QS-Barcode Recognition

Software Developer's Kit

Version 4.0

QS QualitySoft GmbH
Zum Fuerstenmoor 11
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

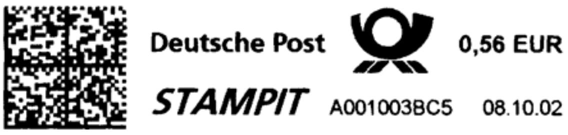
Content

1 Overview ”QS-Barcode-SDK“

QS-Barcode is an efficient software for recognizing barcodes from digital images (monochrome, grayscale and color). The images are created by scanning, digital photography or fax. Barcode in Adobe PDF documents can be read as well.

It 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)** and **QR Code (Model 2)** 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 is today used for small electrical parts, by the pharmaceutical industry for unit dose packaging, by the automotive industry and by NASA. The German Post uses it as "Stampit" Stamps. There it contains binary data. Both Data Matrix Barcodes 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 encodation of binary data and alphanumeric data is also available.
The **QR Code** has a build in error correction like the other 2D barcode have.

QS-Barcode SDK is available as developer library (C-Lib, DLL and OCX) and can easily be integrated to any environment.

QS-Barcode SDK was created from algorithms in the form reading software **"QS-Beleg"**, used successfully for years.
The most common use for barcode reading is fast and reliable identification and indexing of documents for data capture (OCR, ICR, etc.) and archiving. For this task there is also "**QS-DocumentAssembler"** available. It is a product for the enduser, not for the developer.

2 What 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 the 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 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, depends 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.

QS-Barcode is used sucessfully 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 ME, Windows NT 4.0, Windows 2000, Windows XP, Windows 2003, Windows Server 2003, Vista.

3 License Files

QS Barcode-SDK uses a license method. Two license files are included in your distribution of the QS Barcode-SDKs (`QSBC.lic`). The first license file is for use at 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 work station where QS Barcode is used. You must distribute this file and the license library (`1way.dll`) with your application using QS Barcode.
Both files are searched in the application path. If the license file cannot be found, QS Barcode runs in **demo mode** with systematic substitutions of result characters.

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

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.

4 On Barcodes

4.1 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 receive good barcode recognition without errors at this low resolution, the barcodes must be printed very clearly; they must not be positioned too closely. Recommended is maximal 2 characters/cm at 200 dpi and 3 characters/cm at 300 dpi. For example, a barcode of eight numbers 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 barcode

Not only are the number of bars and blanks important for barcodes, but so is 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: support@qualitysoft.de. We are always happy to help you!

4.2 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

1 234567 890128 EAN 13

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

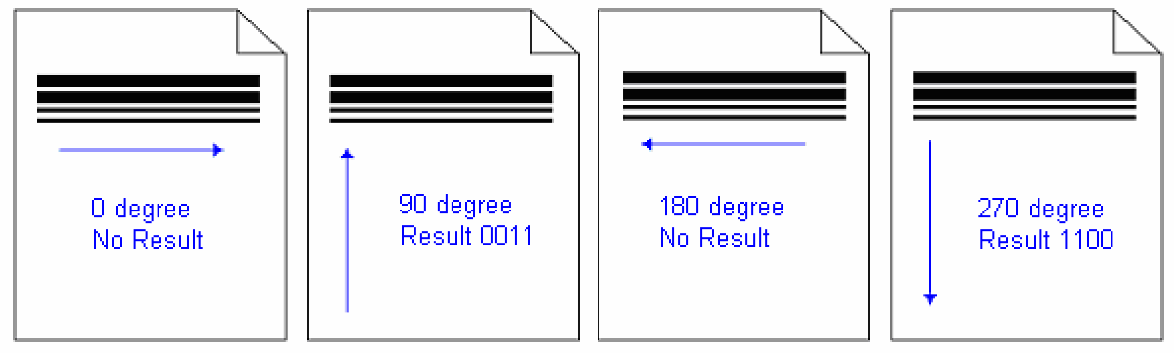
	Characters	Length
Code 39	0-9 A-Z - $ % + /	any
Code 39 extended	Complete ASCII-character set	any
Code 2/5 interleaved	0-9	any (even number)
Code 2/5 Industry	0-9	any
Code 93	0-9 A-Z - $ % + / 4 special chars	any
Codabar	0-9 - $: + / .	any
Code 128	complete ASCII character set	any
EAN 128	complete ASCII 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 can not 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 scanjobs. 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.

666

111111

Samples of Pharmacodes

Pharmacode consists of 4 to 16 bars and code numbers from 1 to 8496. 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.

4.3 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 organisations and vendors.
Nowadays the codes 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.) decoded.

