Barcodes not recognized
What can you do?

A support for QS-Barcode SDK, QS-DocumentAssembler, bcTester, and QS-Beleg

QS QualitySoft GmbH
Tempowerkring 21a
21079 Hamburg
Germany

Tel: +49 (0) 40 790 100 40
Fax: +49 (0) 40 790 100 44
info@qualitysoft.de

Newest information in the internet:
www.qualitysoft.de

White Paper assembled by our Support Team

© 2008 QS QualitySoft GmbH

About barcodes in general

Barcodes serve for coding information. A linear barcode is a sequence of bars (dark lines) and spaces between the bars (white lines). A linear barcode always starts and ends with a bar, a bar is always followed by a space, and a space is always followed by a bar. The width of the bars and spaces are usually different, and the order of the bars and spaces along with their width carry the information encoded by the barcode.

Recording information using barcodes enables a significant decrease of work in comparison with the amount of work involved when data is entered manually. For example item numbers can be read by hand-held scanners instead of typing twelve numerics. Using barcodes is faster and more accurate as OCR (Optical Character Recognition).

There are many different barcode symbologies (codings) available, each of which uses different orders and widths. If a linear barcode symbology distinguishes two different widths for the bars and two different widths for the spaces (thin and narrow), the symbology is binary, otherwise the symbology is non-binary.

Each linear barcode starts and ends with a specific pattern, which provides type destination and ascertains the external integrity of bars.
To verify of data contents, checksums or checksum characters are recommended; for some barcode symbologies it is required.
Barcodes can differ in character set. Basic codes can scramble numerics others can also encode letters.

Linear Bar Codes

Code 2/5, content: 6 characters

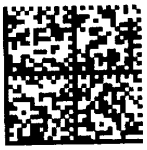
EAN 13, content: 13 characters

In addition to the usual linear barcode described above, the new 2D (two dimensional) barcodes are increasingly used. Their elements are arranged in a plain.

2D Barcodes

PDF417, Content: 165 characters

Data Matrix, 57 characters



QR Code, 25 characters

The system of 2D barcodes enables the compression of more contents on smaller spaces.

The most common 2D barcodes are Data Matrix, PDF 417, QR Code and Aztec Code. These barcodes can be created in many different sizes and can contain more than 2,000 characters, which can also be binary-coded.

Products for Barcode Recognition from Images

QS QualitySoft would like to introduce you to our set of products for barcode recognition. For further information please visit our website http://www.qualitysoft.de

To analyze barcodes you can use our **Barcode-Testapplication for free** http://www.bctester.de/index.php?id=67&L=1.

bcTester is an easy-to-use program for Windows. bcTester contains different examples of barcodes and a detailed help assistant with versatile information on barcodes. Using bcTester you can open an image or Adobe PDF document and test the barcode recognition. You can adapt the adjustments for the barcode recognition if required or you can use the automatic analysis.

Barcodes that are recognizable with bcTester, are also readable by our other products.

With **QS-DocumentAssembler**, we also provide a program that allows recognition of large quantities of barcodes automatically from image files or Adobe PDF documents (batch mode). At the same time, the image files can be indicated or can be structured to multipage documents by their barcode information.

The product **QS-Barcode SDK** (Software Development Kit) is for software engineers. With the SDK barcode recognition can be integrated easily into your own program.

On our website http://www.qualitysoft.de you can find a **free evaluation version** of **QS-DocumentAssembler** and **QS-Barcode SDK** for testing.

Note:
Please be aware that there is a demo mode. If bcTester reads your barcode well and a QS-Barcode Application or **QS-DocumentAssembler** reads a **“3” as a “1”, the demo mode** is activated. This is standard if no licence data file has been found.

Barcodes not recognized - What can you do?

Occasionally, a barcode is not immediately recognized by ”QS-Barcode“. This does not necessarily mean that the barcode cannot be recognized by ”QS-Barcode“. In most cases, the reasons for unsuccessful recognition are errors which can easily be corrected.

You should have your barcodes to be recognized available as an image file or in a Adobe PDF document. If you have problems loading the file into the bcTester program, you may have to choose another format for your image file.

In general the reasons for a non-successful barcode recognition can be divided into three main groups:

- **1. Errors in printing the barcode**
- **2. ”Inappropriate“ scanner settings**
- **3. Inappropriate parameters for recognition**

In the following chapters you will find hints to help you avoid the most frequently occuring errors.

