QS-Barcode Recognition

(32- and 64-Bit Version)

Software Developer's Kit

Version 5.0.1

QS QualitySoft GmbH
Tempowerkring 21a
D-21079 Hamburg, Germany

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

For the latest information visit:
www.qualitysoft.de

Content

1 Introduction 4
2 Supported Barcode Types 5
3 Which interface should I use? 7
4 License Files 9
5 About Barcodes 10
5.1 Overview 10
5.2 Barcode Types 10
5.3 Option 2D Barcodes: Overview 12
6 The application bcTester 13
6.1 Testing Barcodes 13
6.2 Barcode Settings 14
6.3 Barcode Results 18
6.4 Barcode Analysis 18
7 Library with Pointer (p_lib) 19
7.1 Interface 19
7.2 Integration Examples 20
7.3 Definitions 21
7.4 Redistributable Files 23
8 DLL with Handle (h_dll) 24
8.1 Interface 24
8.2 Definitions 27
8.3 Redistributable Files 28
9 DLL with File (f_dll) 29
9.1 Interface 29
9.2 Integration Example 31
9.3 Definitions 33
9.4 Redistributable Files 34
10 ActiveX with File (f_ocx) 35
10.1 Description 35
10.2 Integration 35
10.3 Methods 36
10.4 Call Properties 37
10.5 Result Properties 38
10.6 Short Example Application 39
10.7 Redistributable Files 40
11 Parameters 41
11.1 Barcode-Parameters 41
11.2 Advanced Search parameters 48
11.3 Return Values 50
12 Advanced Search 53
12.1 Parameters of the advanced Search 53
12.2 DynamicThreshold 54
13 Image Processing 56
14 Special Settings 62
14.1 Special settings for patch codes 62
14.2 Special settings for Data Matrix Code 62
14.3 Special settings for QR Code 63
14.4 Special settings for PDF417 63
14.5 Special settings for Aztec Code 63

14.6 Special settings for PostNet Codes 63
15 Troubleshooting 65
15.1 ActiveX registration 65
15.2 DLL not found 65
15.3 Alignment problems 65
15.4 Linker Problem 65
15.5 Barcode cannot be recognized 67
15.6 Systematic result-altering 67
16 Appendix 68
16.1 Supported File Formats 68
16.2 MultiPage support 68
16.3 Adobe PDF Documents – Special Settings 69
16.4 Image enhancement via "qsbc.ini" 69
16.5 QR Code Extended Channel Interpretation (ECI) Mode 72
16.6 Update Notes 73
17 Final Note 73

1 Introduction

QS-Barcode is an efficient software for recognizing barcodes from digital images (monochrome, grayscale and color) which is available as a 32-Bit and 64-Bit version since version 4.9 and which is ready for multithreading since version 5.0.

HINT	Looking for software to **Create** and **Print Barcodes**? Have a look at our partners page: http://www.qualitysoft.de/index.php?id=25&L=1

QS-Barcode SDK has gone a long way since its first release in 1998. A lot of new barcode types have been added in the last years, the recognition algorithms have been optimized and the recognition rates are constantly rising.
Barcode recognition with **QS-Barcode SDK** runs on more than 50.000 machines worldwide.

The digital images of the barcodes are created by scanning, digital photography or fax. Barcode in Adobe PDF documents can be read as well.

QS-Barcode SDK supports monochrome (black and white) images as well as grayscale and color images. All common Image File Formats (Bitmap, Tiff, JPEG etc.) are supported. Reading barcodes on Adobe PDF documents works as well. QS-Barcode SDK also offers interfaces to work directly on the image data in memory without file input/output.

QS-Barcode SDK is no stand-alone software; it is a Software Development Kit offering a library for developers to integrate barcode recognition features in other software solutions. It offers the interfaces LIB, DLL and OCX.
QS-Barcode runs on Windows operating systems.

HINT	You want to recognize barcodes but you do not want to start programming? Have a look at our product **QS-DocumentAssembler**, it might be the right choice for you. https://qualitysoft.de/products/qs-documentassembler

This **QS-Barcode SDK** manual explains how to integrate the barcode recognition routines into your environment and how to deploy and install it on your customers' machines.

2 Supported Barcode Types

QS-Barcode recognizes the popular barcode types **Codabar**, **Code 2/5 interleaved, Code 2/5 industry, Code 39, Code 39 extended**, **Code 32**, **EAN 8**, **EAN 13, UPC A, UPC E, Code 128**, **EAN128** and **Code 93**.
It also recognizes barcode types for special purposes: **Code 2/5 IATA**, **Code 2/5 3 Matrix**, **Code 2/5 3 Datalogic**, **Code 2/5 BCD Matrix, Code 2/5 inverted, Code 93 extended** and **Code 11**.
With special parameters it recognizes (Standard One-Track) **Pharmacodes** as well as **Patchcodes** and the Stacked-Barcode **Codablock F.**

As additional **option** the recognition of the two-dimensional Barcodes **PDF417, Data Matrix (ECC 200 and ECC000, ECC050, ECC080, ECC100 and ECC140)**, **QR Code (Model 2)** and **Aztec Code** is available.
These barcodes contain much more information than regular barcodes. Depending on the type and size up to 3,000 characters can be coded. The entire text of this page would fit in one single barcode. Additionally, 2D barcodes are more tolerant towards errors that can occur while printing or scanning, because of the built in error correction.

The **Data Matrix** barcode was designed for **small parts marking** and today is used for small electrical parts by the pharmaceutical industry for unit dose packaging,

by the automotive industry and by the NASA. The German Post uses it as "Stampit" Stamps. It contains binary data. Both Data Matrix Bar-codes samples are printed in **original size**.

The **PDF417** barcode is composed of a stack of rows. You can encode up to **2,700 characters**. Both the number of rows and columns are selectable.

It was selected by the **German National Association of Statutory Health Insurance Physicians** for printing on medical forms.

QR Code was designed in Japan and is widely used in Asia. It is build and optimized to encode Kanji-Characters, but encoding of binary data and alphanumeric data is also available.
The **QR Code** has a built-in error correction like the other 2D barcode have.

The square **Aztec Code** was designed 1995 by Welch Allyn. It has **one** finder element in the center of the code ("The Pyramid"). The Aztec Code does not require a light margin around the barcode as other codes do. Size varies from 15x15 to 151x151 elements. As other 2D barcodes, Aztec Code has an intelligent error correction built-in.

The Aztec Code is used by "Deutsche Bahn AG" to encode online-Tickets data.

3 Which interface should I use?

QS-Barcode is available in various interfaces for different needs. This is a short presentation so that you can decide which interface is suitable for your project.

In a first step we distinguish between interfaces that themselves read the images, where the barcode is searched for and those that work on images already existing in memory.

If you wish to sort scanned images of a directory according to barcodes printed on the forms, you can use the **file interface** and you do not have to open them yourself. There are two interfaces which load the images themselves: **DLL with file (f_dll)** or **ActiveX with file (f_ocx)**. These two interfaces do support Adobe PDF documents as well!

On the other hand, if you search for barcodes on images you have already loaded into your program, because you want to manipulate the image or you need high performance or want to use only areas of the image, you can use interfaces that work on these **images in the memory** directly. This way saving images in between is no longer necessary. Using either the **C-library with pointer (p_lib)** that has a pointer to the bits of the image spots, or the interface **DLL with handle (h_dll)**, which expects a handle to a loaded image, depending on your developer environment.

It also depends on the type of image material which interface you should choose:
The high-level interfaces **f_ocx, f_dll** and **h_dll** support color, grayscale and monochrome images.
The **p_lib** interface supports only monochrome images.

Delivered versions of Libs/DLLs

All QS-Barcode DLLs and LIBs are shipped in four versions. (MD, MDd, MT, MTd).

1. Multi-threaded (MT)
2. Multi-threaded Debug (MTd)
3. Multi-threaded DLL (MD)
4. Multi-threaded DLL Debug (MDd)

They are distinguished by the different embedded runtime libraries and exist as a debug and a release version.

The versions of MT and MTd mean that libcmt.lib was statically linked.

The versions of MD and MDd mean that msvcrt.lib was linked dynamically.
This has the consequence that the msvcr120.dll and msvcp120.dll must be delivered separately. These files are located in the "Microsoft Visual C ++ 2013Redistributable" package.

If this distinction does not say anything to you, choose the version of the "MT".
QS-Barcode is used successfully in various development environments:
C# und .NET, Visual Basic, Delphi, Fortran, Visual Basic for Applications, C and C++,

Java, SQL Windows, Windows Scripting Host.
You find samples for integration in the following chapters.
QS-Barcode runs reliably with Microsoft Operating Systems Windows Server 2003, Windows Server 2008, Windows Server 2012, Windows Vista, Windows 7 and Windows 8.x.

4 License Files

QS Barcode-SDK uses a file based license method. Two license files named "qsbc.lic" are included in your distribution of the QS Barcode-SDK. The first license file is for use on the developers work station where the SDK is installed and the application is developed. The second license file is the first "run time license". In addition to the first runtime license file you need to purchase one runtime license for every additional work station where QS Barcode is used. You must distribute this file with your application using QS Barcode. License files are searched for in the application path.

If the license file cannot be found, QS Barcode runs in **demo mode** with systematic substitutions of result characters. For 2D barcodes, a message appears that the barcode type is not supported.

The following license types are used.

Name	**Mode**	**Value (HEX)**
BC_LIC_DEMO	Demo-Mode	0x00000001
BC_LIC_LINEAR	Mode for linear Barcodes	0x00000002
BC_LIC_PDF417	Mode for PDF417 Barcodes	0x00000004
BC_LIC_DATAM	Mode for DataMatrix Barcodes	0x00000008
BC_LIC_QRCODE	Mode for QR Code Barcodes	0x00000010
BC_LIC_AZTEC	Mode for Aztec Barcodes	0x00000020

A combination of the different types is possible using the OR operator.
A value of 0x00000007 means BC_LIC_DEMO OR BC_LIC_LIN OR BC_LIC_PDF417, QS-Barcode reads linear and PDF417 Barcodes in Demo-Mode.

To find out about the current license mode, the system function QSLicense() can be used.

5 About Barcodes

5.1 Overview

Barcode recognition is mainly used for identification and indexing of documents for data recognition (OCR) and archiving. Scanning in this area is usually done with a resolution from 200 to 300 dpi. To achieve good barcode recognition results without errors at this low resolution, the barcodes must be printed very clearly and they must not be positioned too closely to each other. Recommended is a maximum of 2 characters/cm at 200 dpi and 3 characters/cm at 300 dpi. For example, a barcode of eight digits should have at least a 4 cm width to guarantee recognition at 200 dpi.
In addition to scanner resolution and barcode width, there are additional factors that influence reliable recognition:

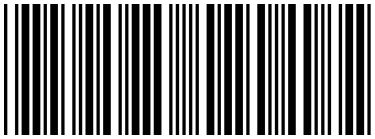
- Barcode type
- Scanner settings
- Height of the barcodes
- Print and paper quality
- Displacement (particularly of sticky labels)
- Light margin around the barcodes

The number of bars and blanks are important for barcodes as well as the relative width.
Anything that changes the appearance of the bars leads to errors in barcode symbol recognition. For example, the contrast setting changes the width of the bars.

QS-Barcode offers several parameters to make recognition quality reliable in terms of basic needs.
Before you begin using barcodes in daily routine, please perform sufficient testing.
The barcodes you wish to use should be tested in near-real situations.
Do not hesitate to send us your questions and test images: info@qualitysoft.de.
We are always happy to help you!

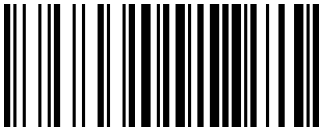
5.2 Barcode Types

During the last decades various types of barcode types have been developed. Below you can see the common linear barcodes:



Code 39: 123456

Code 2 of 5 interleaved: 123456



Code 128: ABC987

EAN 13

Barcodes can contain different information. The following table provides an overview of some linear barcodes recognized by our SDK:

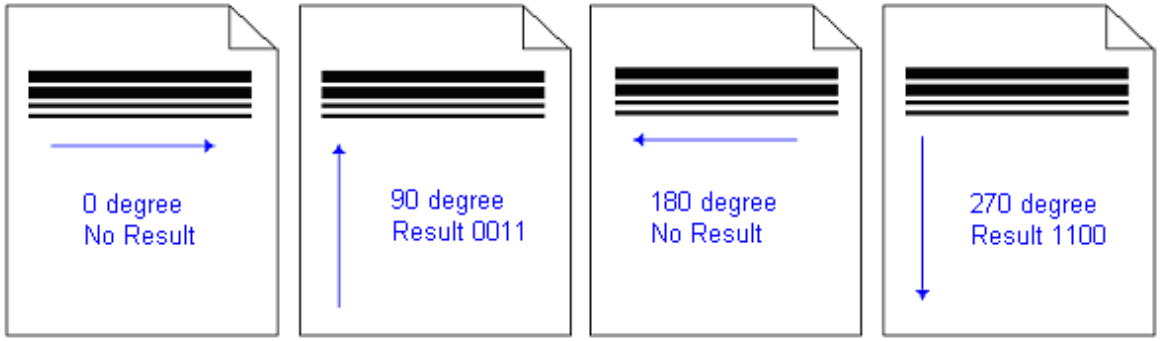
	Characters	Length
Code 39	0-9 A-Z - $ % + /	any
Code 39 extended	Complete ASCII-character set	any
Code 2/5 interleaved	0-9	any (even number)
Code 2/5 Industry	0-9	any
Code 93	0-9 A-Z - $ % + / 4 special chars	any
Codabar	0-9 - $: + / .	any
Code 128	complete ASCII character set	any
EAN 128	complete ASCII character set	any
EAN 8 / EAN 13	0-9	8 / 13
UPC A / UPC E	0-9	12

Besides the types named above, QS-Barcode is able to read several others, most of which are only used in special environments or for historical reasons. You can find the actual list of types in the description of our test application (below).
If you need to recognize other barcode types please contact us. We will always be glad to help you.

Special treatment is needed to read **"Patchcode"** or **"Pharmacode".** Both codes are used in very specific fields, they cannot be used in general. They miss the start and stop characters which are very useful to locate barcodes and distinguish different codes.

To recognize patchcode or pharmacode you must call the barcode recognition just for the wanted type, you cannot mix types as you usually can.

Patchcode is used by some scanners to control and separate scan jobs.
Patchcodes always have just 4 very long bars.



The recognized patchcode will be reported as usual.

Pharmacode is used to identify medicine during production and packaging.
Note: In different countries the name "pharmacode" is used for various codes.

Samples of Pharmacodes

Pharmacode consists of 4 to 16 bars and code numbers from 1 to 131070. You should always specify a zone to search pharma code, otherwise you may get some other patterns recognized as "Pharmacode".
Recognition of Two-Track Pharmacode and 2D-Pharmacode is available on demand.

5.3 Option 2D Barcodes: Overview

2D (two dimensional) barcodes were developed to decode much more data in a single code. In the early years of 2D barcode. a lot of different types were generated from various organizations and vendors.
Nowadays the PDF 417, Data Matrix and the QR Code (in Asia) are used for most purposes. Both can be generated in different sizes and may contain more than 2000 characters.
Since binary data can be decoded as well, these barcodes are often used for ID-Cards with biometric data (finger prints, etc.).