QR Code

PDF 417

Data Matrix

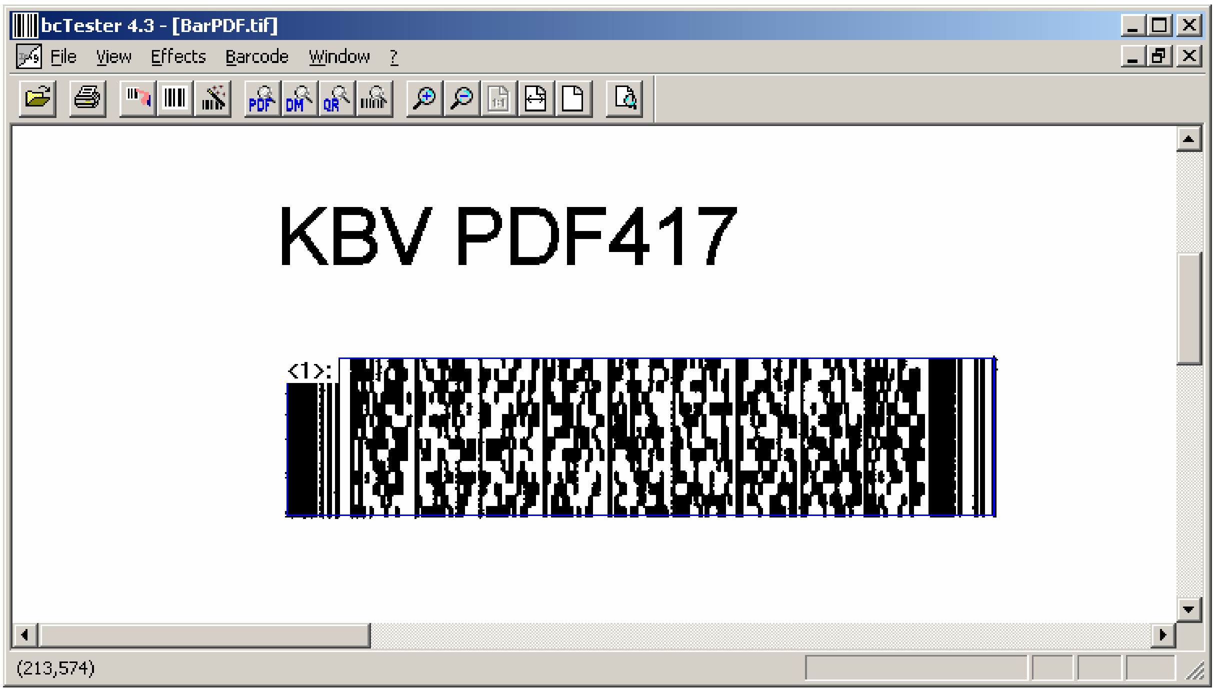
Three options of **QS-Barcode** are available to recognize **PDF 417**, **Data Matrix (ECC 000-140 and ECC 200)** and **QR Code (Model 2).**

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.

5 Program bcTester

Using bcTester you can test your barcodes and easily experiment with different settings.
At http://www.bctester.de/download/bctester_en.zip you can always find the latest version.



5.1 Testing Barcodes

The procedure is as follows:

1. Open an image which contains the barcode you wish to test.

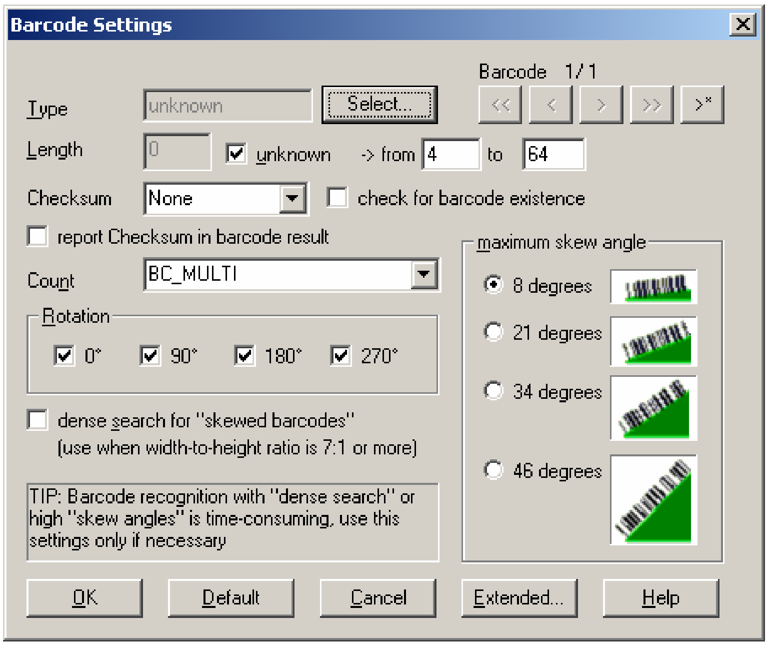
2. Open a rectangle around the barcode with the mouse. If you do not open a rectangle the search is performed on the whole image.

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

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

5.2 Barcode Settings

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



Barcode 1/1

Type: unknown

Length: 0 ☑ unknown -> from 4 to 64

Checksum: None ☐ check for barcode existence

☐ report Checksum in barcode result

Count: BC_MULTI

Rotation: ☑ 0° ☑ 90° ☑ 180° ☑ 270°

☐ dense search for "skewed barcodes"
(use when width-to-height ratio is 7:1 or more)

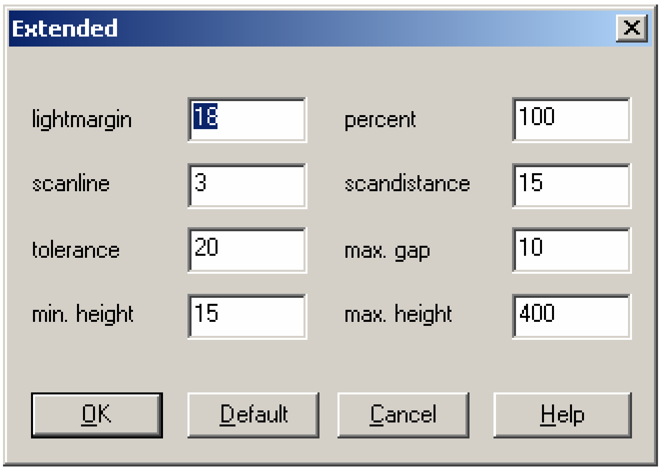
TIP: Barcode recognition with "dense search" or high "skew angles" is time-consuming, use this settings only if necessary

maximum skew angle
● 8 degrees
○ 21 degrees
○ 34 degrees
○ 46 degrees

Under ”Select“ you can select one or several types.
Using the buttons ”<<“, ”<“, ”>“, ”>>“, ”>*“ you can define several configurations. This way you can enter individual parameters for each barcode you wish to search, which may increase the reliability of recognition.
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)

Using the button „Extended“ opens a dialog for special parameters.