We also give you information about the commands for “image preprocessing” which could turn bad barcode images into readables.

If this is not successful, please contact QualitySoft.

We will analyze your barcodes for free!

Just send your images and a short description of your problem to support@qualitysoft.de

For further information about our products, please visit our website

http://www.qualitysoft.de

1. Cause of Error: Errors in printing the barcodes

In practice, errors appear during producing and printing barcodes. As a result the barcodes cannot be recognized. These errors usually involve:

- different thickness of bars and gaps that normally should be the same (resolution too low, bad barcode-fonts or print driver errors)
- omitting the rest zone besides the barcode
- cut-off of several bars at the edges of the barcode

The two latter errors particularly appear while printing labels. Label printing is often very close to the edge of the label. These results in the light marging on both sides of the barcode or even several bars are missing. This can also be caused by some printing programs, if the field for the barcode is too small (and will be cut off).

If the barcode appears ok on the scanned image but cannot be read. This may happen because the start and stop characters were forgotten during printing. With many barcodes you can easily check by counting whether all the necessary bars are contained.

We will explain the general layout of a barcode using Code 39 as an example. In this code, 9 elements (5 bars and 4 gaps) form one character, three of the elements are thick.

Characters of Code 39 (only digits are shown, letters A-Z and the special characters $/+% can be encoded with Code 39 as well).

0	0	4	4	8	8
1	1	5	5	9	9
2	2	6	6	0	0
3	3	7	7	*	*

The * (star) character is used as a start and stop character. This means that the number 1234 is correctly encoded as *1234* and then consists of 6x5=30 bars. For clarification purposes, the following picture shows every single sign, the start and stop characters are red.

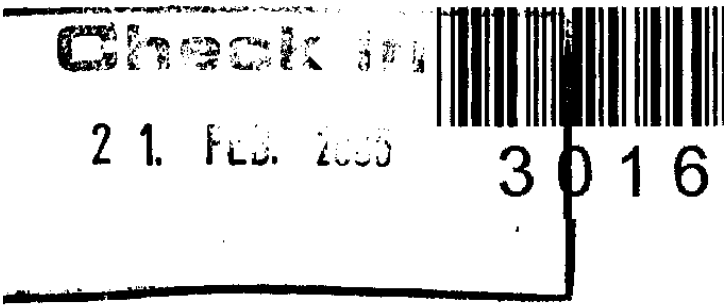
Without the gaps the code looks like this:

The start and stop characters of some major barcode types:

The start / stop characters must be used! Every type has specific characters which help to identify the barcode type. Calculating the number of bars a barcode contains depends on the type and the characters of the barcode. The formula is not always as easy as in Code 39, because there are certain types which contain compressions, checksums, control and escape characters.

If bars of the barcode are missing, reading of the barcode is impossible. This also occurs if the barcode produced does not fit into the targeted field and the barcode is cut off during printing.

You should not write or stamp on the barcode; otherwise proper reading could be prevented. This example shows how a check-in-stamp can cut off the barcode reading.



2. Cause of Error: ”Inappropriate“ scanner settings

Difficulties involving barcode reading often is due to errors that occurred from scanning. Often these barcode errors are already visible in the image.

Potential causes of failure can be:

- scan resolution too low
- scan too dark or too bright.

The following section outlines typical errors based on two examples. Here is an **ideal situation**:

123456

Code 39: 123456

This code only has 2 kinds of thicknesses, which means:

Bar thickness:

- all thin bars have the same width
- all thick bars have the same width
- the thick bars are three times as wide as the thin ones

Gap thickness:

- all thin gaps have the same width
- all thick gaps have the same width
- the thick gaps are three times as wide as the thin ones

Ratio bars/gaps:

- the thin bars have the same width as the thin gaps
- the thick bars have the same width as the thick gaps

EAN 13

This code has 4 kinds of thicknesses, which means:

Bar thickness:

- four different kinds of bar thicknesses are possible. They can be twice, three or four times as wide as the thinnest bar.

Gap thickness:

- four different kinds of gap thickness are possible. They can be twice, three or four times as wide as the thinnest bar.

Ratio bars/gaps:

- the bar thicknesses match the thicknesses of the gaps.