QR Code PDF 417 Data Matrix Aztec Code

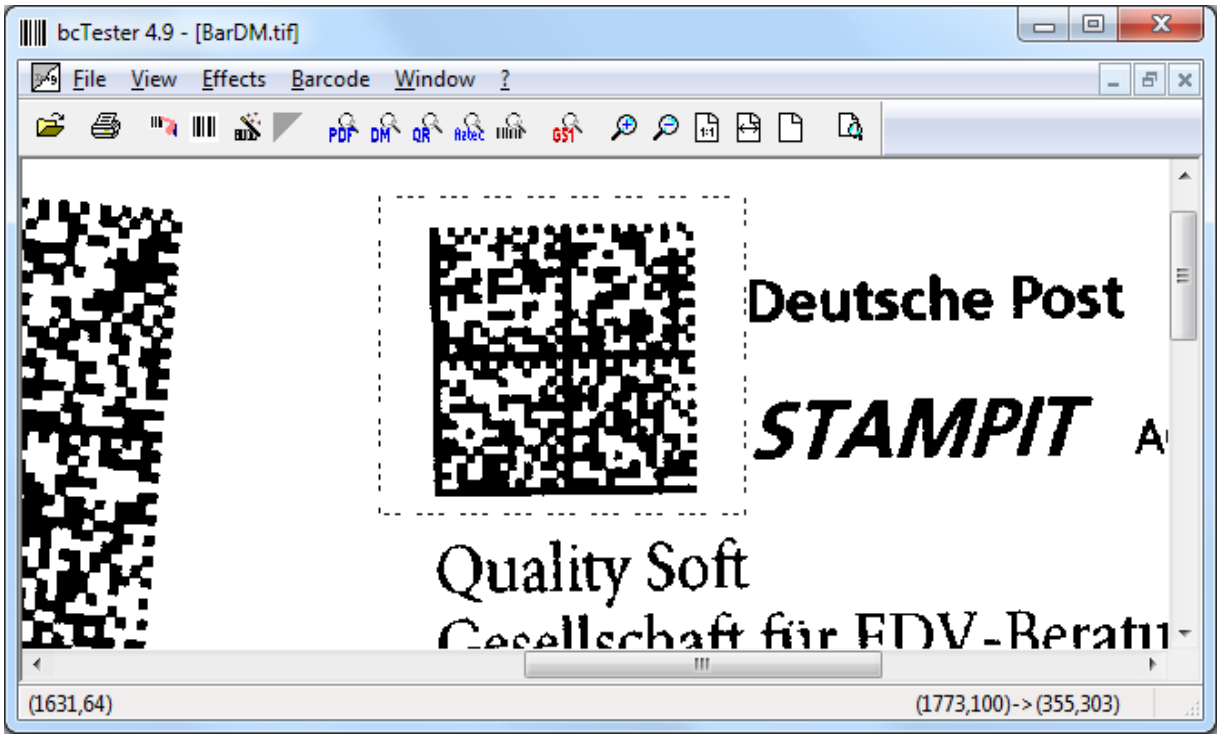
Three options of **QS-Barcode** are available to recognize **PDF 417**, **Data Matrix (ECC 000-140 and ECC 200)**, **QR Code (Model 2)** and **Aztec Code.**

Integration of the Library in your own code is similar to the Linear Version. The options are shipped with sample images and sample codes.
At our website www.qualitysoft.de and in the test application you find interesting case studies for 2D-barcodes.

Please note: If you want to recognize linear and 2D barcodes on one image, you will have to start the recognition function twice: Once for the linear types and once for the 2D barcode type. This is because the recognition parameters are different for the 2D barcode types. See further down.

6 The application bcTester

Using bcTester you can test your barcodes and easily experiment with different barcode settings.
At https://bctester.de/download-2 you can always find the latest version.



6.1 Testing Barcodes

The procedure is as follows:

1. Open an image which contains the barcode you wish to test (Drag and Drop is supported.).

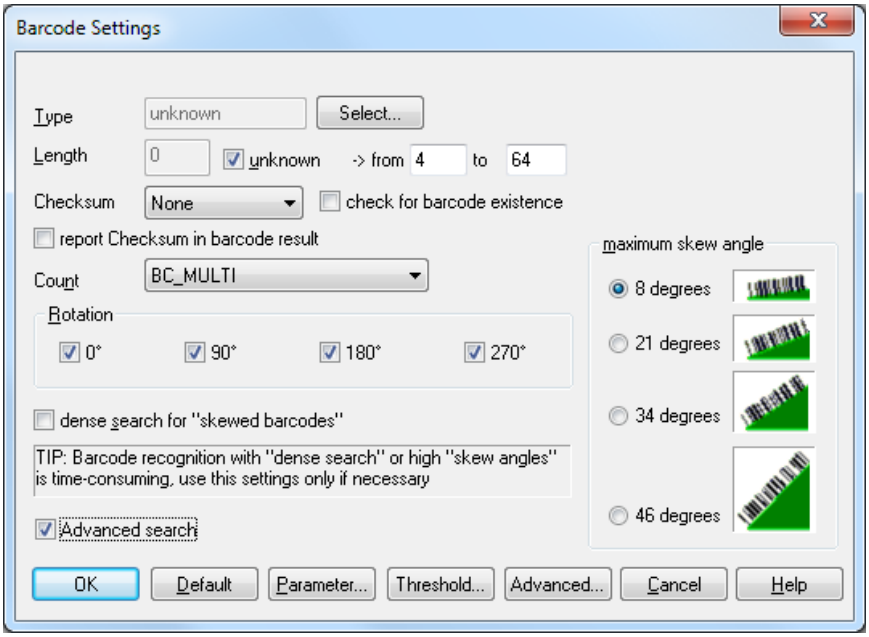
2. Open a rectangle around the barcode with the mouse. If you do not open a rectangle the search is performed on the whole image.

3. Set up the barcode parameter under <Barcode><Settings...> (CTRL-N).

4. Start the recognition with <Barcode><Recognize> (CTRL-L). The results are displayed.

6.2 Barcode Settings

In this section you find the settings that are described in the "Barcode-Parameters" chapter.



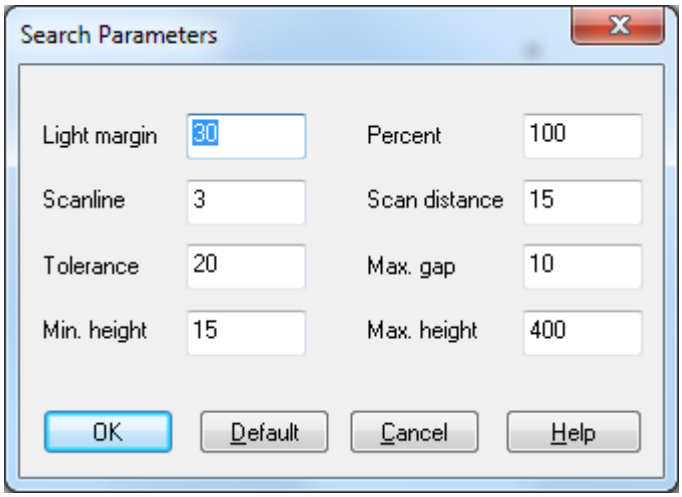
Under "Select" you can select one or several types.

Use the button "Extended" to enter settings for the light margin and the scan distance. These are used more rarely.

For additional help on settings please click "Help".

Settings for bcTester	**"Barcode-Parameters" in SDK**
Type	iBC_Type
Length	iBC_Length
unknown	iBC_Length = 0
from	iLaengeVon
to	iLaengeBis
Checksum	iBC_Checksum
check for barcode existence	iBC_Checksum_EXISTENCE
report checksum in barcode result	iBC_Checksum or BC_REPORT_CHECKSUM
Count	iBC_ReadMultiple
Rotation	iBC_Orientation (values: BC_0, BC_90, BC_180, BC_270, may be linked with OR)
maximum skew angle	iBC_Orientation (value: SKEW_LIGHT OR BC_SKEW_MEDIUM OR BC_SKEW_HEAVY)
dense search	iBC_Orientation (value: OR BC_SKEW_DENSE_SEARCH) For DataMatrix the value is DM_INTENSIVE_SEARCH
Advanced search	AdvancedSearch

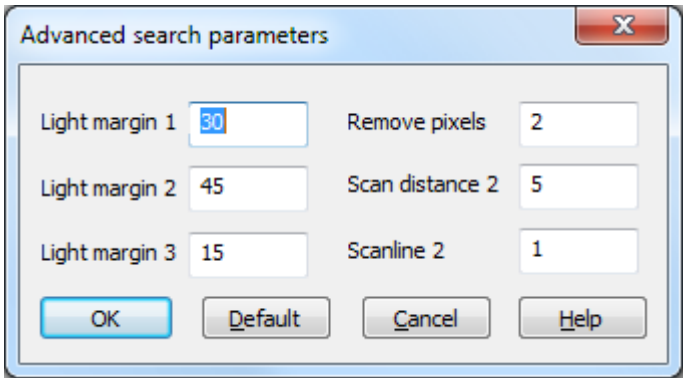
Using the button "Parameter" opens a dialog for special search parameters.



The default values are the best choice for barcodes of normal form and size (have a look at our sample images).
If you have special codes or bad quality barcodes adapting the parameters usually helps to get good recognition results.
For 2D barcodes, only some on none of these options at all are available. The meaning of the parameters is different there as well. See further down for more information.

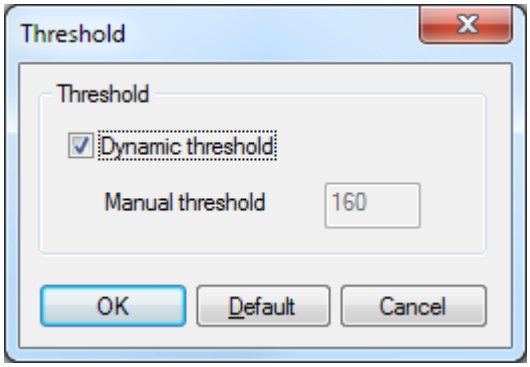
Settings for bcTester	**"Barcode-Parameters" in SDK**
Light Margin	iBC_LightMargin
Percent	iBC_Percent
Scanline	iBC_ScanDistBarcode
Scan distance	iBC_ScanDistance
Tolerance	iBC_Tolerance
Max. gap	iBC_MaxNoVal
Min. height	iBC_MinHeight
Max. height	iBC_MaxHeight

Using the button "Advanced" opens a dialog for advanced search parameters.



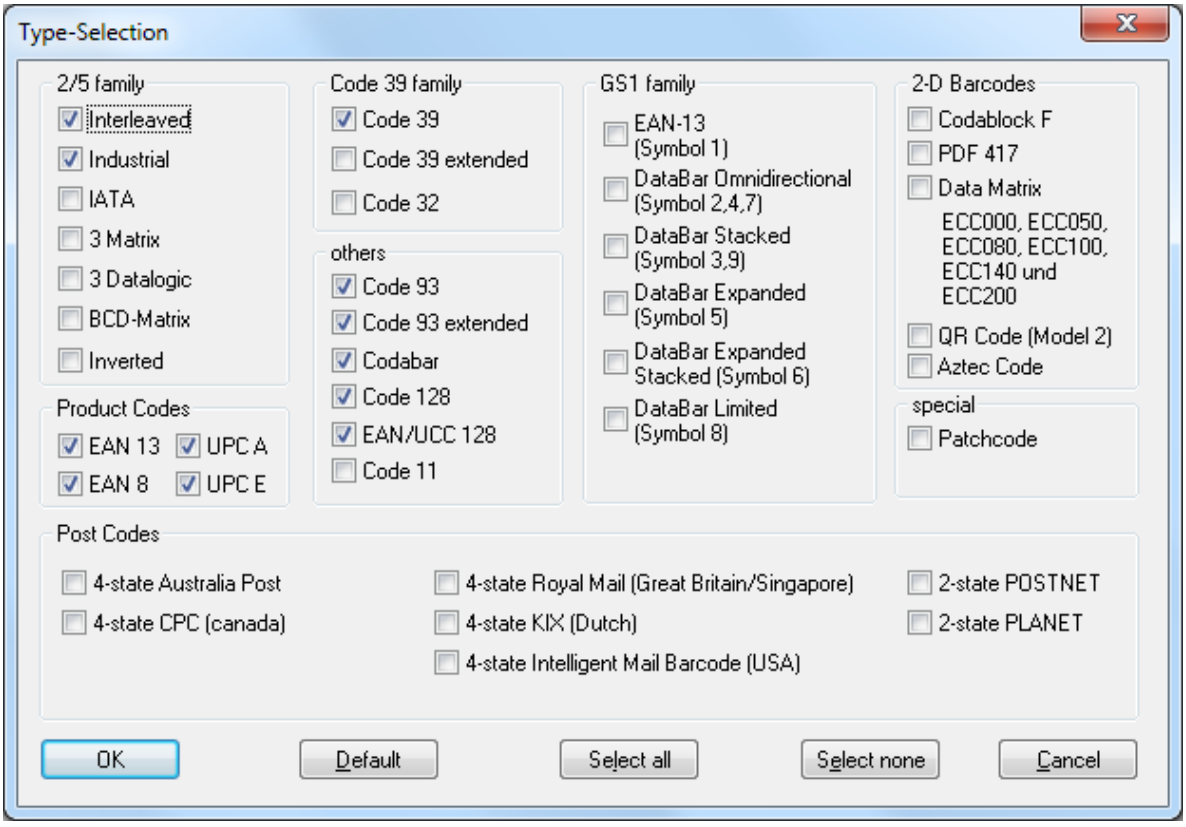
Settings for bcTester	**"Barcode-Parameters" in SDK**
Light margin 1	iBC_Lightmargin1
Light margin 2	iBC_Lightmargin2
Light margin 3	iBC_Lightmargin3
Remove pixels	iBC_RemovePixel
Scan Distance 2	iBC_ScanDistance2
Scanline 2	iBC_ScanDistBarcode2

Using the button "Threshold" opens a dialog for setting the threshold.