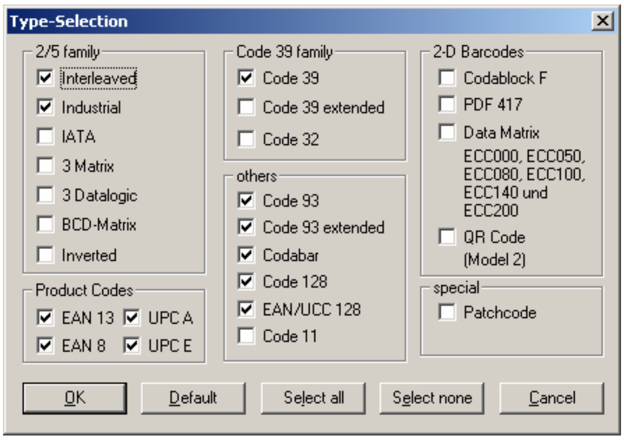


lightmargin: 18
percent: 100
scanline: 3
scandistance: 15
tolerance: 20
max. gap: 10
min. height: 15
max. height: 400

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**
lightmargin	iBC_LightMargin
percent	iBC_Percent
scanline	iBC_ScanDistance
scandistance	iBC_ScanDistBarcode
tolerance	iBC_Tolerance
max. gap	iBC_MaxNoVal
min. height	iBC_MinHeight
max. height	iBC_MaxHeight

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



2/5 family
☑ Interleaved
☑ Industrial
☐ IATA
☐ 3 Matrix
☐ 3 Datalogic
☐ BCD-Matrix
☐ Inverted

Product Codes
☑ EAN 13 ☑ UPC A
☑ EAN 8 ☑ UPC E

Code 39 family
☑ Code 39
☐ Code 39 extended
☐ Code 32

others
☑ Code 93
☑ Code 93 extended
☑ Codabar
☑ Code 128
☑ EAN/UCC 128
☐ Code 11

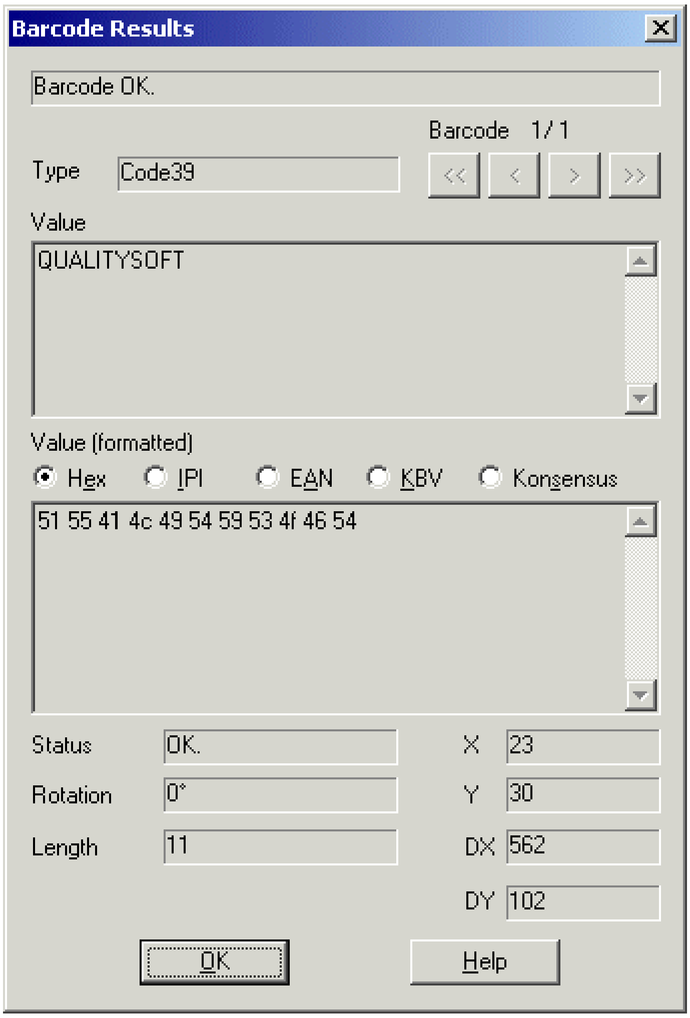
2-D Barcodes
☐ Codablock F
☐ PDF 417
☐ Data Matrix
ECC000, ECC050, ECC080, ECC100, ECC140 und ECC200
☐ QR Code (Model 2)

special
☐ Patchcode

BC_INTERLEAVED25	BC_CODE39	BC_CODABLOCK
BC_INDUSTRIE25	BC_CODE39EXT	BC_PDF417
BC_25_IATA	BC_CODE32	BC_DATAMATRIX
BC_25_3MATRIX		BC_QR_CODE
BC_25_3DATALOGIC	BC_CODE93	
BC_25_BCDMATRIX	BC_CODE93EXT	
BC_25_INVERTIERT	BC_CODABAR	
	BC_CODE128	BC_PATCHCODE
BC_EAN13	BC_EAN128	
BC_UPC_A	BC_CODE11	
BC_EAN8		
BC_UPC_E		

Table: Defines for type selection

5.3 Barcode Results



Barcode OK.