In comparison with the printed original, **scanned images** are always faulty, e.g. gaps and dots on the edge, step curves, etc. In a good image these errors are very rare, the bars and gaps should comply with the description above.

Good Scan:

Code 39: 123456

Failure cause: scan resolution is too low

If the resolution during scanning is too low, fine bars cannot be separated. Two or more bars become one and the gaps disappear. Thus the program cannot recognize the gaps and the numeral cannot be decoded.

Examples involving a scan resolution that is too low (50 dpi):

Failure cause: scan too dark

If the scan is too dark, the bars enlarge which makes thin gaps disappear. The barcode cannot be recognized. One way to avoid this error is to scan using a brighter setting. If the barcode cannot be scanned in a brighter way (otherwise other bright elements may disappear from the image) the barcode must be printed much wider.

Examples that has been scanned too darkly:

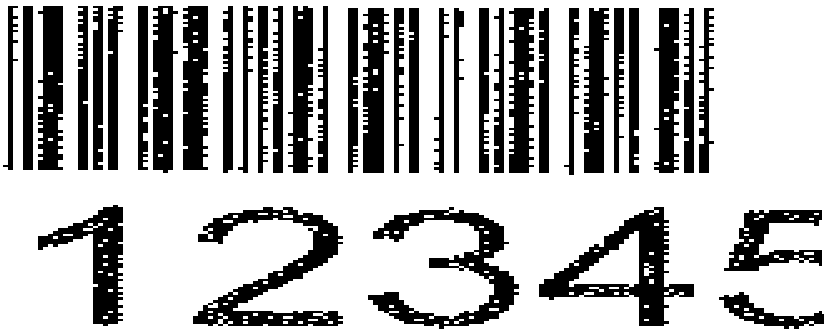
Failure cause: scan too bright

If the scan is too bright, bars which are necessary for the recognition can be missing.

Example that has been scanned too brightly:

Here you see some examples of barcodes which are not readable because of incorrect scanner settings. In part errors are only visible with a “magnifier” (to enlarge the image):

Outlines or enhancement of outlines
This scan adjustment results in displaying only the edges of wider bars. These thick bars are white in the middle and the barcode is not readable.



Dithering
Dithering gray scales are produced by an appropriate arrangement of black and white dots.
If the barcode is not properly printed in black, white dots are generated during scanning, which complicates the recognition. The dots are also visible in the text if the text has the same brightness.

Strong jpg Compression
The jpg image compression produces very small images. The procedure is working - if it is a strong compression - with loss of the image’s quality. The bars are smearing and the width cannot be determined.

3. Cause of Error: Inappropriate parameters for recognition

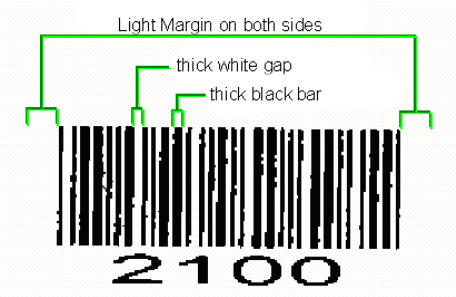
It may be necessary to change the standard parameters before recognition to find the optimal settings for the image you wish to recognize. This chapter explains which changes in the parameters may be helpful.

Proceeding

Perform first tests with the default settings of **bcTester**, and only then begin to change single parameters in the menu “Barcode Settings”. The following section explains some of these parameters in detail.

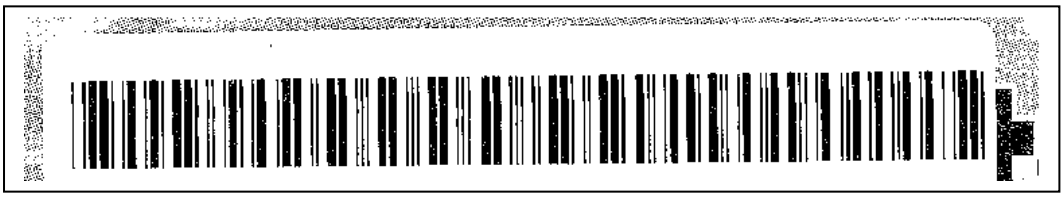
Light Margin

Before and after a barcode a white area as light margin is expected. This white area must be wider than any white gap within the barcode and wider than any black bar in the barcode. The figure at right illustrates this.