Settings for bcTester	**"Barcode-Parameters" in SDK**
Dynamic threshold	DynamicThreshold
Manual threshold	iBC_Threshold

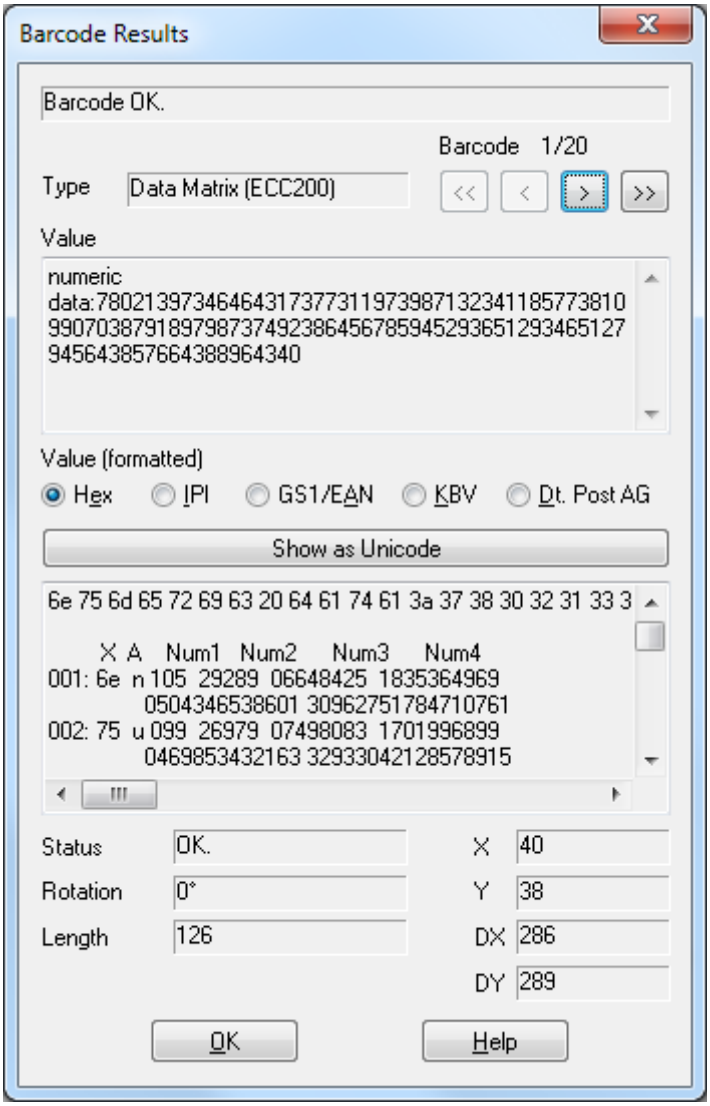
To select the types of barcodes to recognize select the appropriate check boxes.
In the SDK you set iBC_Type with the defined values.



BC_INTERLEAVED25 BC_INDUSTRIE25 BC_25_IATA BC_25_3MATRIX BC_25_3DATALOGIC BC_25_BCDMATRIX BC_25_INVERTIERT BC_EAN13 BC_UPC_A BC_EAN8 BC_UPC_E	BC_CODE39 BC_CODE39EXT BC_CODE32 BC_CODE93 BC_CODE93EXT BC_CODABAR BC_CODE128 BC_EAN128 BC_CODE11	BC_GS1_EAN13 BC_GS1_DATABAR BC_GS1_DATABAR_STACKED BC_GS1_DATABAR_EXP BC_GS1_DATABAR_EXP_STACKED BC_GS1_DATABAR_LIMITED	BC_CODABLOCK BC_PDF417 BC_DATAMATRIX BC_QR_CODE BC_AZTEC BC_PATCHCODE
BC_POSTNET_AUSTRALIAN BC_POSTNET_CPC	BC_POSTNET_ROYAL BC_POSTNET_KIX BC_POSTNET_IMB	BC_POSTNET_ZIP BC_POSTNET_PLANET	

Table: Defines for type selection

6.3 Barcode Results



The window displays your results. Use the buttons "<<", "<",">", ">>" to toggle between the single results. Type and value are displayed along with information on position, length and rotation.
The second list box shows the result in different formats. Hex is useful for 2D barcodes which may contain non printable characters.
Check the settings if a barcode is not found.

6.4 Barcode Analysis

Barcode Analysis is used to find correct settings.
Many recognition calls are performed quickly with various settings.
These settings are changed automatically step by step. For more information please click the <Help> button with the test application.

7 Library with Pointer (p_lib)

7.1 Interface

A section of an image – or also the whole image -, and a set of parameters are transferred to the library. As result you receive a value and one or several result structures. In case of an error, the result value shows the error in detail. Additionally, the result structures show recognition errors.

This is performed using the functions:

```
int QSBarcode2(BarcodeParam2 *pBarcodeParam2);
int QSBarcode3(BarcodeParam2 *pBarcodeParam2, char *szIniFile);
int QSBarcode4(BarcodeParam2 *pBarcodeParam2, char *szIniFile, char
*szLicenseFile);
int QSBarcode5(BarcodeParam2 *pBarcodeParam2, char *szIniFile, char
*szLicenseFile, AdvancedMode *pBC_AdvancedMode);
int QSBarcode6(BarcodeParam2 *pBarcodeParam2, char *szIniFile, char
*szLicenseFile, AdvancedMode *pBC_AdvancedMode, char *szImaging);
```

The **function parameter** p`BarcodeParam2` is a structure and is defined as described in the "Definitions" chapter.

The **function parameter** `szIniFile` is a char pointer which contains the fully qualified filename of a "qsbc.ini". This parameter can be passed as an empty string when using `QSBarcode4()`. If you pass no "qsbc.ini", this parameter is ignored. (See chapter 16.4 Image enhancement via "qsbc.ini"

The **function parameter** `szLicenseFile` is a char pointer which contains the fully qualified filename of a license file "QSBC.lic".

The **function parameter** `pBC_AdvancedMode` is a structure and is defined as described in the "Definitions" chapter.

The **function parameter** `szImaging` is a char pointer which contains semicolon separated commands for image enhancement.

To determine your **license** type use the function

```
int QSLicense( void );
```

or

```
int QSLicense1( char *szLicenseFile );
```

by directly passing the fully qualified path of the license file.

These functions return a combination of the following constants: `BC_LIC_DEMO`, `BC_LIC_LINEAR`, `BC_LIC_PDF417`, `BC_LIC_DATAM`, `BC_LIC_QR_CODE`, `BC_LIC_AZTEC`

7.2 Integration Examples

This example (which cannot be compiled!) shows the integration of the library into the code. For example files that can be compiled please see "C-Example program".

```
LPSTR lpBits;                   // points to the bits of the pixel
int nHeight;                    // contains the height of the image
int nWidth;                     // contains the width of the image
BarcodeData2 barData2;          //Barcode data
BarcodeParam2 barParam2;        //Barcode parameter
void FAR *lpResultMem = 0;      //Pointer to the result
int iNumResults=10;             //Maximal Number results
// Supply needed memory
lpResultMem = malloc(sizeof(BarcodeResult2) * iNumResults);
// At first empty structures
memset(&barParam2, 0, sizeof(BarcodeParam2));
memset(&barData2, 0, sizeof(BarcodeData2));
// fill in the BarcodeData- Structure with suitable values ...
barData2.iBC_Type = BC_INTERLEAVED25 | BC_CODE39; //search these types
barData2.iBC_Type2 = BC_NONE;
barData2.iBC_Length = 0; //Length unknown
barData2.iLaengeVon = 4; //somewhere between 4
barData2.iLaengeBis = 12; //and 12
barData2.iBC_Checksum = 0;
barData2.iBC_Orientation = BC_0 | BC_90 | BC_180 | BC_270;
barData2.iBC_LightMargin = 18;
barData2.iBC_ScanDistance = 3;
barData2.iBC_ScanDistBarcode = 1;
barData2.iBC_ReadMultiple = 1;
barData2.iBC_Percent = 100;
barData2.iBC_MaxNoVal = 10;
barData2.iBC_Tolerance = 20;
barData2.iBC_MinHeight = 15;
barData2.iBC_MaxHeight = 400;
barData2.pBC_NextBarcodeData2 = NULL;
// fill in the BarcodeParam2-structure with the correct values ...
barParam2.lpstrBC_Image = lpBits;          //Bits of the pixel
barParam2.pBC_BarcodeData2 = &barData2;    //BarcodeSettings
barParam2.iBC_Width = nWidth;              //Width of the image
barParam2.iBC_Height = nHeight;            //Height of the image
barParam2.iBC_StartX = 0;
barParam2.iBC_StartY = 0;
barParam2.iBC_SizeX = 0;
barParam2.iBC_SizeY = 0;
barParam2.BC_RotInfo.dBC_cos = 1.0;        //Important: 1.0!
barParam2.BC_RotInfo.dBC_sin = 0;
barParam2.BC_RotInfo.iBC_BMoffsetX = 0;
barParam2.BC_RotInfo.iBC_BMoffsetY = 0;
barParam2.BC_RotInfo.fBC_bRotated = 0;
barParam2.BC_RotInfo.iBC_offsetX = 0;
barParam2.BC_RotInfo.iBC_offsetY = 0;
barParam2.BC_RotInfo.dBC_XKorr = 0;
barParam2.BC_RotInfo.dBC_YKorr = 0;
barParam2.pBC_Memory = lpResultMem;        //results to this point
barParam2.lBC_MemorySize = sizeof(BarcodeResult2) * iNumResults;
barParam2.iBC_Debug = 0;
// now we can start the search ...
iReturn = QSBarcode2(&barParam2);

//put out results
for(nResult=0;nResult < barParam2.iBC_ResultCount;nResult++) {
```

```
        printf("%i. result: %s\n", nResult,
               barParam2.pbrBC_Result2[nResult].szBC_Barcode);
    }

    if(lpResultMem)
        free(lpResultMem);
    lpResultMem = NULL;
```

7.3 Definitions

Information about single values are found in the "Barcode-Parameters" chapter.

```
typedef struct BarcodeParam2_tag
{
    char           szBC_Version [30];
    PBarcodeData2  pBC_BarcodeData2;
    LPSTR          lpstrBC_Image;
    int            iBC_Width;
    int            iBC_Height;
    int            iBC_StartX;
    int            iBC_StartY;
    int            iBC_SizeX;
    int            iBC_SizeY;
    RotInfo        BC_RotInfo;

    void FAR       *pBC_Memory;
    long           lBC_MemorySize;
    BarcodeResult *pbrBC_Result;
    int            iBC_ResultCount;
    char           szBC_SubstitutionString [10];
    char           szBC_SubstitutionChar;

    int            iBC_Debug;
    HFILE          hBC_ErrorFile;
    HANDLE         hBC_Reserve;
} BarcodeParam2;


typedef struct BarcodeData2_tag
{
    int            iBC_Type;
    int            iBC_Type2;
    int            iBC_Length;
    int            iBC_Checksum;
    int            iBC_Orientation;
    int            iBC_ReadMultiple;
    int            iBCLightMargin;
    int            iBC_ScanDistance;
    int            iBC_Percent;
    int            iBC_ScanDistBarcode;
    int            iBC_MaxHeight;
    int            iBC_MinHeight;
    int            iBC_MaxNoVal;
    int            iBC_Tolerance;
    int            iLaengeVon;
    int            iLaengeBis;
    PBarcodeData2  pBC_NextBarcodeData2;
```

```
} BarcodeData2;


typedef struct BarcodeResult2_tag
{
    int         iBC_Type;
    int         iBC_Type2;
    int         iBC_Status;
    char        szBC_Barcode[64];
    int         iBC_StartX;
    int         iBC_StartY;
    int         iBC_SizeX;
    int         iBC_SizeY;
    int         iBC_Orientation;
    TwoDimResult      BC_PDFRes;
} BarcodeResult2;


typedef struct tag_RotInfo
{
    double      dBC_cos;
    double      dBC_sin;
    int         iBC_BMoffsetX;
    int         iBC_BMoffsetY;
    BOOL        fBC_bRotated;
    int         iBC_offsetX;
    int         iBC_offsetY;
    double      dBC_XKorr;
    double      dBC_YKorr;
} RotInfo;


typedef struct TwoDimResult_tag
{
    HANDLE      hBC_TwoDimRes;
    int         iBC_TwoDimLen;
    int         iBC_TwoDimRows;
    int         iBC_TwoDimCols;
    int         iBC_PdfECL;
} TwoDimResult;


typedef struct AdvancedMode_tag
{
    int               AdvancedSearch;
    int               DynamicThreshold;
    int               iBC_Threshold;
    int               iBC_RemovePixel;
    int               iBC_LightMargin1;
    int               iBC_LightMargin2;
    int               iBC_LightMargin3;
    int               iBC_ScanDistance2;
    int               iBC_ScanDistBarcode2;
} AdvancedMode;
```

7.4 Redistributable Files

To integrate QS-Barcode Library in its form as p_lib interface, you must integrate the `QSBarLib.lib` into your application.
It might be necessary to install to "Microsoft Visual C++ 2013 Redistributable " package. Download the latest version at the Microsoft webpage.

You also need to distribute your **license file** (`QSBC.lic`) with your application and install it in the application path.

A **QS Barcode runtime license** is required for each workstation where users use applications with the `QSBarLib.lib` integrated.

With the runtime license, you must deliver the following files with your application:

- **qsbLog.dll**
- **GraphLib.DLL**
- **Graph_eng.DLL**
- **qsImgImp.dll**

8 DLL with Handle (h_dll)

8.1 Interface

This DLL reads the barcodes from the passed image section (or the whole image) using a handle:

```
int QSReadBarcode2(HANDLE hDIB, Barcode2 *pBarcode2,
                   int iNumResults);
int QSReadBarcode3(HANDLE hDIB, Barcode2 *pBarcode2,
                   int iNumResults, char *szIniFile);
int QSReadBarcode4(HANDLE hDIB, Barcode2 *pBarcode2,
                   int iNumResults, char *szIniFile, char
                   *szLicenseFile);
int QSReadBarcode5(HANDLE hDIB, Barcode2 *pBarcode2,
                   int iNumResults, char *szIniFile, char
                   *szLicenseFile, AdvancedMode *pBC_AdvancedMode);

int QSReadBarcode6(HANDLE hDIB, Barcode2 *pBarcode2,
                    int iNumResults, char *szIniFile,
                    char *szLicenseFile, AdvancedMode *pBC_AdvancedMode,
                    char *szImaging);
int QSGetNextBarResult2(BarcodeResult2 *pBarcodeResult2);
int QSFreeBarResult2(void);
```

Due to the setup of `QSReadBarcode2` the indicated image barcodes are read and saved in an internal structure. This way a maximum of `iNumResults` results are saved. If the barcode recognition is error-free and barcodes were found, the return value of the function is `BC_OK`. If no barcode could be found, the return value is `BC_NO_BARCODE`. (For additional errors see header file).
If you use `QSReadBarcode3` you can additionally pass a "qsbc.ini" to the function. (See chapter 16.4 Image enhancement via "qsbc.ini")

If you use `QSReadBarcode4` you can additionally pass a license file "qsbc.lic" to the function. The path to the license file has to be fully qualified. If you pass the "qsbc.ini" as an empty string when using `QSBarcode4,` this parameter is ignored.

If you use `QSReadBarcode5` you can additionally pass an AdvancedMode structure to the function for starting an "Advanced Search". (See chapter 12 Advanced Search) The path to the license file has to be fully qualified. If you pass the "qsbc.ini" as an empty string when using `QSBarcode5,` this parameter is ignored.

If you use `QSReadBarcode6` you can additionally pass a string with semicolon separated commands for image enhancement. (See chapter 13 Image Processing)

To receive the **barcode results** in a loop, the function `QSGetNextBarResult2` has to be called. As long as this function returns `BC_OK`, barcodes still exist. For every recognized barcode a `BarcodeResult2`-structure is set up.
At the end of the recognition the function `QSFreeResult2` **must** be called always in order to release internally used memory.
Before performing a second search, the results must be read from the function `QSReadBarcode2,` otherwise they will be lost.