Barcode 1/1

Type: Code39

Value: QUALITYSOFT

Value (formatted)

● Hex ○ IPI ○ EAN ○ KBV ○ Konsensus

51 55 41 4c 49 54 59 53 4f 46 54

Status: OK.
Rotation: 0°
Length: 11
X: 23
Y: 30
DX: 562
DY: 102

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.

5.4 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.

6 Library with Pointer (p_lib)

6.1 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 function:

```
int QSBarcode( BarcodeParam *pBarcodeParam );
```

The **function parameter** `BarcodeParam` is a structure and is defined as described in the “Definitions” chapter.

To determine your **license** type (Linear barcode (L), Data Matrix (D), PDF417 (P)) use the function

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

This function returns a combination of the following constants: `BC_LIC_DEMO`, `BC_LIC_LINEAR`, `BC_LIC_PDF417`, `BC_LIC_DATAM`, `BC_LIC_QR_CODE`

6.2 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
BarcodeData barData; //Barcode data
BarcodeParam barParam; //Barcode parameter
void FAR *lpResultMem = 0; //Pointer to the result
int iNumResults=10; //Maximal Number results

// Supply needed memory
lpResultMem = malloc(sizeof(BarcodeResult) * iNumResults);

// At first empty structures
memset(&barParam, 0, sizeof(BarcodeParam));
memset(&barData, 0, sizeof(BarcodeData));

// fill in the BarcodeData- Structure with suitable values ...
barData.iBC_Type = BC_INTERLEAVED25 | BC_CODE39; //search these types
barData.iBC_Length = 0; //Length unknown
barData.iLaengeVon = 4; //somwhere between 4
barData.iLaengeBis = 12; //and 12
barData.iBC_Checksum = 0;
barData.iBC_Orientation = BC_0 | BC_90 | BC_180 | BC_270;
barData.iBC_LightMargin = 18;
barData.iBC_ScanDistance = 3;
barData.iBC_ScanDistBarcode = 1;
barData.iBC_ReadMultiple = 1;
barData.iBC_Percent = 100;
barData.iBC_MaxNoVal = 10;
barData.iBC_Tolerance = 20;
barData.iBC_MinHeight = 15;
```

```
    barData.iBC_MaxHeight = 400;
    barData.pBC_NextBarcodeData = NULL;

    // fill in the BarcodeParam-structure with the correct values ...
    strcpy( barParam.szBC_Version, BC_VERSION );
    barParam.lpstrBC_Image = lpBits; //Bits of the pixel
    barParam.pBC_BarcodeData = &barData; //BarcodeSettings
    barParam.iBC_Width = nWidth; //Width of the image
    barParam.iBC_Height = nHeight; //Height of the image
    barParam.iBC_StartX = 0; //Start
    barParam.iBC_StartY = 0;
    barParam.iBC_SizeX = 0;
    barParam.iBC_SizeY = 0;

    barParam.BC_RotInfo.dBC_cos = 1.0; //Important: 1.0!
    barParam.BC_RotInfo.dBC_sin = 0;  //Rest can be 0, if no
                                      //rotation is wished
    barParam.BC_RotInfo.iBC_BMoffsetX = 0;
    barParam.BC_RotInfo.iBC_BMoffsetY = 0;
    barParam.BC_RotInfo.fBC_bRotated = 0;
    barParam.BC_RotInfo.iBC_offsetX = 0;
    barParam.BC_RotInfo.iBC_offsetY = 0;
    barParam.BC_RotInfo.dBC_XKorr = 0;
    barParam.BC_RotInfo.dBC_YKorr = 0;

    barParam.pBC_Memory = lpResultMem; //results to this point
    barParam.lBC_MemorySize = sizeof(BarcodeResult) * iNumResults;
    barParam.iBC_Debug = 0;

    // now we can start the search ...
    iReturn = QSBarcode(&barParam);

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

        printf("%i. result: %s\n", nResult,
               barParam.pbrBC_Result[nResult].szBC_Barcode);
    }

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

6.3 Example programs

6.3.1 C-Example program

The directory P_Lib/Sample/c contains a simple example program for integrating the library in C. The example program has been written and tested with MS Visual C++ 6.0.

The program expects the file name of a monochrome BMP file (for grayscale/color files, select another interfaces) as the command line parameter. This file is read, searched for barcodes and the barcodes located are given in the standard output. The barcode parameters are set up in the code.

6.4 Definitions

Information about single values are found in the “Barcode-Parameters” chapter. Click the corresponding parameter to move here.

```
typedef struct BarcodeParam_tag
{
    char           szBC_Version [30];
    PBarcodeData  pBC_BarcodeData;
    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;
} BarcodeParam;
```

```
typedef struct BarcodeData_tag
{
    int            iBC_Type;
    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;
    PBarcodeData  pBC_NextBarcodeData;
} BarcodeData;
```

```
typedef struct BarcodeResult_tag
{
    int         iBC_Type;
    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;
} BarcodeResult;


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;
```

Recognition of the two-dimensional PDF 417 and Data Matrix barcodes is not part of the standard delivery!