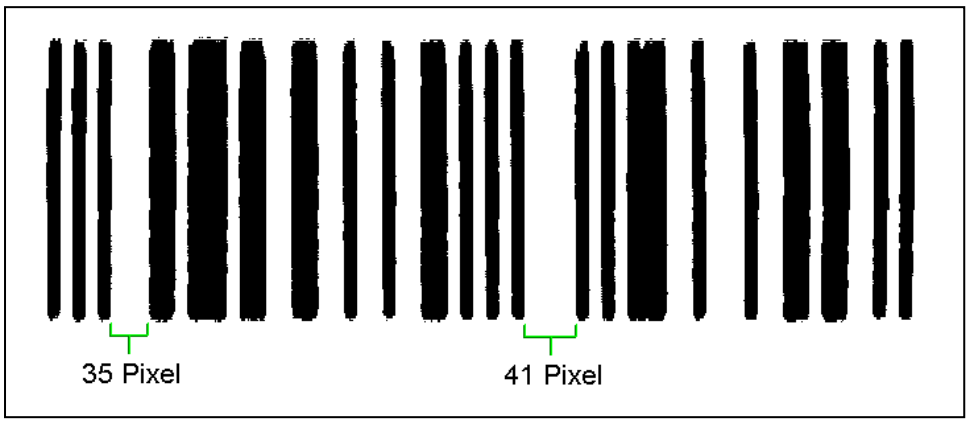


In bcTester, you find the light margin in the dialog “Barcode Settings”, <Extended Parameters>. There you can choose its size in pixels. The default value of 30 pixels refers to common 200dpi images. You should work with a higher value if your scans have a higher resolution.

The following example shows a barcode label on which the barcode is printed too close to the edge. The light margin behind the barcode is so small that the end of the barcode is not detected and the black bar next to the label is detected as black bar of the barcode. This prevents a correct recognition. In this case, reducing the light margin would not be helpful because then the white gaps contained in the barcode would also be recognized as light margins.



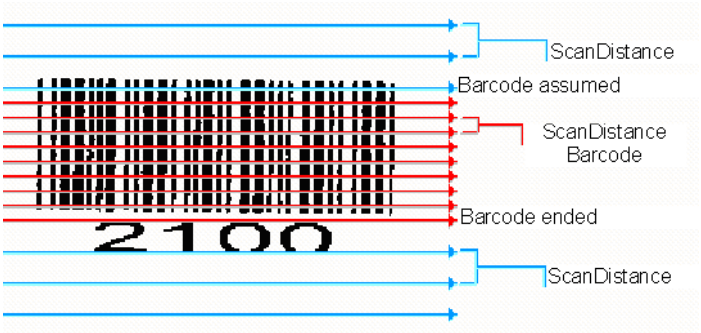
The following example was scanned at 600dpi. The high scanner resolution leads to white gaps contained in the barcode whose width is far beyond the standard value of 30 pixels. If you increase the light margin value to 45 pixels, the barcode can be recognised without any problems.



ScanDistance and ScanDistanceBarcode

The algorithm for Barcode Search searches the image row by row. In most cases it is not necessary to analyze every single row, which saves a lot of time.

Usually a “ScanDistance” of 15 rows is set. If a barcode is assumed, the algorithms changes to “ScanDistanceBarcode” (standard value: 3 rows) to collect more information inside the barcode.

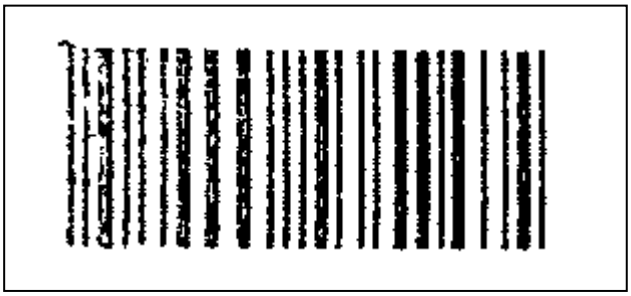


As a result, the “ScanDistanceBarcode” should always be smaller than or as large as the “ScanDistance”. Both parameters can be set in the dialog “Barcode Settings”, <Extended Parameters>.