The **function parameter** `pBarcode2` and `pBC_AdvancedMode` are structures that are

defined as described in the "Definitions" chapter.

The structures `BarcodeResult2` and `TwoDimResult` correspond to those of the library version that were described in the "Definitions" chapter.

To determine your **license** type use the function

```
int QSLicense( void );
```

or

```
int QSLicense1( char *szLicenseFile );
```

by directly passing the fully qualified path of the license file.
These functions return a combination of the following constants: `BC_LIC_DEMO`, `BC_LIC_LINEAR`, `BC_LIC_PDF417`, `BC_LIC_DATAM`, `BC_LIC_QR_CODE`, `BC_LIC_AZTEC`

Integration Examples
This example (which cannot be compiled!) shows the integration of the library into the code. For example files that can be compiled see "C-Example program".

```
int    iReturn,nResult,iNumResults;
HANDLE hDIB;
Barcode2 pBarcode2;
BarcodeResult2 pBarcodeResult2;

//hDIB = Handle on the bits of the image
if(hDIB==NULL)
    return 0;
// At first empty structure
memset(&pBarcode2, 0, sizeof(Barcode2));
memset(&pBarcodeResult2, 0, sizeof(BarcodeResult2));
// Fill in the barcode-structure with suitable values.
pBarcode2.iBC_Type = BC_INTERLEAVED25 | BC_CODE39;
pBarcode2.iBC_Type2 = BC_NONE;
pBarcode2.iBC_Length = 0;
pBarcode2.iBC_Checksum = 0;
pBarcode2.iBC_Orientation = BC_0 | BC_90 | BC_180 | BC_270;
pBarcode2.iBC_ReadMultiple = 1;
pBarcode2.iBC_LightMargin = 18;
pBarcode2.iBC_ScanDistance = 2;
pBarcode2.iBC_Percent = 100;
pBarcode2.iBC_ScanDistBarcode = 1;
pBarcode2.iBC_MaxHeight = 400;
pBarcode2.iBC_MinHeight = 15;
pBarcode2.iBC_MaxNoVal = 10;
pBarcode2.iBC_Tolerance = 10;
pBarcode2.iBC_StartX = 0;
pBarcode2.iBC_StartY = 0;
pBarcode2.iBC_SizeX = 0;
pBarcode2.iBC_SizeY = 0;
pBarcode2.iLaengeVon = 4;
pBarcode2.iLaengeBis = 12;
iNumResults=10;
```

```
// Now we can start the search
iReturn = QSReadBarcode2(hDIB, &pBarcode2, iNumResults);
//put out results
for(nResult=0;nResult < iNumResults;nResult++) {
    iReturn = QSGetNextBarResult2(&pBarcodeResult2);
    if(iReturn!=BC_NO_BARCODE)
        printf("%i. result: %s\n",
               nResult, pBarcodeResult2.szBC_Barcode);
    else
        break;
}
// Do not forget to free the results!
GlobalUnlock(hDIB);
GlobalFree(hDIB);
iReturn = QSFreeBarResult2();
```

8.2 Definitions

```
typedef struct Barcode2_tag
{
    int iBC_Type;
    int iBC_Type2;
    int iiBC_Length;
    int iBC_Checksum;
    int iBC_Orientation;
    int iBC_ReadMultiple;
    int iBC_LightMargin;
    int iBC_ScanDistance;
    int iBC_Percent;
    int iBC_ScanDistBarcode;
    int iBC_MaxHeight;
    int iBC_MinHeight;
    int iBC_MaxNoVal
    int iBC_Tolerance
    int iBC_StartX;
    int iBC_StartY;
    int iBC_SizeX;
    int iBC_SizeY;
    int iLaengeVon;
    int iLaengeBis;
} Barcode2;
```

The structures `BarcodeResult2, TwoDimResult` and `AdvancedMode` correspond to those of the library version and have already been described in the "Definitions" chapter.

8.3 Redistributable Files

It might be necessary to install to "Microsoft Visual C++ 2013 Redistributable " package. Download the latest version at the Microsoft webpage.

They must be located either in the application path or in the path, where the system DLLs are located.

You must also distribute your **license file** (`QSBC.lic`) with your application and install them in the application path.

You must purchase a **runtime license** for each workplace running your application.

With the runtime license, you must deliver the following files with your application:

- **QSBarDll_h.DLL**
- **GraphLib.DLL**
- **Graph_eng.DLL**
- **QSBImage.DLL**
- **FreeImage.dll**
- **qsImgImp.dll**
- **qsbLog.dll**

9 DLL with File (f_dll)

9.1 Interface

This DLL reads barcodes from a file specified by names:

```
int QSReadBarcode2(char *szImageName, Barcode2 *pBarcode2,
                   int iNumResults);
int QSReadBarcode3(char *szImageName, Barcode2 *pBarcode2,
                   int iNumResults, char *szIniFile);
int QSReadBarcode4(char *szImageName, Barcode2 *pBarcode2,
                   int iNumResults, char *szIniFile, char
                   *szIniFile);
int QSReadBarcode5(char *szImageName, Barcode2 *pBarcode2,
                    int iNumResults, char *szIniFile, char
                    *szLicenseFile, AdvancedMode *pBC_AdvancedMode);

int QSReadBarcode6(char *szImageName, Barcode2 *pBarcode2,
                    int iNumResults, char *szIniFile, char
                    *szLicenseFile, AdvancedMode *pBC_AdvancedMode,
                    char *szImaging);
int QSGetNextBarResult2(BarcodeResult2 *pBarcodeResult2);
int QSFreeBarResult2(void);
```

Due to the setup of `QSReadBarcode2` the indicated image barcodes are read and saved in an internal structure. This way a maximum number of `iNumResults` results are saved. If the barcode recognition is error-free and barcodes were found the return value of the function is `BC_OK`. If no barcode could be found the return value is `BC_NO_BARCODE`. (For additional errors see header file). If you use `QSReadBarcode3` you can additionally pass a "qsbc.ini" to the function. (See chapter 16.4 Image enhancement via "qsbc.ini")
If you use `QSReadBarcode4` you can additionally pass a license file "qsbc.lic" to the function. The path to the license file has to be fully qualified. If you pass the "qsbc.ini" as an empty string when using `QSBarcode4`, this parameter is ignored.
If you use `QSReadBarcode5` you can additionally pass an AdvancedMode structure to the function for starting an "Advanced Search". (See chapter 12 Advanced Search) The path to the license file has to be fully qualified. If you pass the "qsbc.ini" as an empty string when using `QSBarcode5`, this parameter is ignored.
If you use `QSReadBarcode6` you can additionally pass a string with semicolon separated commands for image enhancement. (See chapter 13 Image Processing)

Hint	**Adobe PDF Documents** Please have a look at Appendix "16.3Adobe PDF Documents – Special Settings" **MultiPage Files** Accessing all pages of a MultiPage File (MultiTiff or Adobe PDF document) is possible. See Appendix "16.2 MultiPage support"

To receive the **barcode results** in a loop, the function `QSGetNextBarResult2` is called. As long as this function returns `BC_OK`, barcodes still exist. For every recognized barcode a `BarcodeResult2`-structure is set up.
At the end of the recognition the function `QSFreeResult2` **must** be called in order to release the internal used memory.

Before performing a second set up, the results must be read from the function `QSReadBarcode2`, otherwise they will be lost.
The **function parameters** `pBarcode2` and `pBC_AdvancedMode` are structures that are defined as described in the "Definitions" chapter.
The structures `BarcodeResult2` and `TwoDimResult` correspond to those of the library version and have already been described in the "Definitions" chapter.

To determine your **license** type use the function

```
int QSLicense( void );
```

or

```
int QSLicense1( char *szLicenseFile );
```

by directly passing the fully qualified path of the license file.

These functions return a combination of the following constants: `BC_LIC_DEMO`, `BC_LIC_LINEAR`, `BC_LIC_PDF417`, `BC_LIC_DATAM`, `BC_LIC_QR_CODE`, `BC_LIC_AZTEC`

9.2 Integration Example

This example (which has been shortened and cannot be compiled!) is part of the Visual Basic example program.
Declare functions:

```
Declare Function QSReadBarcode2 Lib "QSBarDll.F_Dll" _
      (ByVal szImageName As String, pBarcode2 As WU_Barcode2, _
       ByVal iNumResults As Long) As Long
Declare Function QSGetNextBarResult2 Lib "QSBarDll.F_Dll" _
      (pBarcodeResult2 As WU_BarcodeResult2) As Long
Declare Function QSFreeBarResult2 Lib "QSBarDll.F_Dll" () As Long
```

Structures and constants (Visual Basic Notation of the QSBarLib.h)

```
'TwoDim_Result
Type WU_TwoDimResult
    hBC_TwoDimRes As Long
'...
    iBC_PdfECL As Long
End Type

'  Type WU_Barcode2
Type WU_Barcode2
    iBC_Type As Long
    iBC_Type2 As Long
    iBC_Length As Long
'...
    iLaengeBis As Long
End Type
' Type WU_BarcodeResult2
Type WU_BarcodeResult2
    iBC_Type As Long
    iBC_Type2 As Long
    iBC_Status As Long
'...
    sBC_TwoDimRes As WU_TwoDimResult
End Type

'*** Defaultvalues
Global Const BC_LIGHTMARGIN = 30
Global Const BC_SCANDISTANCE = 15
'...

'*** Barcodetypes
Global Const BC_NONE = 0
Global Const BC_INTERLEAVED25 = 1
'...
Global Const BC_ALL = &H67F 'all except PDF417 and Data Matrix

'*** Checksumtypes
Global Const BC_CHECKNONE = 0
Global Const BC_MOD10 = 1
'...
```

```
'*** Orientations
Global Const BC_0 = 1
Global Const BC_90 = 2
'...

'*** Errorcodes
Global Const BC_OK = 0
Global Const BC_NO_BARCODE = 1
'...
Global Const BC_INVALID_BMP = &H10
Global Const BC_ERROR = -1
```

Set up of the DLL-function (limited to the basics)

```
Function TestBARCODERead(BMPName As String, newName As String,
                         BCBarcode2 As WU_Barcode2) As String
Dim iNumResults As Integer
Dim BCResult2 As WU_BarcodeResult2
Dim BCResultNr As Byte
Dim a As Integer

'Define values before calling up function
    BCBarcode2.iBC_ReadMultiple = 2
    BCBarcode2.iBC_LightMargin = 24
    '...
    BCResult2.iBC_SizeY = 0
    BCResult2.iBC_Orientation = 0

    iNumResults=10
    'Calling uo with image name, barcode-Parameters and wished
    'results.
    QSReadBarcode2 BMPName, BCBarcode2, iNumResults
    'Check results (in this example only a single one)
    QSGetNextBarResult2 BCResult2

    'If no barcodes result has been read
    If QSTrim(BCResult2.szBC_Barcode) = "" Then
        TestBARCODERead = 1
        newName = ""
    Else 'otherwise give back result in newName
        TestBARCODERead = 0
        newName = QSTrim(BCResult2.szBC_Barcode)
    End If
    'IMPORTANT: set free results.
    rc = QSFreeBarResult2()

End Function
```

9.3 Definitions

The structure `Barcode2_tag` corresponds to the structure of the H_Dll and has been described in "Definitions" chapter. The structures `BarcodeResult` and `TwoDimResult` correspond to those of the library version and have been described in "Definitions" chapter.

9.4 Redistributable Files

The runtime license allows you to deliver the following files:

- **QSBARDLL_F.DLL**
- **GraphLib.DLL**
- **Graph_eng.DLL**
- **QSBImage.DLL**
- **FreeImage.dll**
- **qsImgImp.dll**
- **qsbLog.dll**

If you want to work with Adobe PDF Documents

- **pstilldll.dll / pstilldll64.dll**

It might be necessary to install the "Microsoft Visual C++ 2013 Redistributable " package. Download the latest version at the Microsoft webpage.

The files must be located either in the application path or in another path where the system locates DLLs.

You must also distribute your **license file** (`QSBC.lic`) with your application and install it in the application path.

You must purchase a **runtime license** for each workplace running your application .

10 ActiveX with File (f_ocx)

10.1 Description

In a parameter block, an image file name is transferred to the OCX. An image section may be specified if barcode recognition must be performed on part of the image only. In addition, parameters are set up to specify the type of barcode search.
You receive a result value (barcode found/not found) as a return value and one or several result structures according to the result value. In case of error the result value displays the error in detail. The result structures also display errors during the recognition stage.

Hint	**Adobe PDF Documents** Please have a look at Appendix 16.3 Adobe PDF Documents – Special Settings **MultiPage Files** Accessing all pages of a MultiPage File (MultiTiff or Adobe PDF document) is possible. See Appendix "14.2 MultiPage support"

10.2 Integration

On the developer's workstation where you installed "QS-Barcode SDK" the QS-Barcode Active X is already installed. For the runtime-workstations we supply a separate setup for "QS-Barcode ActiveX" (f_ocx\x86\setup).

Please notice that the setup registers the control, the VisualBasic Runtime environment and copies the required **DLLs**:

- **QSBARDLL_F.DLL**
- **GraphLib.DLL**
- **Graph_eng.DLL**
- **QSBImage.DLL**
- **FreeImage.dll**
- **qsImgImp.dll**
- **qsbLog.dll**

If you want to work with Adobe PDF Documents

- **pstilldll.dll**

You should install the OCX and the required **DLLs** into the system directory to avoid errors.

In case of problems, registration of ActiveX controls may be done manually using the command

```
regsvr32.exe qsbcocx.ocx
```

10.3 Methods

The setup is performed using the function

```
rcBC = QSBCOCX1.getBarcodeResult2()
```

This method can be performed as long as there are no more results.
Following recognition of the result parameter the method

```
rcBC = QSBCOCX1.freeBarcodeResult2()
```

must be performed to release the memory used internally by the OCX.

Use the following method to require the description of barcode type names and error reports:

```
QSBCOCX1.getTypeName(QSBCOCX1.ResultType)
QSBCOCX1.getTypeName2(QSBCOCX1.ResultType2)
QSBCOCX1.getOrientationName(QSBCOCX1.ResultOrientation)
QSBCOCX1.getErrorName(rcBC).
```

The property `License` shows you with a combination of the constants `BC_LIC_DEMO, BC_LIC_LINEAR, BC_LIC_PDF417, BC_LIC_DATAM, BC_LIC_QR_CODE, BC_LIC_AZTEC` depending on the **License version** you are using. The function `getLicenseName` returns a text version of the license version.

Special method

The method `fillResultBinary()` handles a special issue: The OCX property `ResultBarcode` which is an ASCII string might be converted to an Unicode string in some system environments. In this case, non-printable characters and characters out of the 7-bit region will be converted codepage based.
For 2D barcodes with binary output data, it might be important go get the un-converted un-changed barcode content.
Use `fillResultBinary()` to solve this (example is in Visual Basic).

```
Dim b() As Byte
Dim rc as Long
[..]
rcBC = QSBCOCX1.freeBarcodeResult2()
[..]
ReDim b(QSBCOCX1.ResultLength)
rc = QSBCOCX1.fillResultBinary(b, QSBCOCX1.ResultLength)
[..]
```

The byte array `b` is filled with the barcode data byte by byte by the method.
The return code of the method is <0 in case of error, otherwise >=0 where the value indicates the number of bytes filled in the array.

