	X	X	56
57	Y	Y	57
58	Z	Z	58
59	[[59
60	\	\	60
61]]	61
62	^	^	62
63	_	_	63
64	<NUL>	<NUL>	64
65	<SOH>	a	65
66	<STX>	b	66
67	<ETX>	c	67
68	<EOT>	d	68
69	<ENQ>	e	69
70	<ACK>	f	70
71	<BEL>	g	71
72	<BS>	h	72
73	<HT>	i	73
74	<LF>	j	74
75	<VT>	k	75
76	<FF>	l	76
77	<CR>	m	77
78	<SO>	n	78
79	<SI>	o	79
80	<DLE>	p	80
81	<DC1>	q	81
82	<DC2>	r	82
83	<DC3>	s	83
84	<DC4>	t	84
85	<NAK>	u	85
86	<SYN>	v	86
87	<ETB>	w	87
88	<CAN>	x	88
89		y	89
90	<SUB>	z	90
91	<ESC>	{	91
92	<FS>		92
93	<GS>	}	93
94	<RS>	~	94
95	<US>		95
96	<FNC3>	<FNC3>	96
97	<FNC2>	<FNC2>	97
98	<SHIFT>	<SHIFT>	98
99	reserved	reserved	99
100	reserved	<FNC4>	reserved
101	<FNC4>	reserved	reserved
102	<FNC1>	<FNC1>	<FNC1>

Supported Code39 charsets

Value	Code	Value	Code
1	0	23	M
2	1	24	N
3	2	25	O
4	3	26	P
5	4	27	Q
6	5	28	R
7	6	29	S
8	7	30	T
9	8	31	U
10	9	32	V
11	A	33	W
12	B	34	X
13	C	35	Y
14	D	36	Z
15	E	37	-
16	F	38	.
17	G	39	[Space]
18	H	40	\$
19	I	41	/
20	J	42	+
21	K	43	%
22	L	44	*

To use this kind of code, a scan-resolution of 240DPI or higher is recommended.

Supported 2/5, 2/5 interleaved and EAN/UPC charsets

Value	Code
1	0
2	1
3	2
4	3
5	4
6	5
7	6
8	7
9	8
10	9

To use 2/5 and 2/5 interleaved codes, a scan-resolution of 200DPI or higher is recommended.
For EAN/UPC charsets, a scan-resolution of 300DPI is recommended.

Declarations and calling examples

Please refer to the *Declares* folder in your INBarcodeOCR ZIP file. There you can find declarations to some common programming languages.

Please remark, that the declarations in the declares folder may not contain declarations to all available methods of INBarcodeOCR. These are meant as examples for how to declare the functions described in this document.