To achieve optimal recognition results, choose 1 for both parameters. In this way *every* row of the image is read individually. Of course, this slows down the recognition speed. Thus these parameter settings are useful for testing purposes: if the recognition is successful you can enhance the values step by step.

Number of Barcodes: BC_ONE_BREAK
(ReadMulti)

In special cases, especially if barcodes are printed or scanned in a poor quality, you can try recognition using the special parameter ”BC_ONE_BREAK“.
As always, the image is read row by row. As soon as a barcode is recognized in one row, the recognition is stopped immediately and the barcode is reported as the result.
By choosing this mode, the general check



of quality and assurance that the pattern of gaps and bars is actually a barcode does not occur.
The pros and cons of this setting are described in the following table.

Pros	Cons
The setting „BC_ONE_BREAK“ allows the recognition of highly damaged barcodes even if only one row of the barcode is readable The setting „BC_ONE_BREAK“ allows fast recognition because the reading is stopped very early	Only one barcode can be found **It is possible that text characters or gray sharpes are wrongly interpreted as a barcode which is followed by an *incorrect* result! If you choose other settings, this error is eliminated by analyzing several rows to check the result.**

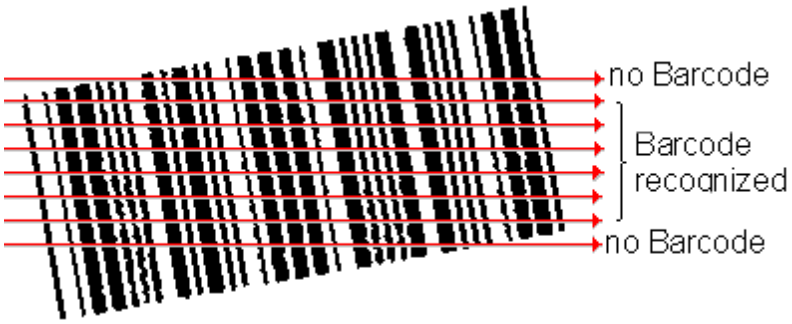
In most cases of use, it is better to achieve no results than to receive errors. In these cases you definitely should not use the setting ”BC_ONE_BREAK“.

The specification of an exact length, orientation and type reduces the error rate.

The barcode example above is of very poor quality; the first bars in particular are highly damaged. With this quality the used barcode should contain a checksum, so that erroneous results can easily be noticed!

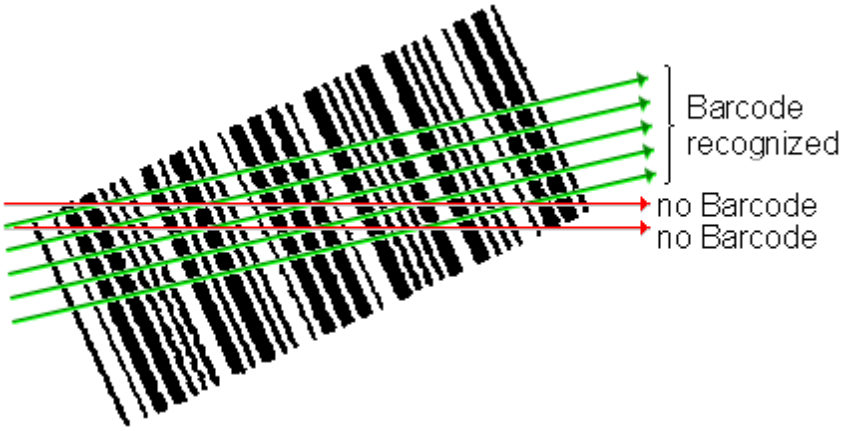
Rotations

With default settings, barcodes are searched with 0° / 180° and 90° / 270° rotation. Small variations up to 8° - as they emerge e.g. during scanning - are tolerated. If barcodes occur with a much higher rotation, the parameter “maximum skew angle” should be set to 21°, 34° or 46°. If you choose 46°, all directions are searched for.



The barcode on the left can be recognized with a skew angle of 0°.

The barcode on the right cannot be recognized with a “skew angle” of 0° (**red arrows**), but with a “skew angle” of 21°, because then the image is also searched diagonally (**green arrows**).