10.4 Call Properties

The control prepares **properties** for specification of the searched barcode. They should be defined before the start of the recognition. An exact specification of type and length leads to a fast and reliable recognition of the barcode. It is also possible to search for unknown types (`BC_ALL`) and unknown lengths (`0`).

Example code to set up parameters in the code:

```
QSBCOCX1.PictureName = "c:\Qsbcocx\quelle\BarTest.tif"
QSBCOCX1.iNumResults = 10
QSBCOCX1.BarcodeType = BC_ALL
QSBCOCX1.BarcodeType2 = BC_NONE
QSBCOCX1.BarcodeChecksum = BC_CHECKNONE
QSBCOCX1.BarcodeOrientation = BC_ALL_ORI
QSBCOCX1.BarcodeLength = 0
QSBCOCX1.BarcodeLength_from = 0
QSBCOCX1.BarcodeLength_to = 0

QSBCOCX1.LightMargin = BC_DEFAULTVALUES_TYPES.BC_LIGHTMARGIN
QSBCOCX1.ReadMultiple = BC_DEFAULTVALUES_TYPES.BC_READMULTIPLE
QSBCOCX1.ScanDistance = BC_DEFAULTVALUES_TYPES.BC_SCANDISTANCE
QSBCOCX1.ScanDistBarcode = BC_DEFAULTVALUES_TYPES.BC_SCANDISTBAR
QSBCOCX1.MaxHeight = BC_DEFAULTVALUES_TYPES.BC_MAXHEIGHT
QSBCOCX1.MinHeight = BC_DEFAULTVALUES_TYPES.BC_MINHEIGHT
QSBCOCX1.MaxNoVal = BC_DEFAULTVALUES_TYPES.BC_MAXNOVAL
QSBCOCX1.Tolerance = BC_DEFAULTVALUES_TYPES.BC_TOLERANCE

QSBCOCX1.DynamicThreshold = 1

QSBCOCX1.Percent = 100
QSBCOCX1.StartX = 0
QSBCOCX1.StartY = 0
QSBCOCX1.SizeX = 0
QSBCOCX1.SizeY = 0
```

Barcode-Properties in detail:
The property names differ slightly from those of the library and DLL. The properties are described in detail in "Barcode-Parameters" chapter.

Property name	Corresponds to
PictureName	Name of image file (*.tif) to be recognized incl. path name
iNumResults	Number of maximal results
BarcodeType	iBC_Type
BarcodeType2	iBC_Type2
BarcodeChecksum	iBC_Checksum
BarcodeOrientation	iBC_Orientation
BarcodeLength	iBC_Length
BarcodeLength_from	iLaengeVon
BarcodeLength_to	iLaengeBis
LightMargin	iBC_LightMargin
ReadMultiple	iBC_ReadMultiple
ScanDistance	iBC_ScanDistance
ScanDistBarcode	iBC_ScanDistBarcode
MinHeight / MaxHeight	iBC_MinHeight / iBC_MaxHeight
Percent	iBC_Percent
MaxNoVal	iBC_MaxNoVal
Tolerance	iBC_Tolerance
StartX	iBC_StartX
StartY	iBC_StartY
SizeX	iBC_SizeX
SizeY	iBC_SizeY
AdvancedSearch	AdvancedSearch
DynamicThreshold	DynamicThreshold
Threshold	iBC_Threshold
RemovePixel	iBC_RemovePixel
LightMargin1	iBC_LightMargin1
LightMargin2	iBC_LightMargin2
LightMargin3	iBC_LightMargin3
ScanDistance2	iBC_ScanDistance2
ScanDistBarcode2	iBC_ScanDistBarcode2

10.5 Result Properties

Further **properties** form the result type of one or several barcodes.
The initialization of the result values is not necessary, they are automatically overwritten with the results.
Here again, the property names differ slightly from those of the library and DLL. The properties are described in detail in the "Return Values" chapter.

Barcode Result Properties in detail:

Property Name	Corresponds to
ResultType	iBC_Type
ResultType2	iBC_Type2
ResultOrientation	iBC_Orientation

ResultBarcode	szBC_Barcode (for 2dim. barcodes: hBC_TwoDimRes)
ResultStatus	iBC_Status
ResultStartX	iBC_StartX
ResultStartY	iBC_StartY
ResultSizeX	iBC_SizeX
ResultSizeY	iBC_SizeY
Result2dCols	iBC_TwoDimCols (only filled for 2dim barcodes)
Result2dRows	iBC_TwoDimRows (only filled for 2dim barcodes)
ResultPdfECL	iBC_PdfECL (only filled for PDF417)
ResultLength	length of ResultBarcode in bytes (for 2dim barcodes: iBC_TwoDimLen)

Special Properties

As the method `fillResultBinary()` does, the two properties `ResultBarcode_Base64` and `ResultLength_Base64` offer an alternative way to read out the barcode result data. The property `ResultBarcode_Base64` contains the barcode data encoded in the Base64 format (see Wikipedia or other sources for details). `ResultLength_Base64` contains the length for this Base64 encoded result data.

10.6 Short Example Application

```
Public Sub ReadBCBtn_Click()
' short demonstration for using QSBCOCX
Dim rcBC As BC_ERROR_TYPES
Dim BCResultNr As Long

  rcBC = BC_OK

  'some settings for Barcode
  QSBCOCX1.PictureName = "c:\Qsbcocx\quelle\BarTest.tif"
  QSBCOCX1.BarcodeType = BC_ALL
  QSBCOCX1.BarcodeType2 = BC_NONE
  QSBCOCX1.BarcodeChecksum = BC_CHECKNONE
  QSBCOCX1.BarcodeOrientation = BC_ALL_ORI
  QSBCOCX1.BarcodeLength = 0
  QSBCOCX1.BarcodeLength_from = 0
  QSBCOCX1.BarcodeLength_to = 0
  QSBCOCX1.iNumResults = 10
    'get the first result
  rcBC = QSBCOCX1.getBarcodeResult2

  While rcBC = BC_OK
    BCResultNr = BCResultNr + 1
    With frmTest.ListErgebnisse
      .AddItem BCResultNr & ". Result: "
      .AddItem "Type = " & QSBCOCX1.getTypeName(QSBCOCX1.ResultType)
      .AddItem "Type2 = " &
                          QSBCOCX1.getTypeName2(QSBCOCX1.ResultType2)
      .AddItem "Result = " & QSBCOCX1.ResultBarcode
```

```
      .AddItem "Orientation = " & QSBCOCX1.ResultOrientation
      .AddItem "Status = " & QSBCOCX1.ResultStatus
      .AddItem "StartX = " & QSBCOCX1.ResultStartX
      .AddItem "StartY = " & QSBCOCX1.ResultStartY
      .AddItem "SizeX = " & QSBCOCX1.ResultSizeX
      .AddItem "SizeY = " & QSBCOCX1.ResultSizeY
      .AddItem ""
    End With
    'get the following results
    rcBC = QSBCOCX1.getBarcodeResult2
  Wend
  MsgBox "Results stopped with " & QSBCOCX1.getErrorName(rcBC)
  'free the memory
  rcBC = QSBCOCX1.freeBarcodeResult2
  MsgBox "Free stopped with " & QSBCOCX1.getErrorName(rcBC)
End Sub
```

10.7 Redistributable Files

The following files can be delivered using the runtime license:

- **QSBCOCX.OCX**
- **QSBARDLL_F.DLL**
- **GraphLib.DLL**
- **Graph_eng.DLL**
- **QSBImage.DLL**
- **FreeImage.dll**
- **qsImgImp.dll**
- **qsbLog.dll**

If you want to work with Adobe PDF Documents

- **pstilldll.dll / pstilldll64.dll**

It might be necessary to install to "Microsoft Visual C++ 2008 Redistributable (SP1)" package. Download the latest version at the Microsoft webpage.

The DLLs must be located in the application path or the path where the system locates DLLs.

You must also distribute your **license file** (`QSBC.lic`) with your application and install it in the application path.

You must **register** the QSBCOCX.ocx , e.g. with the command `regsvr32.exe qsbcocx.ocx`, or use the supplied setup (`F_ocx\x86\setup`) for the ocx file.

You must purchase a **runtime license** for each workplace running your application.

11 Parameters

This section provides an overview of the available parameters and return values. In correspondence to each version, not all parameters may be available, or they are available in different parameter structures. For more see the corresponding chapter on the version used.

Parameters are divided according to **barcode parameters** and **return values** and are sorted in **alphabetical order** within these chapters.

11.1 Barcode-Parameters

11.1.1 BC_RotInfo

Is a structure with rotation information and should be initialized with 0, except for `dBC_cos` which has to be initialized with 1.0.

(p_lib: BarcodeParam2_tag)

11.1.2 hBC_ErrorFile

Ignore this parameter. It is no longer in use. Initialize it with NULL.

(p_lib: BarcodeParam2_tag)

11.1.3 iBC_Checksum

This defines whether checksum is used and identifies the one used. You may use: `BC_NONE`, `BC_MOD10`, `BC_MOD10_EXT`, `BC_MOD43`, `BC_EXISTENCE` and `BC_REPORT_CHECKSUM`.

With these settings optional checksums are activated. Some barcode types (e. g. Code 128) always include an internal checksum, which is specified in the specification of the code. These checksums cannot be switched off.

For other codes the checksum is activated by `BC_MOD10`, `BC_MOD10EXT` or `BC_MOD43`. The last character in the barcode is interpreted as the checksum.

Barcodes with wrong checksums will be ignored completely. The function return code is `BC_NO_BARCODE`.

When the checksum is correct, the result of the barcode is reported, normally without the checksum.

To get the check character reported, you have to set `BC_REPORT_CHECKSUM`.

Note: If you set `iBC_length` exactly and set a checksum with iBC_Checksum, please specify the length including the checksum character.

A special case is `BC_EXISTENCE`, which can be set together with the other value by or-operator.

`BC_EXISTENCE` activates the check for "barcode suspicion". For controlling this check, `iBC_Percent` is used (for details look there).

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: BarcodeChecksum)

11.1.4 iBC_Debug

Ignore this parameter. It is no longer in use. Initialize it with 0.

(p_lib: BarcodeParam2_tag)

11.1.5 iBC_Height

Height of the image in pixels.

(p_lib: BarcodeParam2_tag)

11.1.6 iBC_Length

This defines the length of the single barcodes, that is the count of characters in the barcode. It is also possible to pass on 0. In this case the interval from iLaengeVon, that is (iLengthFrom) to iLaengeBis, that is (iLengthTo), is used. If these parameters are set to zero as well, the length is calculated automatically with setting lengths from 4 to 64.

The maximum value for iBC_length is 64, if the barcode contains less than 4 characters iLaengeVon must be set explicit.

Note: If you set iBC_length exactly and set a checksum with iBC_Checksum, please specify the length including the checksum character.

This parameter is irrelevant for 2D barcodes!

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: BarcodeLength)

11.1.7 iBC_LightMargin

This corresponds to the light margin around the barcode.
Default value is 18 pixel or BC_LIGHTMARGIN.

The light margin should not be too small, so that the "blanks" in the barcode are not erroneously interpreted as the light margin.

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: LightMargin)

11.1.8 iBC_MaxHeight

A maximal height of the searched barcode is specified in pixel. Default value for the maximal height of linear barcodes is 400 pixel or `BC_MAXHEIGHT`.

For 2D barcodes, the defaults have other values.

If BC_EXISTENCE is activated the parameter is used to suppress lines which are longer.

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: MaxHeight)

11.1.9 iBC_MaxNoVal

Defines after how many lines where no barcode was detected the barcode is marked "finished".

Default value: 10

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: MaxNoVal)

11.1.10 iBC_MinHeight

A minimal height of the searched barcode is specified in pixel. Default value for the minimal height for linear barcodes is 15 pixel or `BC_MINHEIGHT`.

For 2D barcodes, the defaults have other values.

If BC_EXISTENCE is activated the parameter is used to suppress lines which are shorter.

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: MinHeight)

11.1.11 iBC_Orientation

You can define the orientation of the barcode by combining the following constants with OR: `BC_0`, `BC_90`, `BC_180`, `BC_270`. These orientation values are valid for linear barcodes and PDF417 barcodes only. Other barcode types ignore these settings.

If the barcode is heavily skewed, you can add additional scan directions by using the following constants: `BC_SKEW_LIGHT` (13°), `BC_SKEW_MEDIUM` (26°) und `BC_SKEW_HEAVY`(39°).
Attention: Using `BC_SKEW_HEAVY` does NOT include the lower degrees. To check all rotations, combine them with OR.
Barcodes that are very low could slip through the check. Use the constant `BC_SKEW_DENSE_SEARCH` to add another scan between the other orientations.
All of these skew values are valid for linear barcodes only. Other barcode types ignore these settings.

For DataMatrix codes you can set DM_INTENSIVE_SEARCH to force QS-Barcode to try some variations while searching for DataMatrix (which of course takes some time). This is the only valid setting for iBC_Orientation, if you do not want to use "intensive search", set iBC_Orientation to 0. You might set `BC_0`, `BC_90`, `BC_180`, `BC_270`, it does not hurt but does not have any influence.
Have a look at chapter "Special settings for Data Matrix Code" for more details.

Please keep in mind that the more orientations and skew options you set, the longer the barcode search takes! Only use the options you really need!

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: BarcodeOrientation)

11.1.12 iBC_Percent

In some cases barcodes on images are not readable. Setting iBC_Checksum to BC_EXISTENCE tells the recognition engine to check for "barcode suspicion".

For linear barcodes, the sensitivity can be controlled by different values of iBC_Percent (the parameter is irrelevant for 2D barcodes). The "suspicion" return

code is set by the recognition if "enough percent" of the calculated bar are found. Let us have an example. If you use barcode type code 39 and have 8 characters in the barcode, the correct code has 50 bars.

If QS-Barcode finds only 44 lines, these are 88 percent. If iBC_Percent is lower than this value, BC_Exists will be set in iBC_Status in the result structure.

Note: With iBC_MinHeight and iBC_MaxHeight too short and too long lines will be ignored.

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: Percent)

11.1.13 iBC_ReadMultiple

You can define if one or more barcodes shall be recognized on each image. In most cases BC_MULTI is used to read one or more barcodes. The defines are in the MultiRead section:
`BC_ONE`: One barcode is searched, only a unique value for the field is reported. If more than one barcode is found none will be reported!
`BC_ONE_BREAK`: One barcode is searched. After one barcode was found, the routine stops and does not search for further barcodes. There is no check for other barcodes. The iBC_MinHeight is not checked either.
his option will get the fastest results, but it should only be used if type and length are known and in combination with checksum to provide erroneous reading.
`BC_MULTI`: Multiple barcodes are searched; each different value is reported once.
`BC_MULTI_ONE`: Multiple barcodes are searched, only one value for each barcode is reported, only unique values are reported.
`BC_MULTI_BESTGUESS`: Multiple barcodes are searched, only one value for each barcode is reported; values that are clear or values that are most found are reported.
`BC_MULTI_MULTI`: Multiple barcodes are searched. Each value that is found is reported. If you have two barcodes with the same value these will be two results with the same value. This can be used to count barcodes of same values.
(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: ReadMultiple)

11.1.14 iBC_ResultCount

Indicates the number of result structures in `brBC_Result`.