6.5 Redistributable Files

To integrate QS-Barcode Library in its form as P_Lib interface, you must integrate the `QSBarLib.lib` into your application.

☞ Also you need to distribute your **license file** (`QSBC.lic`) and the **license-dll** (`1way.dll`) with your application and install them in the application path.

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

7 DLL with Handle (h_dll)

7.1 Interface

This DLL reads the barcodes from the passed on image section (or the whole image) using Handle:

```
int QSReadBarcode(HANDLE hDIB, Barcode *pBarcode, int iNumResults);
int QSGetNextBarResult(BarcodeResult *pBarcodeResult);
int QSFreeBarResult(void);
```

Due to the set up of `QSReadBarcode` the indicated image barcodes are read and saved in an internal structure. This way maximal `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).
To receive the **barcode results** in a loop, the function `QSGetNextBarResult` is started. As long as this function returns `BC_OK`, barcodes still exist. For every recognized barcode a `BarcodeResult`-structure is set up.
At the end of the recognition the function `QSFreeResult` **must** be started in order to release internal used memory.
Before performing a second set up, the results must be read from the function `QSReadBarcode,` otherwise they will be lost.

The **function parameter** `Barcode` is a structure and is defined as described in the “Definitions” chapter.

The structures `BarcodeResult` and `TwoDimResult` correspond to those of the library version that were described in the “Definitions” chapter.

Use the following function to determine your **license** type (Linear barcodes (L), Data Matrix (D), PDF417 (P) or Demo):

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

This function returns a combination of the following constants: `BC_LIC_DEMO, BC_LIC_LINEAR, BC_LIC_PDF417, BC_LIC_DATAM, BC_LIC_QR_CODE`

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 see “C-Example program”.

```
    int    iReturn,nResult,iNumResults;
    char   cBMName[500];
    HANDLE hpfile,hDIB;
    Barcode pBarcode;
    BarcodeResult pBarcodeResult;

    //hDIB = Handle on the bits of the image

    if(hDIB==NULL)
        return 0;
    // At first empty structure
    memset(&pBarcode, 0, sizeof(Barcode));
    memset(&pBarcodeResult, 0, sizeof(BarcodeResult));

    // Fill in the barcode-structure with suitable values.
    pBarcode.iBC_Type = BC_INTERLEAVED25 | BC_CODE39;
    pBarcode.iBC_Length = 0;
    pBarcode.iBC_Checksum = 0;
    pBarcode.iBC_Orientation = BC_0 | BC_90 | BC_180 | BC_270;
    pBarcode.iBC_ReadMultiple = 1;
    pBarcode.iBC_LightMargin = 18;
    pBarcode.iBC_ScanDistance = 2;
    pBarcode.iBC_Percent = 100;
    pBarcode.iBC_ScanDistBarcode = 1;
    pBarcode.iBC_MaxHeight = 400;
    pBarcode.iBC_MinHeight = 15;
    pBarcode.iBC_MaxNoVal = 10;
    pBarcode.iBC_Tolerance = 10;
    pBarcode.iBC_StartX = 0;
    pBarcode.iBC_StartY = 0;
    pBarcode.iBC_SizeX = 0;
    pBarcode.iBC_SizeY = 0;
    pBarcode.iLaengeVon = 4;
    pBarcode.iLaengeBis = 12;
    iNumResults=10;

    // Now we can start the search
    iReturn = QSReadBarcode(hDIB, &pBarcode, iNumResults);
    //put out results
    for(nResult=0;nResult < iNumResults;nResult++) {

        iReturn = QSGetNextBarResult(&pBarcodeResult);

if(iReturn!=BC_NO_BARCODE)
            printf("%i. result: %s\n",
                   nResult, pBarcodeResult.szBC_Barcode);
        else
            break;
    }

    // Do not forget to set free the results!
    GlobalUnlock(hDIB);
    GlobalFree(hDIB);
    iReturn = QSFreeBarResult();
```

7.3 Example programs

7.3.1 C-Example program

The directory H_DLL/Sample/c contains a simple example program for the integration of the DLL in C. The example program has been written and tested with MS Visual C++ 6.0.

The program expects the file name of a monochrome, grayscale or color BMP file as command line parameter. This file is read and a handle of to a Windows Device Independent Bitmap is generated. This DIB is searched for barcodes and the barcodes located are given in the standard output. The barcode parameters are set up in the code.

7.4 Definitions

```
typedef struct Barcode_tag
{
    int iBC_Type;
    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;
} Barcode;
```

The structures `BarcodeResult` and `TwoDimResult` correspond to those of the library version and have already been described in the “Definitions” chapter.

7.5 Redistributable Files

The runtime license allows your application to deliver the files required by this component:

- **`GraphLib.DLL`**, **`Graph_eng.DLL`** and **`QSBImage.DLL`**
- **`lfbmp13n.dll`**
- **`LFCMP13n.DLL`**
- **`lffax13n.dll`**
- **`ltgif13n.dll`**
- **`Lfpng13n.dll`**
- **`lftif13n.dll`**
- **`LTCLR13n.dll`**
- **`LTDIS13n.dll`**
- **`ltefx13n.dll`**
- **`ltfil13n.DLL`**
- **`Ltimg13n.dll`**
- **`ltkrn13n.dll`**
-

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

☞ You also must distribute your **license file** (`QSBC.lic`) and the **license-dll** (`1way.dll`) with your application and install them in the application path.