Improvement of Images - Preprocessing Functions

General

It is always better to improve the quality of an image by altering the image production than improving the image after its initial creation. Sometimes retroactive improvement of an images is not possible, for instance if generation of images cannot be affected.
The following examples contain descriptions of options for retroactive image improvements. Often these actions decrease recognition speed.

These functions for the following errors are provided:

- soiled images due to dots
- holes in the bars of the barcode
- other errors in the image, especially errors in the bars and gaps
- extreme values for brightness in color or grayscale images
- unequal exposure in color or grayscale images

The methods for image improvements are controlled by instructions. Instruction format is decribed below.

Controlling the image improvement

The image improvement functions are determined by a data file called `QSBC.ini;` if the file is available it is located in the program's directory. Computer applications, which use the QS-Barcode SDK interfaces F_DLL, H_DLL or F_OCX, use the `QSBC.ini` data file to improve the image.

Barcode recognition checks whether a data file called `QSBC.ini` exists. The file has the common format of a Windows-Parameter file and can be created using the text editor.

For instance, this is a typical `QSBC.ini` file:

```
[Imaging]
Steps=5
Step1=GraphLib:SubImage("c:\_test1.tif");
Step2=FreeImage:Threshold,160;
Step3=GraphLib:SubImage("c:\_test3.tif");
Step4=GraphLib:DespeckEx(10,2);
Step5=GraphLib:SubImage("c:\_test5.tif");
```

These commands convert color images to black and white images (command: Threshold) and delete black and white dots which can prevent the barcode recognition (command: DespeckEx).

The line `Steps=5` defines that there are 5 steps. For each step there has to be a line like `StepX=instruction;` X stands for a step number and `instruction` can be one of the instructions that are described beneath. The final semicolon in the instruction line is very important.

The improvement of the image is only performed on a copy of the image. The original image is not altered. All assigned steps are independant of barcode recognition. To check the image improvements, the instructions

```
Step1=GraphLib:SubImage("c:\_test1.tif");
Step3=GraphLib:SubImage("c:\_test3.tif");
Step5=GraphLib:SubImage("c:\_test5.tif");
```

are used. They save new images with the assigned file name. As a result, you can evaluate how the improved images appear. The file format is given by the file extension. Possible extensions are ”tif“ or „bmp“.

Dot removal

A lot of images show black and white pixels after the image production and binarization.

On this scanned image

you can see black dots between the bars. During the line-by-line scanning these black dots result in additional changing, which avoid the recognition of a barcode.

The function `Despeck`

```
GraphLib:DespeckEx(10,10);
```

removes black dots as well as white holes in barcode lines. The two numbers define the maximum diameter of pixel for the dots that should be removed. The first number is for the black dots and the second number for the white.

Only “solitary” dots with the assigned pixel-size will be removed. Dots which are in contact with others are not going to be removed.

The result for the upper image after the improvement looks like this and the barcode can be correctly recognized:

Note: Removing dots only works for monochrome (black-and-white) images. Color and Grayscale images have to be converted first. The content of a possible `QSBC.ini` file which can convert images looks like this:

```
[Imaging]
Steps=3
Step1=FreeImage:GrayScale;
Step2=GraphLib:Dynamic;
Step3=GraphLib:DespeckEx(10,10);
```

For further infomation on how to convert an image from a grayscale to a monochrome image please see the `Dynamic` section below.

Adjustment of Image Error

This function is an alternative to “Dot removal”. It works differently and offers other advantages and disadvantages:

Advantages:

- Correction of errors in spots on the images, in the white gaps as well as in the black bars.
- Is faster than “Dot removal“, time needed to complete is nearly zero!
- Whole vertical lines can be masked. These lines can occur due to defective scanner or fax or dirt on optical elements
- Can also be used for QS-Barcode SDK interface P_LIB

Disadvantages:

- Has to be configured according to the used barcodes and the resulution of their data images.
- Defective adjustments can completely inhibit barcode recognition.

How does it work?
During row by row scanning of the image **short black and white dots** are read over.

In the running direction all short dots that are smaller than a configurable width of pixels are read over.
In the figure on the right the settings

```
[CleanBresenham]
WhitePoints=2
BlackPoints=0
```

eliminate all dots in the red circles. If you choose `WhitePoints=1` the two dots in the green circles will not be deleted, because they are wider than the configured width.