(p_lib: BarcodeParam2_tag)

11.1.15 iBC_ScanDistance

This corresponds to the scan distance. This indicates that reading of the image in the y-direction is done incrementally to the height of the scan distance. Default value is 5 pixel or `BC_SCANDISTANCE`.

This value should be kept low if barcodes are poorly printed. Doing so may lead to a slower recognition, however.

Note: In programs like bcTester, this parameter is called "Searchdistance".

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: ScanDistance)

11.1.16 iBC_ScanDistBarcode

This corresponds to barcode scanning in the y-direction. Default value is 2 pixel or `BC_SCANDISTBAR`.

If the quality of the barcode is poor this value should be reduced, which slowes down the recogniton a little.

Note: In programs like bcTester, this parameter is called "Scandistance".

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: ScanDistBarcode)

11.1.17 iBC_SizeX

Width of search area for the barcode.

Can also be 0. If iBC_SizeX, iBC_SizeY, iBC_StartX and iBC_StartY are all set to 0, the whole image is taken.

(p_lib: BarcodeParam2_tag, f_dll: Barcode2_tag, f_ocx: SizeX)

11.1.18 iBC_SizeY

Height of search area for the barcode.

Can also be 0. If iBC_SizeX, iBC_SizeY, iBC_StartX and iBC_StartY are all set to 0, the whole image is taken.

(p_lib: BarcodeParam2_tag, f_dll: Barcode2_tag, f_ocx: SizeY)

11.1.19 iBC_StartX

x-coordinate of search area for the barcode.

Can also be 0. If iBC_SizeX, iBC_SizeY, iBC_StartX and iBC_StartY are all set to 0, the whole image is taken.

(p_lib: BarcodeParam2_tag, f_dll: Barcode2_tag, f_ocx: StartX)

11.1.20 iBC_StartY

y-coordinate of search area for the barcode.

Can also be 0. If iBC_SizeX, iBC_SizeY, iBC_StartX and iBC_StartY are all set to 0, the whole image is taken.

(p_lib: BarcodeParam2_tag, f_dll: Barcode2_tag, f_ocx: StartY)

11.1.21 iBC_Tolerance

Maximal distortion of the barcode (line tolerance).

Default value: 10

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: Tolerance)

11.1.22 iBC_Type

This defines the barcode type to be recognized. The constants contain:
BC_CODABAR, BC_CODE128, BC_CODE39, BC_CODE39EXT, BC_CODE93, BC_EAN128, BC_DATAMATRIX, BC_EAN8, BC_EAN13, BC_INDUSTRIE25, BC_INTERLEAVED25, BC_PDF417, BC_UPCA, BC_UPCE, BC_CODABLOCK, BC_25_IATA, BC_25_3MATRIX, BC_25_3DATALOGIC, BC_25_BCDMATRIX, BC_25_INVERTIERT, BC_CODE32, BC_25_INVERTIERT, BC_CODE32, BC_PHARMA, BC_CODE93EXT, BC_PATCHCODE, BC_QR_CODE, BC_AZTEC.
The types also might be linked with OR. This is not true with Patchcode, Pharmacode and 2D-Code. To read these codes separate function calls are necessary.
(The 2D barcode types PDF417, QR Code, Aztec Code and Data Matrix are not part of the standard delivery!)

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: BarcodeType)

11.1.23 iBC_Type2

This defines additional barcode types to be recognized. The constants contain:
BC_GS1_DATABAR_STACKED, BC_GS1_DATABAR_EXP, BC_GS1_DATABAR_EXP_STACKED, BC_GS1_DATABAR_LIMITED, BC_POSTNET_IMB, BC_POSTNET_ROYAL, BC_POSTNET_KIX, BC_POSTNET_AUSTRALIAN, BC_POSTNET_CPC, BC_POSTNET_ZIP, BC_POSTNET_PLANET.
The GS1_DATABAR types might be linked with OR, but the POSTNET types may not be combined!
If you have selected Patchcode, Pharmacode or a 2D-Code in parameter iBC_Type, iBC_Type2 should be BC_NONE as these codes need separate function calls and cannot be combined with other types.

If you are looking explicitly for an ECI code in QR Codes, enter BC_QR_CODE_ECI at iBC_Type2, otherwise BC_NONE.

(The 2D barcode types PDF417, QR Code, Aztec Code and Data Matrix are not part of the standard delivery!)

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: BarcodeType2)

11.1.24 iBC_Width

Width of the image in pixels.

(p_lib: BarcodeParam2_tag)

11.1.25 iLaengeBis (= iLengthTo)

Maximal possible length in characters when length of barcode is unknown.

This value can also be 0. If iLaengeVon, iLaengeBis and iBC_Length are set to 0, the length is calculated automatically.

This parameter is irrelevant for 2D barcodes!

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx BarcodeLength_to)

11.1.26 iLaengeVon (= iLengthFrom)

Minimal possible length in characters when length of barcode is unknown.

This value can also be 0. If iLaengeVon, iLaengeBis and iBC_Length are set to 0 the length is calculated automatically.

This parameter is irrelevant for 2D barcodes!

(p_lib: BarcodeData2_tag, f_dll: Barcode2_tag, f_ocx: BarcodeLength_from)

11.1.27 lBC_MemorySize

Use this to enter the amount of memory allocated in `pBC_Memory`.

(p_lib: BarcodeParam2_tag)

11.1.28 lpstrBC_Image

This is a pointer to the image bits. The image must be monochrome (the p_lib interface only supports monochrome images). Representation must be one bit per pixel. Bits must be 32-bit aligned and start with the top line of the image e.g. (0,0) is top left, black pixels are 0-bits, white pixels are 1-bits. 32-bit aligned means if you have an image width of 310 bits, each line in the image memory block needs to be filled up to 320 bit (next multiply of 32).
Black pixels are encoded as 0, white pixels as 1.

(p_lib: BarcodeParam2_tag)

11.1.29 pBC_BarcodeData

This is the pointer to the `BarcodeData`-structures which must be allocated properly for each barcode type searched.

(p_lib: BarcodeParam2_tag)

11.1.30 pBC_Memory

Results are written into this memory area allocated by the user. Results are accessed via the `brBC_Result`-pointer. The number of results in the memory area are stored in `iBC_ResultCount`.

(p_lib: BarcodeParam2_tag)

11.1.31 pBC_NextBarcodeData

This is the link to the next `BarcodeData`-block. A structure must be set up for each recognized type.

Two options are available to search for several barcodes in one field: Either a `BarcodeData2_tag`-block is filled, where the different barcode types are combined with OR and the length is set to the expected results (e. g. 4 to 11), or a `BarcodeData2_tag`-block is created for each barcode. This has the advantage that you can enter the exact length for each type, if known. This variation also can be performed with other parameters in the example orientation.

If the setting BC_Multi is selected, the types may also be combined with OR and the length can be entered in the interval. Then no additional `BarcodeData2_tag`-block has to be assigned and the pointer is 0. More detailed settings of the searched barcode lead to a more reliable recognition.

(p_lib: BarcodeData2_tag)

11.1.32 pbrBC_Result

This is a pointer to the result-structures.

(p_lib: BarcodeParam2_tag)

11.1.33 szBC_SubstitutionChar

As described in "szBC_SubstitutionString" below, this is the substitution character which is used for non-interpreted characters of the barcode.

(p_lib: BarcodeParam2_tag)

11.1.34 szBC_SubstitutionString

If the string is not 0, the corresponding string is returned if the barcode has not been read properly (in QS-Beleg e.g. "\bf" for barcode error). If the string is 0, all non-interpreted characters of the barcodes are returned with the character selected in `szBC_SubstitutionChar`. Here the default is '?'. Example: 12??567.

(p_lib: BarcodeParam2_tag)

11.1.35 szBC_Version

Ignore this parameter. It is no longer in use. Initialize it with 0 bytes.

To get the QS-Barcode SDK Version, use the function `QSVersion()` or check the contents of the OCX-Property "Version".

(p_lib: BarcodeParam2_tag)

11.2 Advanced Search parameters

11.2.1 AdvancedSearch

This is where the enhanced scan is activated and deactivated. 0=off, 1=on.
The advanced search offers the option of varying some of the following parameters during a barcode scan. This means that dirt pixels of differing width can be removed during a scan and up to three different light margins can be used. If no barcode was found, another scan can be started automatically which then scans the image with a narrower scan distance. iBC_ScanDistance2 and iBC_ScanDistBar2 are used for the new scan of the image.
Default value is 1 or `BC_ADVANCED_SEARCH`.

Attention: The "Advanced Search" is only applicable for linear barcodes.
(See chapter 12 Advanced Search)
(f_ocx: AdvancedSearch)

11.2.2 DynamicThreshold

Here you can set the dynamic threshold.
The threshold value is determined from the brightness values occurring in the image or from the passed detail of an 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.
0=off, 1=on.
Default value is 1 or `BC_DYNAMIC_THRESHOLD`.
(See chapter 12.2 DynamicThresholdDynamicThreshold.
(f_ocx: DynamicThreshold)

11.2.3 iBC_Threshold

If you do not require any dynamic determination of the threshold value, you can explicitly indicate the desired threshold value. The value for the threshold value lies in the range of 0-255. Our standard threshold value is 160. The lower the threshold value is, the more likely it is that a grey pixel is converted into a white pixel and the lighter the image overall.
Default value is 160 or `BC_THRESHOLD`.
(f_ocx: Threshold)

11.2.4 iBC_RemovePixel

Removal of white and black "dirt pixels" which are frequently caused by scanners of a poorer quality. The value indicates the pixel width which is removed within a line. The valid value range is 0-2.
Default value is 2 or `BC_REMOVE_PIXEL`.
(f_ocx: RemovePixel)

11.2.5 iBC_LightMargin1/iBC_LightMargin2/iBC_LightMargin3

In the advanced search, a total of up to three quiet zones can be indicated that are used for the barcode scan.
Default values are 30/45/15 or
`BC_LIGHTMARGIN1/BC_LIGHTMARGIN2/BC_LIGHTMARGIN3/`
(f_ocx: LightMargin1/ LightMargin2/ LightMargin3)

11.2.6 iBC_ScanDistance2

The advanced search offers the possibility to start a second run, if no barcodes have been found in the first run. The second run uses this value as initial search distance. If no second scan is to be executed, this value must be set to 0. (See chapter 11.1.15 iBC_ScanDistance)

If no barcodes were found in the first run, a second scan can automatically be started that scans the image with a narrower scan distance. iBC_ScanDistance2 and iBC_ScanDistBar2 are used for the new scan of the image. The existing values for the removal of dirt pixels and the quiet zones are also used in this scan.
If no second scan is to be executed, both values must be set to 0.

Default value is 5 or `BC_SCANDISTANCE2`.
(f_ocx: ScanDistance2)

11.2.7 iBC_ScanDistBar2

The second scan uses this value to scan the barcode. If no second scan is to be executed, this value must be set to 0. (See chapter 11.1.16 iBC_ScanDistBarcode)
Default value is 1 or `BC_SCANDISTBAR2`.
(f_ocx: ScanDistBar2)

11.3 Return Values

11.3.1 dBC_cos

This is the cos value of the rotation angle. If no routine is meant to follow, all parameters must be set to 0. Only the cos value must be set to 1.0.

This parameter mainly supports the compatibility to QS-Beleg.

(p_lib: tag_RotInfo)

11.3.2 dBC_sin

This is the sin value of the rotation angle.

This parameter mainly supports the compatibility to QS-Beleg.

(p_lib: tag_RotInfo)

11.3.3 dBC_XKorr

This is the x-correction factor. It is derived by dividing the actual width of the image and the target width of the image.

This parameter mainly supports the compatibility to QS-Beleg.

(p_lib: tag_RotInfo)

11.3.4 dBC_YKorr

This is the y-correction factor. It is derived by dividing the actual height of the image and the target height of the image.

This parameter mainly supports the compatibility to QS-Beleg.

(p_lib: tag_RotInfo)

11.3.5 fBC_bRotated

Indicates that the form has already been rotated. If this value is set to `TRUE`, set all other values to 0, except the cos value which should be set to 1.0.

This parameter mainly supports the compatibility to QS-Beleg.

11.3.6 hBC_TwoDimRes

Handle to the result of a 2D barcode. For a 2D barcode `szBC_Barcode` is always NULL.

(p_lib: TwoDimResult_tag, f_dll: TwoDimResult_tag, f_ocx: ResultBarcode)

11.3.7 iBC_BMoffsetX

This is the x-value of the rotation point in pixel.

This parameter mainly supports the compatibility to QS-Beleg.

(p_lib: tag_RotInfo)

11.3.8 iBC_BMoffsetY

This is the y-value of the rotation point in pixel.

This parameter mainly supports the compatibility to QS-Beleg.

(p_lib: tag_RotInfo)

11.3.9 iBC_TwoDimCols

Number of recognized columns in the barcode.

(p_lib: TwoDimResult_tag, f_dll: TwoDimResult_tag, f_ocx: Result2dCols)

11.3.10 iBC_PdfECL

Error correction level. Used only for PDF417.

(p_lib: TwoDimResult_tag, f_dll: TwoDimResult_tag, f_ocx: ResultPdfECL)

11.3.11 iBC_offsetX

This is the x-displacement of the image in pixel.

This parameter mainly supports the compatibility to QS-Beleg.

(p_lib: tag_RotInfo)

11.3.12 iBC_offsetY

This is the y-displacement of the image in pixel.

This parameter mainly supports the compatibility to QS-Beleg.

(p_lib: tag_RotInfo)

11.3.13 iBC_Orientation

Returns the orientation of the found barcode.

(p_lib: BarcodeResult2_tag, f_dll: BarcodeResult2_tag, f_ocx: ResultOrientation)

11.3.14 iBC_TwoDimLen

Length of the result in bytes.

(p_lib: TwoDimResult_tag, f_dll: TwoDimResult_tag, f_ocx: ResultLength)

11.3.15 iBC_TwoDimRows

Number of recognized rows in the barcode.

(p_lib: TwoDimResult_tag, f_dll: TwoDimResult_tag, f_ocx: Result2dRows)

11.3.16 iBC_SizeX

Width of the barcode found in pixel.

(p_lib: BarcodeResult2_tag, f_dll: BarcodeResult2_tag, f_ocx: ResultSizeX)

11.3.17 iBC_SizeY

Height of the barcode found in pixel.

(p_lib: BarcodeResult2_tag, f_dll: BarcodeResult2_tag, f_ocx: ResultSizeY)

11.3.18 iBC_StartX

x-coordinate of the barcode found in pixels (top left corner) in pixel.