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

8 DLL with File (f_dll)

8.1 Interface

This DLL reads barcodes from a file specified by names:

```
int QSReadBarcode(char *szImageName, Barcode *pBarcode,
                  int iNumResults);
int QSGetNextBarResult(BarcodeResult *pBarcodeResult);
int QSFreeBarResult(void);
```

Due to the set up of `QSReadBarcode` the indicated image barcodes are read and saved in an internal structure. This way maximal `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).

New	Support for Adobe PDF documents added! You can use Adobe PDF documents in the same way you use image files with QS-Barcode SDK. See Appendix “12.2 Adobe PDF documents – special settings” for some additional hints. Accessing single pages of a MultiPage File (MultiTiff or Adobe PDF document) is now possible, see Appendix “12.1 MultiPage Support” for details

To receive the **barcode results** in a loop, the function `QSGetNextBarResult` is started. As long as this function returns `BC_OK`, barcodes still exist. For every recognized barcode a `BarcodeResult`-structure is set up.
At the end of the recognition the function `QSFreeResult` **must** be started in order to release the internal used memory.
Before performing a second set up, the results must be read from the function `QSReadBarcode`, otherwise they will be lost.
The **function parameter** `Barcode` is a structure and is defined as described in the “Definitions” chapter.
The structures `BarcodeResult` and `TwoDimResult` correspond to those of the library version and have already been described in the “Definitions” chapter.

Use the following function to determine your **license** type (Linear barcodes (L), Data Matrix (D), PDF417 (P) or Demo)

```
int QSLicense();
```

This function returns a combination of the following constants: `BC_LIC_DEMO, BC_LIC_LINEAR, BC_LIC_PDF417, BC_LIC_DATAM, BC_LIC_QR_CODE`

8.2 Integration Example

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

```
Declare Function QSReadBarcode Lib "QSBarDll.F_Dll" _
     (ByVal szImageName As String, pBarcode As WU_Barcode, _
      ByVal iNumResults As Long) As Long
```

```
Declare Function QSGetNextBarResult Lib "QSBarDll.F_Dll" _
     (pBarcodeResult As WU_BarcodeResult) As Long
Declare Function QSFreeBarResult 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_Barcode.
Type WU_Barcode
    iBC_Type As Long
    iBC_Length As Long
'...
    iLaengeBis As Long
End Type
' Type WU_BarcodeResult.
Type WU_BarcodeResult
    iBC_Type As Long
    iBC_Status As Long
'...
    sBC_TwoDimRes As WU_TwoDimResult
End Type

'*** Defaultvalues
Global Const BC_LIGHTMARGIN = 18
Global Const BC_SCANDISTANCE = 5
'...

'*** 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,
                         BCBarcode As WU_Barcode) As String
Dim iNumResults As Integer
Dim BCResult As WU_BarcodeResult
Dim BCResultNr As Byte
Dim a As Integer

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

    iNumResults=1
    'Calling uo with image name, barcode-Parameters and wished
    'results.
    QSReadBarcode BMPName, BCBarcode, iNumResults
    'Check results (in this example only a single one)
    QSGetNextBarResult BCResult

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

End Function
```

8.3 Example Programs

8.3.1 Access 2000

Path for example: F_dll/sample/Acc00

By starting the MS Access 2000 database in the directory, a simple form with one button is displayed. This starts the barcode recognition for the attached image `BarTest.tif.`
Use F11 to enter the database. Use the QSUtils module to see the QSTestBarcodeRead function. There the set up of the DLL from Access is demonstrated. The barcode parameters and the file name used are coded.

8.3.2 Visual Basic 6.0

Path for example: F_dll/sample/vb

The example program reads images out of a source path, names them according to barcodes found and transfers them into a target directory.
Both the paths and some of the barcode parameters are selectable.

Notice that on the attached image you find different barcodes. If you select more than one barcode in the option dialog, the image cannot be renamed!

8.3.3 Delphi 6.0

Path for example: F_dll/sample/delphi

The example program reads the sample image `bartest.tif` and returns the barcode results as console output. It demonstrates how accessing QS-Barcode within the Delphi environment can be managed.

8.3.4 Java

Path for example: F_dll/sample/java

The example program reads the sample image `bartest.tif` and returns the barcode results as console output. It shows how accessing QS-Barcode within the Java environment can be managed. Look at the readme.txt for more details.

8.4 Definitions

The structure `Barcode_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.

8.5 Redistributable Files

The runtime license allows you to deliver the following files:

- **QSBARDLL_F.DLL**
- **GraphLib.DLL, Graph_ger.DLL** and **QSBImage.DLL**
- **lfbmp13n.dll, LFCMP13n.DLL, lffax13n.dll, ltgif13n.dll**
- **Lfpng13n.dll, lftif13n.dll, LTCLR13n.dll, LTDIS13n.dll**
- **ltefx13n.dll, ltfil13n.DLL, Ltimg13n.dll, ltkrn13n.dll**

- **If you want to work with Adobe PDF Documents:**
  cimage.dll, pdf2image.dll, pdf2img.ini, pvsdk.ini, winfont.map
  The whole folder \ENCODING (133 Files).

☞ They must be located either in the application path or in another path where the system locates DLLs.

☞ The Folder \ENCODING has to be a sub-folder of the folder where **`pdf2image.dll`** is stored. Don’t change the folder name „ENCODING“.