In the INI-file in paragraph `[CleanBresenham]` you find the parameters for the setting “image error adjustment“.
To read over white dots use `WhitePoints=N`, to read over black dots use `BlackPoints=M`. It is also possible to read over both using both alternatives. N, M provide the maximum amount of pixels that should be read over.

Note! `N` and `M` must be chosen this way: in ensuring that regular gaps or bars in the barcode are not read over, they must be clearly wider than the configurated value.

"Hairline Errors"

Sometimes malfunctions of printer or scanners lead to optical errors in the image. There are very thin lines that block the barcode recognition. The "adjustment function" described above helps with these errors.

The following figures will clarify this error:

Image errors: vertical white bar

vertical bar marked in red

During row by row reading with setting "image error adjustment" the narrow white bar is read over (red marking)

Barcodes that have been scanned too brightly or too darkly

If scanned images are too bright or too dark, but are available as color or gray scale images, this may be corrected by the conversion (binarization) to a black and white image (monochrome).

The transformation usually occurs automatically prior to barcode recognition. In **bcTester** this standard transformation can be performed in the menu under <Effects> <Change to Monochrome >. Following this, the binarized image will be shown.

If the scanned barcode is too dark or too bright, you can affect the transformation with the following functions.

A threshold is used for the transformation and binarization of images. Pixels that are brighter than the threshold will be white; the other pixels will be black. Possible values for the threshold range from 0 (black) to 255 (white).

Internally a value of 160 is used for converting the images to black and white. It is a good value for most cases.
For the internal conversion to black and white usually a threshold of 160 is used. This value has been found to be good in many cases.

You may view the black and white image in bcTester with **<Effects> <Convert to monochrome>**: If the image is to light create a qsbc.ini file with paragraph [Imaging] and one line Threshold=175.

More functions for binarization

The following table includes possible options for different methods for binarization; they differ in achieving the threshold.

Note
The following GraphLib commands works only on gray scale images (8 bits).
Color images can be converted to gray scale as a first step:

```
Step1=FreeImage:GrayScale;
```

Construction	Desciption
`GraphLib:Dynamic;`	The threshold value is determined from the brightness values occurring in the image. This corresponds to the "automatic brightness adjustment" (which is used by most copying machines). For the determination of the "dynamic threshold" the whole image is used, not just the barcode area.
`GraphLib:DynamicEx("20");`	Threshold is determined the same way as using `Dynamic`, but it is adjusted by an offset value. If the value of the offset is positive, image will get darker, if the mismatch is negative, the image will get brighter.
`FreeImage:Threshold,128;`	This method provides the threshold used directly. Possible values are between 0 (black) and 255 (white).

Usually the value `Dynamic` delivers good results. Sometimes thresholds that are acquired automatically are unsuitable, for example if barcode labels are printed brighter than the remaining image. In that case you can eventually achieve a barcode recognition by changing the threshold with `DynamicEx`, or by specifying the threshold used with `Binarize`. Generally, methods that acquire thresholds dynamically are better because they assimilate with changed scan settings.

Example

Barcodes that are scanned too brightly

`Binarize(128);`
middle value chosen is too low

`Binarize(200);`
a higher threshold fits for the same image

`Dynamic;`
By using dynamic a good value is acquired

Uneven exposure

The upper right corner of the image

is brighter than the lower left corner. The image is transformed into a black-and-white image as described in the paragraph above. As one threshold is used for the entire image, the following result will be shown:

This barcode is unrecognizable.
The reason that the barcode can be easily recognized by the human eye is due to the eye' s ability to evaluate black and white locally.

By using the instruction

```
GraphLib:LocalThreshold(0,20,"0.9");
```

a "local threshold" is also used for binarization.

This image improvement is affected by three figures. The first 0 is a reserve parameter for later improvements of this function and cannot be altered. The second position affected the size of the area in which the threshold is acquired. The value 20 turned out to be good.

Choosing a lower value increases speed, but also increases the risk that holes occur in consistently white or black areas caused by small variations of brightness. The “0.9“ on the last position indicates that all pixels of the image are considered as white, if their brightness is above 90% of the average brightness of the surrounding areas.

The result

no longer shows effects of shading and the barcode recognition is good. However, calculating this function requires a lot of time.

Note: QualitySoft updates this documentation regularly. We welcome your suggestions in correcting errors or to clarify any details. QS-Barcode products are continuously managed, updated and improved.