(p_lib: BarcodeResult2_tag, f_dll: BarcodeResult2_tag, f_ocx: ResultStartX)

11.3.19 iBC_StartY

y-coordinate of the barcode found in pixels (top left corner)in pixel.

(p_lib: BarcodeResult2_tag, f_dll: BarcodeResult2_tag, f_ocx: ResultStartY)

11.3.20 iBC_Status

The status of the recognized barcode:
- `BC_OK` for clearly recognized barcodes
- `BC_GUESS` for the most often occurring value in the mode `BC_MULTI_BESTGUESS`
- `BC_EXISTS` if no value or existence verification could be found

(p_lib: BarcodeResult2_tag, f_dll: BarcodeResult2_tag, f_ocx: ResultStatus)

11.3.21 iBC_Type

Here the barcode type found is returned. If several types were searched, the recognized type(s) is/are displayed. .

(p_lib: BarcodeResult2_tag, f_dll: BarcodeResult2_tag, f_ocx: ResultType)

11.3.22 szBC_Barcode

This is the real result of the search. The barcode value is returned here.

(p_lib: BarcodeResult2_tag, f_dll: BarcodeResult2_tag, f_ocx: ResultBarcode)

12 Advanced Search

The advanced search (from QS-Barcode version 4.9.1 upwards) offers the option of varying different parameters during a barcode scan. If no barcode is found with this scan, another run is started automatically that scans the image with a narrower scan distance.

Attention: The "Advanced Search" is only applicable for linear barcodes

12.1 Parameters of the advanced Search

12.1.1 iBC_RemovePixel

A lot of images show black and white spots after the image production

On this scanned image

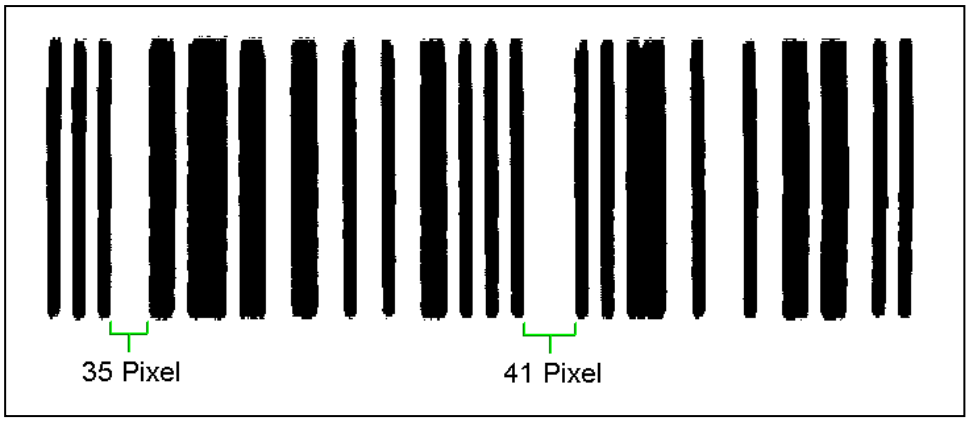
you can see many black dots between the bars.
During linear scanning, this results in additional transitions that prevent recognition of the barcode. Black dots between as well as white gaps in the barcode lines can be removed. The indication of the pixel width (1.2) always applies for white and black pixels in equal measure.
If this function is not to be used, the value is set to 0.

12.1.2 iBC_LightMargin1/iBC_LightMargin2/iBC_LightMargin3

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 again.

The light margin is measured 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 image has been scanned at 600 dpi. The high resolution leads to white gaps that are much wider than the default light margin value of 30 pixels. When increasing the light margin to 45 pixels, the barcode is recognized properly.



The next 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. Reducing the light margin to 15 pixels is necessary so that the end of the barcode can be recognized properly.

During a scan, barcodes can therefore be scanned with up to three different light margins at the same time. Barcodes of different light margins can thus be found in one document. In addition, this offers more flexibility when the light margins on the documents examined changes e.g. due to the barcode being affixed.
Light margins that are not required are deactivated with the value 0.

12.1.3 Automatic second run

If no barcodes were found in the first run, a second scan can automatically be started that scans the image with a narrower scan distance. **iBC_ScanDistance2** and **iBC_ScanDistBar2** are used for the new scan of the image. The existing values for the removal of dirt pixels and the light margins are also used in this scan.
If no second scan is to be executed, both values must be set to 0.

12.2 DynamicThreshold

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

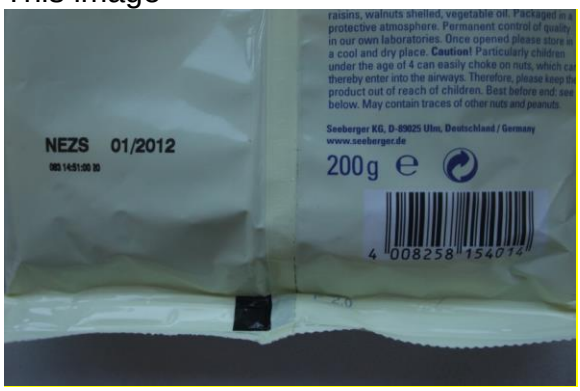
If scanned images are too bright or too dark, but are available as color or gray scale images, this may be corrected by the conversion (binarization) to a black and white image (monochrome).

For the internal conversion usually a threshold of 160 is used. This value has been found to be good in many cases. But if the image is too bright or too dark, a fixed threshold can lead to problems.

If images are too dark, the black bars of the barcode can e.g. merge so that recognition is no longer possible.

Here you now have the opportunity to let the threshold of the whole image be determined automatically. The threshold value is determined from the brightness values occurring in the image or from the passed detail of an 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.

This image



binarized with a threshold of 160

and binarized with dynamic threshold.

Valid values for DynamicThreshold are 0=off, 1=on.
If you want to determine the threshold yourself, you can turn off the DynamicThreshold and specify an individual threshold (iBC_Threshold).

13 Image Processing

Through the interfaces QSBarcode6() of p_lib and QSReadBarcode6() of f_dll and h_dll commands for the image processing can now be passed directly, which were read out in the older functions from a "qsbc.ini" file.

The commands are carried out on a copy of the image before the barcode recognition is processed. Thus, the input file is not changed.

For the image processing commands there are two libraries internally available. These are the FreeImage Library and Graphlib library.

Only commands from the Graphlib library can be used for the p_lib. In addition, some commands have no effect, because processing is always done on pictures with a color depth of 1 bit.

For the h_dll and the f_dll the commands from both libraries can be used. Mixed use of the command is also possible.

One or more commands for image processing can be passed to the respective barcode function through the variable "szImaging". Every single command must be completed with a semicolon, e.g.

"FreeImage:GrayScale,8;GraphLib:SubImage("c:\test1.bmp");"

The commands of the FreeImage Library:

There are currently only two commands available in this library.

Command	Description
FreeImage:GrayScale,8;	Converting an image to grayscale (8 bit color depth). The conversion can be made from any arbitrary color depth.
FreeImage:Threshold,160;	Transformation of the image into a b/w image. The threshold value for the conversion can be directly specified. Possible values for the threshold are between 0 (black) and 255 (white).

The command of the GraphLib Library:

Command	Description
GraphLib:Binarize(160);	Transformation of the image into a b/w image. The threshold value to for the conversion can be directly specified. Possible values for the threshold are between 0 (black) and 255 (white).
GraphLib:GrayScale(0);	Converting an image to grayscale (8 bit color depth). The conversion can only be carried out on images with 24-bit color depth. The value passed is not taken into account, it must be only numeric.
GraphLib:Dynamic;	The threshold value is computed from the brightness values occurring in the entire image. This command only works on grayscale images.

Command	Description
GraphLib:DynamicEx("0");	The threshold value is determined as in "Dynamic", but can be shifted by an offset. A positive offset is darkening the image; a negative value is brightening the image. This command only works on grayscale images. Range: -255 to +255

Command	Description
GraphLib:Contrast(2.0);	Adjusting the contrast (passed value always as a decimal). At the threshold of 1.0, the contrast is not changed. Values between 0.0 to 1.0 increase the contrast, all values above 1 reduce the contrast. This command only works on grayscale images.

Before: **After:**

Command	Description
GraphLib:Brighten(50);	Brightness of the image in %. The value range is between -100 and 100. Values between 0-100 make the image brighter, values between -100 - 0 make the image darker. This command only works on grayscale images.

Before: **After:**

Command	Description
GraphLib:DespeckEx(5,5);	Removal of black and white dots. The two values specify the maximum diameter of the black and white pixels to be removed. Only "stand-alone" dots in the specified size are removed. This command only works on images with 1-bit color depth.

Before:

After:

Command	Description
GraphLib:Dilation;	General black enhancement. This command works only on images with 1-bit color depth.

Before:

After:

Command	Description
GraphLib:Linearcontrast;	Linear contrast level adjustment. This command only works on grayscale images.

Before:

After:

Command	Description
GraphLib:Subimage("c:\test.bmp");	Saving the current state of the image. The file format is determined by the file name extension ".tif" or ".bmp".
GraphLib:SubimageEx("c:\test.bmp",0,0,500,500);	Saving an area of an image. Destination file name, StartX, StartY, width and height of the snippet will be passed.
GraphLib:Invert;	Inverting the image. This command only works on images with 1-bit color depth.

Command	Description
GraphLib:LocalThreshold(0,20,"0.9");	Three parameters control this command. The first parameter is 0 and cannot be changed. The second parameter controls the size of the region, where the local threshold is determined. Generally 20 has here proved to be good value. A lower value increases the speed but also the risk that there occur holes in solid white or black regions by brightness variations. The "0.9" as the last parameter says that all pixels, whose brightness is over 90% of medium brightness of the environment, are considered white. The command works only on grayscale images Caution: This function is very time consuming.

Before:

After:

Command	Description
GraphLib:VMedian(2);	Blurring black areas. The color values of neighbor pixels of a passed pixel size are sorted and the current pixel is replaced by the mean value. The larger the value passed in, the greater the change in area. This command works on images with 1-bit and 8-bit color depth.

Before:

After:

Command	Description
GraphLib:VDilation(1);	Reinforcing black areas. The color values of neighbor pixels of a passed pixel size are sorted and the current pixel is replaced by the largest value. The larger the value passed in, the greater the change in area. This command works on images with 1-bit and 8-bit color depth.

Before: **After:**


Command	Description
GraphLib:VErosion(1);	Thinning (too dark) black areas. The color values of neighbor pixels of a passed pixel size are sorted and the current pixel is replaced by the smallest value. The larger the value passed in, the greater the change in area. This command works on images with 1-bit and 8-bit color depth.

Before: **After:**


Command	Description
GraphLib:VOpening(3);	Opens thin connections between black areas. The larger the value passed in, the more connections are opened. This command works on images with 1-bit and 8-bit color depth.

Before: **After:**

Command	Description
GraphLib:VClosing(3);	Closing thin white gaps in black areas The larger the value passed in, the more gaps are closed. This command works on images with 1-bit and 8-bit color depth.

Before:

After:

For more information, please refer to the document "bcTipps_en.pdf".

14 Special Settings

14.1 Special settings for patch codes

Patchcodes have just 4 very long lines. We suggest setting iBC_MinHeight to a much higher value than 15. The value of iBC_LightMargin must be set higher than the thickest gap and also higher than the thickest bar. Please set iBC_Orientation to the right value, because the results change with different orientations.
The result is returned in the result structure as usually. The following table lists the result (0/1 combination of length 4).

Table of Patchcodes (1=thick line, 0=thin line)

Patch 1	1100
Patch 2	1001
Patch 3	1010
Patch 4	0110
Patch T	0101
Patch 6	0011

14.2 Special settings for Data Matrix Code

Recognition of the 2-D Code Data Matrix is available as an option. To get good results you should use the following settings:

iBC_LightMargin	DM_DEFAULT_LIGHT_MARGIN = 10
iBC_Percent	*irrelevant, not in use*
iBC_ScanDistBarcode	*irrelevant, not in use*
iBC_ScanDistance	DM_DEFAULT_SEARCH_DISTANCE = 10
iBC_Tolerance	DM_DEFAULT_TOLERANCE = 2
iBC_MaxNoVal	*irrelevant, not in use*
iBC_MinHeight	DM_DEFAULT_MIN_HEIGHT = 30
iBC_MaxHeight	DM_DEFAULT_MAX_HEIGHT = 2000

Beginning with QS-Barcode version 4.7 it is now possible to activate/deactivate the "Intensive Search" for DataMatrix barcodes. In version 4.6 and earlier, this "Intensive Search" was always active by default. But this intensive search is time consuming and there are many cases were the standard search does already reliably recognize all barcodes. Now you are able to decide by yourself if you need the "intensive search" or not.
"Intensive Search" does vary some of the position data parameters whilst searching and decoding the DataMatrix, thus trying slightly different settings for the recognition process. It helps if you have dirty barcodes with blurred or frayed finder borders.
Set iBC_Orientation to DM_INTENSIVE_SEARCH to activate "Intensive Search", set iBC_Orientation to 0 in all other cases. You might set `BC_0`, `BC_90`, `BC_180`, `BC_270`, it does not hurt but does not have any influence, the DataMatrix search always searches for all codes ignoring the orientation.

14.3 Special settings for QR Code

Recognition of the 2-D Code QR Code is available as an option. The extended parameters (iBC_LightMargin, iBC_Percent, iBC_ScanDistBarcode, iBC_ScanDistance, iBC_Tolerance, iBC_MaxNoVal, iBC_MinHeight, iBC_MaxHeight) are irrelevant for this barcode type.

14.4 Special settings for PDF417

Recognition of the 2-D Code PDF417 is available as an option. To get good results you should use the following settings:

iBC_LightMargin	PDF_DEFAULT_LIGHTMARGIN = 4 *Important: DO NOT CHANGE!*
iBC_Percent	*irrelevant, not in use*
iBC_ScanDistBarcode	*irrelevant, not in use*
iBC_ScanDistance	PDF_DEFAULT_SCANDISTANCE = 10
iBC_Tolerance	*irrelevant, not in use*
iBC_MaxNoVal	*irrelevant, not in use*
iBC_MinHeight	PDF_DEFAULT_MIN_HEIGHT = 15
iBC_MaxHeight	PDF_DEFAULT_MAX_HEIGHT = 2000

14.5 Special settings for Aztec Code

Recognition of the 2-D Code Aztec is available as an option. The extended parameters (iBC_LightMargin, iBC_Percent, iBC_ScanDistBarcode, iBC_ScanDistance, iBC_Tolerance, iBC_MaxNoVal, iBC_MinHeight, iBC_MaxHeight) are irrelevant for this barcode type.

14.6 Special settings for PostNet Codes

The license for linear barcode types includes the recognition of 2-state and 4-state Post Ident Codes. Without a valid (L) license option, the results of PostNet codes will be patched (see 13.6 Systematic result-altering).