☞ You must also distribute your **license file** (`QSBC.lic`) and the **license-dll** (`1way.dll`) with your application and install them in the application path.

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

9 ActiveX with File (f_ocx)

9.1 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.

New	Support for Adobe PDF documents added! You can use Adobe PDF documents in the same way you use image files with QS-Barcode SDK. See Appendix “12.2 Adobe PDF documents – special settings” for some additional hints. Accessing single pages of a MultiPage File (MultiTiff or Adobe PDF document) is now possible, see Appendix “12.1 MultiPage Support” for details

9.2 Integration

At 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\setup).

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

- **QSBARDLL_F.DLL**
- **GraphLib.DLL, Graph_ger.DLL** and **QSBImage.DLL**
- **lfbmp13n.dll**
- **LFCMP13n.DLL**
- **lffax13n.dll**
- **ltgif13n.dll**
- **Lfpng13n.dll**
- **lftif13n.dll**
- **LTCLR13n.dll**
- **LTDIS13n.dll**
- **ltefx13n.dll**
- **ltfil13n.DLL**
- **Ltimg13n.dll**
- **ltkrn13n.dll**
- **cimage.dll, pdf2image.dll, pdf2img.ini, pvsdk.ini, winfont.map**
  The whole folder \ENCODING (133 Files)

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
```

9.3 Methods

The set up is performed using the function

```
rcBC = QSBCOCX1.getBarcodeResult()
```

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

```
rcBC = QSBCOCX1.freeBarcodeResult()
```

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.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`, which **License version** (Linear barcodes (L), Data Matrix (D) and PDF417 (P) maybe Demo) you are using. The function `getLicenseName` returns a text version of the license version.

9.4 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.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.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
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

9.5 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
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)

9.6 Short Example Application

```
Option Explicit

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

  rcBC = BC_OK
  BCResultNr = 0

  'some settings for Barcode
    QSBCOCX1.PictureName = "c:\Qsbcocx\quelle\BarTest.tif"
    QSBCOCX1.BarcodeType = BC_ALL
    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.getBarcodeResult

  While rcBC = BC_OK
    BCResultNr = BCResultNr + 1
    With frmTest.ListErgebnisse
      .AddItem BCResultNr & ". Result: "
      .AddItem "Type = " & QSBCOCX1.getTypeName(QSBCOCX1.ResultType)
      .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.getBarcodeResult
  Wend
  MsgBox "Results stopped with " & QSBCOCX1.getErrorName(rcBC)
  'free the memory
  rcBC = QSBCOCX1.freeBarcodeResult
  MsgBox "Free stopped with " & QSBCOCX1.getErrorName(rcBC)