The PostNet Code recognition is activated by choosing the right barcode type and setting iBC_Type2 to this type. It is not possible to combine two or more PostNet types in one recognition and it is not possible to add other barcode types.
To get good results you should use the following settings:

iBC_LightMargin	PN_ DEFAULT_LIGHT_MARGIN = 1
iBC_Percent	PN_ DEFAULT_PERCENT = 0
iBC_ScanDistBarcode	*irrelevant, not in use*
iBC_ScanDistance	PN_ DEFAULT_SEARCH_DISTANCE = 4
iBC_Tolerance	PN_DEFAULT_TOLERANCE = 2
iBC_MaxNoVal	PN_ DEFAULT_MAX_NO_VAL = 0
iBC_MinHeight	PN_DEFAULT_MIN_HEIGHT = 15
iBC_MaxHeight	PN_DEFAULT_MAX_HEIGHT = 200

Here is some information in case you want to change the default settings:

iBC_Lightmargin

Minimal number of white pixels in front of each bar in the PostNet Code. This also defines the minimum white gap between each of the bars in the Code. Changing the default value of 1 to a higher value allows a faster recognition process as fewer objects on the images need to be analyzed.

iBC_ScanDistance

Searching for the barcode on an image in y-direction is completed in steps. The distance between the steps is defined with this parameter. Higher values improve the performance since fewer lines are tested but of course it might happen that a barcode is not detected at all if the value is too high.
This value has to be 33% of the minimum height or less.

iBC_Tolerance

Defines the tolerance for data errors and image errors within the bars of the PostNet Code.

iBC_MinHeight und iBC_MaxHeight

The minimum height has to be less than the height of the small "tracker" bars of the Code and the maximum has to be more than the height of a "full" bar.
But do not choose a value much higher than the highest bar, try to keep the range as exact as possible (with some tolerance of course).

iBC_Percent (min. Width)

(the parameter "min. width" for PostNet Codes is called "percent" for other barcode types and has a quite different meaning there)
Defines the minimum width (pixel) for bars of the PostNet Code.
The default value 0 indicates "any". Changing this to a higher value improves the performance as fewer objects on the images need to be analysed.
Please note: If you choose a high value here, thin bars will be ignored and recognition might become impossible.

iBC_MaxNoVal (max. Width)

(the parameter "max. width" for PostNet Codes is called "max. gap" for other barcode types and has a quite different meaning there)
Defines the maximum width (pixel) for bars of the PostNet Code.
The default value 0 indicates "any". Changing this to a higher value improves the performance as fewer objects on the images need to be analysed.
Please note: If you choose a low value here, thick bars will be ignored and recognition might become impossible.

Recommendation

Be sure to set proper values for min. height and max. height, otherwise you will have trouble recognizing PostNet codes.
Change the values for light margin, min. width and max. width only if you need to reduce the overall recognition time.

15 Troubleshooting

15.1 ActiveX registration

Registration of ActiveX controls can be done manually in case of problems using the command

```
regsvr32.exe OCX-Name.ocx
```

Please notice that also the VisualBasic Runtime environment and the required DLLs must be present.

15.2 DLL not found

If a "DLL not found" error message is received, ensure that you have all DLLs required by the interface component used. The DLLs must be located in a directory where the system will find them, e.g. the Windows system directory.
For more information on the required DLLs, please refer to the "Redistributable Files" section.
Please note that an error "QSBarDLL_F.DLL not found" can also mean that the DLL is missing other DLLs!

15.3 Alignment problems

All programs in this barcode library are compiled with 8byte alignment, e.g. when using the p_lib, the project where the library is integrated should also use 8byte alignment. Should this prove difficult, please contact us and we can develop a customized version for you.
Alignment is also important if using the DLL interface, especially if transferring structures to QS-Barcode DLLs. The program must transfer the structures in the correct size.

15.4 Linker Problem

When using the **QS-Barcode Library-Interface** (especially with QSBarLib.Lib, but also with QSBarDLL_F.lib and QSBarDLL_H.lib) with **MS-VisualStudio**, the following linker error may appear:

Linker Tools Error LNK2005
<symbol> already defined in <object>

This often occurs when the **QS-Barcode** Library has different linker settings than your project.
All **QS-Barcode** libraries are compiled using the option "**Use Run-Time Library**" Single-Threaded. Your project must use the same run-time libraries.
If required, **QS QualitySoft GmbH** can provide you with different compiled versions of the QS-Barcode Library.

The following is an extract of Microsoft sources on this error:

Online help on linker error:

```
Linker Tools Error LNK2005
symbol already defined in object

The given symbol, displayed in its decorated form, was multiply defined.
```

```
Tips

One of the following may be a cause:

The most common cause of this error is accidentally linking with both the
single-threaded and multithreaded libraries. Ensure that the application
project file includes only the appropriate libraries and that any third-
party libraries have appropriately created single-threaded or multithreaded
versions.

The given symbol was a packaged function (created by compiling with /Gy)
and was included in more than one file but was changed between
compilations. Recompile all files that include the symbol.

The given symbol was defined differently in two member objects in different
libraries, and both member objects were used.

An absolute was defined twice, with a different value in each definition.
This error is followed by fatal error LNK1169.
```

MSDN-Article "/MD, /ML, /MT, /LD (Use Run-Time Library)"
Link: https://learn.microsoft.com/en-us/previous-versions/visualstudio/visual-studio-2013/2kzt1wy3(v=vs.120)

15.5 Barcode cannot be recognized

- Check whether the barcode can be read with your settings using the program described in chapter "The application bcTester ".

- Read our whitepaper "Barcodes not recognized – what can I do?" (BCTipps.pdf), which you can find in the directory "Documents" of bcTester or in the download area of our homepage.

- Try to read the barcode with a variation of the example programs.

- If uncertain, please send your scanned images to QS QualitySoft; we are happy to analyze them for you, mail to support@qualitysoft.de.

15.6 Systematic result-altering

If run in **demo mode**, QS Barcode SDK systematically alters the results. The following substitutions are made:
3->1, A,a -> Q,q, B,b -> S,s
QS Barcode SDK runs in demo mode unless it finds a valid **license file** (QSBC.lic) in the application path. Make sure your license file is in the appropriate position.

The license file is searched for in:

1. The directory from which the application loaded.
2. The current directory.
3. The Windows system directory.
4. The Windows directory.
5. The directories that are listed in the PATH environment variable.

16 Appendix

16.1 Supported File Formats

QS-Barcode uses the FreeImage open source image library to open image files.
See http://freeimage.sourceforge.net for details.
FreeImage is used under the Free Image Public License, version 1.0.

QS-Barcode supports a lot of image file formats with the OCX or F_DLL interface:

Standard-Shipment

- BMP files
- DDS files
- EXR files
- Raw Fax G3 files
- GIF files
- HDR files
- ICO files
- IFF files
- JNG files
- JPEG/JIF files
- JPEG-2000 File Format
- JPEG-2000 codestream
- KOALA files
- Kodak PhotoCD files
- MNG files
- PCX files
- PBM files
- PGM files
- PNG files
- PPM files
- PhotoShop files
- Sun RAS files
- SGI files
- TARGA files
- TIFF files
- WBMP files
- XBM files
- XPM files

A lot of other formats are available on request, just send us an mail and a sample image.

16.2 MultiPage support

MultiPage files such as MultiTiff files and Adobe PDF Documents often include more than one page in a file.

QS-Barcode 3.6 (and earlier) was able to read Barcodes on the first page of MultiTiff pages, but there was no way to access the second and further pages and read barcodes on these pages.

Beginning with QS-Barcode 3.7, this is now possible. Using the **f_ocx** or the **f_dll** interface, you can specify the page to read by setting the Image File Name Parameter to *"ImageFileName<Page"*, i.e. `"c:\temp\multipage.tif<2"` to access the second page of c:\temp\multipage.tif. The Page number starts with 1, calling `"c:\temp\multipage.tif<1"` is the same as calling `"c:\temp\multipage.tif"`.
If the specified page does not exist, the recognition will exit with returncode BC_NO_SUCHPAGE (hexadecimal = 0x0013, decimal=19).

16.3 Adobe PDF Documents – Special Settings

When processing Adobe PDF documents, QS-Barcode SDK needs to convert the page to read in a raster image format.
This allows a universal support for PDF documents. PDF documents created with a document scanner, including an image for each scanned page, are supported as well as PDF documents generated from within software, containing a mix of fonts, images and other objects.
As a user of QS-Barcode SDK, you may configure this conversion if necessary. This is done using a config file QSBC.INI.
Place this config file in the same folder as the QSBarDLL_F.DLL which is used by your application.

The content of QSBC.INI should be:

```
[PDF]
DPI=200
```

These are the default settings.

DPI = Resolution at which the document is read. If you have problems recognizing barcodes, it may be a good idea to increase the DPI value. But be careful: Higher values are time and memory consuming and it may be necessary to increase iBC_LightMargin as well. The maximum DPI value can be 400.

An indication of the color depth when reading PDF documents is not possible. PDF documents are always read with 24-bit color.

16.4 Image enhancement via "qsbc.ini"

Using the configuration file "qsbc.ini" extended parameters can be used for image enhancement prior to barcode recognition. Up to and including version 4.8.1 this "qsbc.ini" had to reside in the program directory of the calling program. From version 4.8.2 there are additional interfaces for explicitly passing a "qsbc.ini". This file must not be called "qsbc.ini" any longer. It can be any file name and can be located in any directory. Thus, it is now possible to better control the recognition of different barcode types and qualities.

The following calls can be used from version 4.8.2:

Library with pointer (p_lib)

```
int QSBarcode3( BarcodeParam2 *pBarcodeParam2, char *szIniFile );
int QSBarcode4( BarcodeParam2 *pBarcodeParam2, char *szIniFile, char
* szLicenseFile);
```

DLL with handle (h_dll)

```
int QSReadBarcode3(HANDLE hDIB, Barcode2 *pBarcode2,
     int iNumResults, char *szIniFile);
int QSReadBarcode4(HANDLE hDIB, Barcode2 *pBarcode2,
     int iNumResults, char *szIniFile, char *szLicenseFile);
```

DLL with file (f_dll)

```
int QSReadBarcode3 (char *szImageName, Barcode2 *pBarcode2,
     int iNumResults, char *szIniFile);
int QSReadBarcode4 (char *szImageName, Barcode2 *pBarcode2,
     int iNumResults, char *szIniFile, char *szLicenseFile);
```

If the passed "qsbc.ini" does not exist, the error code returned is `BC_QSBCINI_NOT_EXISTS`. When passing an empty string for the "qsbc.ini" using `QSBarcode4/QSReadBarcode4`, this parameter is ignored.

The behavior regarding the "qsbc.ini" using `QSBarcode2()` or `QSReadBarcode2()` has not changed.

Example:

Sometimes it happens by poor scan quality that a barcode shows black specks or small white pixel gaps in the black bars.

These gaps in the black bars can prevent the recognition of barcodes. The image enhancement "image error compensation" lets you fill these gaps so that a recognition of the barcode is possible again.

You need the following entry in a "qsbc.ini":

```
[CleanBresenham]
WhitePoints=1
BlackPoints=0
```

This "qsbc.ini" can be saved under any name, e.g. `"d:\barcode\ini\qsbc_white_pixel.ini"`.

The call to recognise the barcode can then for example read as follows:

```
QSReadBarcode3 (szImageName, pBarcode2,iNumResults,
     "d:\barcode\ini\qsbc_white_pixel.ini")
```

For more information about the parameters in the "qsbc.ini" see the document "\ sdk_doku \ bcTipps.pdf.

16.5 QR Code Extended Channel Interpretation (ECI) Mode

The QR code specification provides the ability to interpret the existing data in a QR code with different character sets in the ECI mode.

The current return of the data was performed in the format "]Q2\000009ÁÂÃÄÅ". The string started with "] Q2 \" followed by ECI code (000009) and the associated data "ÁÂÃÄÅ. This type of return will continue, if the type of return delivery described here is not used.

To obtain the data in the ECI mode, the following parameters must be set in the structure passed to the barcode reading function.

```
bcParams2.iBC_Type = (int)BC_Types.BC_QR_CODE;
bcParams2.iBC_Type2 = (int)BC_Types.BC_QR_CODE_ECI;
```

The return of the read ECI codes and their associated data is similar to the return of the data of 2D barcodes. Again, the variables in the extended structure `BC_TwoDimRes` are filled. `hBC_TwoDimRes` contains a handle to a memory area in which the ECI codes and their associated data are stored in a possibly multi-dimensional array of type QRECIResult_tag. The result of the total size of all structures is also stored in `iBC_TwoDimLen`. The number of structures in the handle can be determined by the total size in relation to the structure size.

The construction of the structure is as follows:

```
typedef struct QRECIResult_tag
{
      Int   iECI;                   // 4 BYTE: ECI Type
      void  *hBC_ECI_Item;          // 32-bit: 4 BYTE / 64-bit 8 BYTE:
                                    // ECI value stored in a handle
      Int   iBC_ECI_ItemLength;     // 4 BYTE: Length of hBC_ECI_Item
}QRECIResult;
```

The ECI code is stored in `iECI`, the corresponding data can be found in the Handle `hBC_ECI_Item`, the data length is stored in `iBC_ECI_ItemLength`.

If ECI codes are searched for but are not located in the barcode or it may be a mixture of ECI data and non-data in the barcode, the data is still returned in the ECI structure. For non-ECI data the ECI code is set to the value "0".

For details for reading the data, please see the sample program.

16.6 Update Notes

Important! If you are a customer and used QS-Barcode SDK 4.5 or earlier, this chapter is for you. Learn how to update to version 4.9.

When converting from 32-bit to 64-bit version of the QS Barcode SDK, it is sufficient to recompile the source code. You don't have to change the source code. You must deliver only the required DLLs for the right platform.

All the interfaces (**f_dll, f_ocx**, **h_dll** and **p_lib**) have been changed with the release of version 4.6. If you are already running version 4.6 and use the new interfeces, updating to 4.7 is easy, just use the new redistributable files.

There is a new parameter "iBC_Type2" in the data structure. To avoid mistakes we changed the names of all functions using this new data strucure as well (by adding a "2" at the end of the name).
But: To ensure compability, the old functions are still available and supported.
You will not be able to use the new features by using the new parameter "iBC_Type2" if you keep on using the old functions of course.

If you want to use the old interfaces, there is no need to recompile in case of using **f_dll,** or **h_dll**. If you use the **p_lib** or the **f_ocx** you need to rebuild your software.

So what are the benefits of the new parameter "iBC_Type2"? It was necessary as we reached the 32bit limit for different barcode types. iBC_Type2 allows us to add further barcode types in the future.
We started with the GS1 symbol family in Version 4.6 which will replace EAN and UPC for product identification worldwide in the next years. With support for GS1 barcode recognition, **QS-Barcode SDK** is fit for the future.

There is one other change: The parameters hBC_ErrorFile and iBC_Debug are not in use anymore. Initialize these with NULL resp. 0 and all should work fine.
As far as we know, only very few of our customers ever used these log-file entries. If you are one of those, just contact us and we will discuss other ways to log information.

17 Final Note

QualitySoft is eager to improve their products. Hints about errors or ambiguous parts are very welcome. QS-Barcode is permanently improved. Changes in the program may have happened without notice.

© QS QualitySoft GmbH
Tempowerkring 21a
D 21079 Hamburg
Tel: +49 (0) 40 790 100 40
info@qualitysoft.de
www.qualitysoft.de