End Sub
```

9.7 Example Programs

9.7.1 Visual Basic 6.0

Path for example: F_ocx/sample/vb/project
Directory and program: F_ocx/sample/vb/project

The example program reads images from a source path, names them according to found barcodes and transfers them into a target directory.
Both paths and some barcode parameters can be selected. .

9.8 Redistributable Files

The following files can be delivered using the runtime license:

- **`QSBCOCX.OCX`**
- **`QSBARDLL_F.DLL`**
- **`QSBARDLL_H.DLL`**
- **`GraphLib.DLL`, `Graph_eng.DLL`** and **`QSBImage.DLL`**
- **`lfbmp13n.dll`**
- **`LFCMP13n.DLL`**
- **`lffax13n.dll`**
- **`ltgif13n.dll`**
- **`Lfpng13n.dll`**
- **`lftif13n.dll`**
- **`LTCLR13n.dll`**
- **`LTDIS13n.dll`**
- **`ltefx13n.dll`**
- **`ltfil13n.DLL`**
- **`Ltimg13n.dll`**
- **`ltkrn13n.dll`**
- **`If you want to work with Adobe PDF Documents:`**
  `cimage.dll, pdf2image.dll, pdf2img.ini, pvsdk.ini, winfont.map`
  `The whole Folder \ENCODING (133 Files).`

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

☞ The Folder \ENCODING has to be a sub-folder of the folder where **`pdf2image.dll`** is stored. Don’t change the folder name „ENCODING“.

☞ You must also distribute your **license file** (`QSBC.lic`) and the **license-dll** (`1way.dll`) with your application and install them in the application path.

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

☞ You must **register** the component ltocx13n.ocx and QSBCOCX_F.ocx , e.g. with the command `regsvr32.exe XXX.ocx`, or use the supplied setup (`F_ocx/setup) for the ocx.`

10 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.

10.1 Barcode-Parameters

10.1.1 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: BarcodeParam_tag)

10.1.2 hBC_ErrorFile

Here you pass on a file handle received e.g from OpenFile. If `iBC_Debug` has a value higher than zero, a debug output is written to this file.

(p_lib: BarcodeParam_tag)

10.1.3 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.`

(p_lib: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: BarcodeChecksum)

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 suspection”. For controlling this check, `iBC_Percent` is used (for details look there).

(p_lib: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: BarcodeChecksum)

10.1.4 iBC_Debug

If this value is higher than zero, a debug output is performed to the file that was written with `hBC_ErrorFile`.

The size of this value determines the detail of the output (0-10).

(p_lib: BarcodeParam_tag)

10.1.5 iBC_Height

Height of the image in pixels.

(p_lib: BarcodeParam_tag)

10.1.6 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: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: BarcodeLength)

10.1.7 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: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: LightMargin)

10.1.8 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: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: MaxHeight)

10.1.9 iBC_MaxNoVal

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

Default value: 10

(p_lib: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: MaxNoVal)

10.1.10 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: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: MinHeight)

10.1.11 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`.

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.

Barcode that are very low could slip through the check. Use the constant `BC_SKEW_DENSE_SEARCH` to add another scan between the other orientations.

Please keep in mind that the more orientations are searched, the longer the barcode search takes! Only use orientations you really need!

(p_lib: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: BarcodeOrientation)

10.1.12 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: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: Percent)

10.1.13 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: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: ReadMultiple)

10.1.14 iBC_ResultCount

Indicates the number of result structures in `brBC_Result`.

(p_lib: BarcodeParam_tag)

10.1.15 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 QS-Barcode Demo, this parameter is called “Searchdistance”.

(p_lib: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: ScanDistance)

10.1.16 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 QS-Barcode Demo, this parameter is called “Scandistance”.

(p_lib: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: ScanDistBarcode)

10.1.17 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: BarcodeParam_tag, f_dll: Barcode_tag, f_ocx: SizeX)

10.1.18 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: BarcodeParam_tag, f_dll: Barcode_tag, f_ocx: SizeY)

10.1.19 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: BarcodeParam_tag, f_dll: Barcode_tag, f_ocx: StartX)

10.1.20 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: BarcodeParam_tag, f_dll: Barcode_tag, f_ocx: StartY)

10.1.21 iBC_Tolerance

Maximal distortion of the barcode (line tolerance).

Default value: 10

(p_lib: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: Tolerance)

10.1.22 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.`

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 and Data Matrix are not part of the standard delivery!)

(p_lib: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: BarcodeType)

10.1.23 iBC_Width

Width of the image in pixels.

(p_lib: BarcodeParam_tag)

10.1.24 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: BarcodeData_tag, f_dll: Barcode_tag, f_ocx BarcodeLength_to)

10.1.25 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: BarcodeData_tag, f_dll: Barcode_tag, f_ocx: BarcodeLength_from)

10.1.26 lBC_MemorySize

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

(p_lib: BarcodeParam_tag)

10.1.27 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: BarcodeParam_tag)

10.1.28 pBC_BarcodeData

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

(p_lib: BarcodeParam_tag)

10.1.29 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: BarcodeParam_tag)

10.1.30 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 `BarcodeData_tag`-block is filled, where the different barcode types are linked with OR and the length is set to the expected results (e. g. 4 to 11), or a `BarcodeData_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 for XXX, the types may also be linked with OR and the length can be entered in the interval. Then no additional `BarcodeData_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: BarcodeData_tag)

10.1.31 pbrBC_Result

This is a pointer to the result-structures.

(p_lib: BarcodeParam_tag)

10.1.32 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: BarcodeParam_tag)

10.1.33 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: BarcodeParam_tag)

10.1.34 szBC_Version

This parameter contains the current version number of the library.

(p_lib: BarcodeParam_tag)

10.2 Return Values

10.2.1 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)

10.2.2 dBC_sin

This is the sin value of the rotation angle.

This parameter mainly supports the compatibility to QS-Beleg.

(p_lib: tag_RotInfo)

10.2.3 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)

10.2.4 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)

10.2.5 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.

10.2.6 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)

10.2.7 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)

10.2.8 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)

10.2.9 iBC_TwoDimCols

Number of recognized columns in the barcode.

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

10.2.10 iBC_PdfECL

Error correction level. Used only for PDF417.

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

10.2.11 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)

10.2.12 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)

10.2.13 iBC_Orientation

Returns the orientation of the found barcode.

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

10.2.14 iBC_TwoDimLen

Length of the result in bytes.

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

10.2.15 iBC_TwoDimRows

Number of recognized rows in the barcode.

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

10.2.16 iBC_SizeX

Width of the barcode found in pixel.

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

10.2.17 iBC_SizeY

Height of the barcode found in pixel.

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

10.2.18 iBC_StartX

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

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

10.2.19 iBC_StartY

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

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

10.2.20 iBC_Status

The status of the recognized barcode:

- `BC_OK` for clearly recognized barcodes
- `BC_GUESS` for the most often occuring value in the mode `BC_MULTI_BESTGUESS`
- `BC_EXISTS` if no value or existence verification could be found

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

10.2.21 iBC_Type

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

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

10.2.22 szBC_Barcode

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

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

11 Special Settings

11.1 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

11.2 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

11.3 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.

11.4 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

12 Troubleshooting

12.1 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.

12.2 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!

12.3 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.

12.4 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: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vccore/html/_core_.2f.MD.2c_2f.ML.2c_2f.MT.2c_2f.LD.asp (very slow!)

12.5 Barcode cannot be recognized

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

12.6 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.

13 Appendix

13.1 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).

QS-Barcode Version 3.8 adds support for Adobe PDF Documents, and of course this feature to access each page in a file also works with Adobe PDF Documents.

13.2 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:

```
[VeryPDF]
DPIx=200
DPIy=200
BPP=24
```

These are the default settings.

DPIx and DPIy should always be set to the same value. If you have problems recognizing barcodes, it may be a good idea to increase the DPI values. But be careful: Higher values are time consuming and it may be necessary to increase iBC_LightMargin as well.
BPP = Bits per Pixel defines the color depth of the conversion process. Choose the same depth as you use during scanning or creating the PDF documents.

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

Zum Fuerstenmoor 11
D 21079 Hamburg
Tel: +49 (0) 40 790 100 40   info@qualitysoft.de www.qualitysoft.de