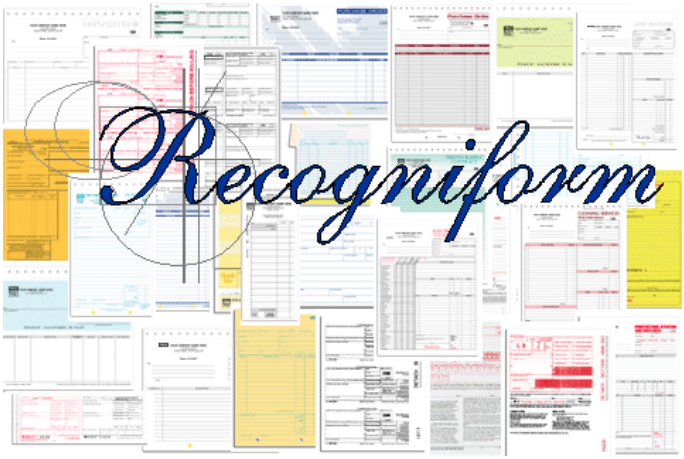


Software Development Kit

BCR

Recogniform Technologies SpA

HOW TO CONTACT US

Recogniform Technologies SpA
Contrada Concistocchi
87036 Rende (CS), Italy

Phone : +39 0984 404174

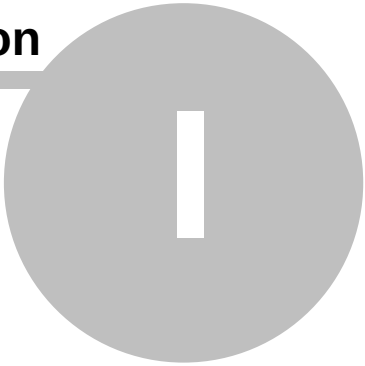
Fax : +39 0984 830299

Internet : www.recogniform.com

E-Mail : info@recogniform.com

Table of contents

Introduction



1 Introduction

1.1 Copyright

1.2 License

1.3 Overview

The Recogniform Barcode Recognition Library 5.0 allows to recognize barcodes from images acquired using a scanner.
The barcodes type recognizable are:

- 2of5Interlaved

- 2of5IATA
- 2of5Industrial
- 2of5Matrix
- 2of5DataLogic
- Code39
- PharmaCode
- EAN13
- EAN8
- UPCA
- UPCE
- EXT2
- EXT5
- CodaBar
- MSI
- Code11
- Code128
- Code93
- Code39Extended

The barcodes can be recognized in any orthogonal orientation:

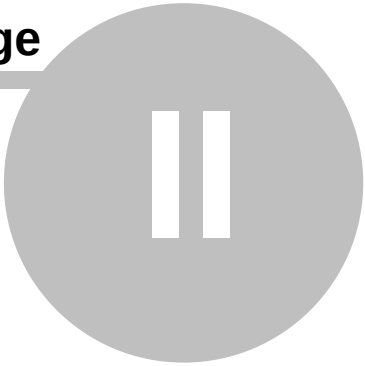
- LeftToRight
- RightToLeft
- TopToBottom
- BottomToTop.

The recognition process is fast and accurate and for each recognized barcode you can get:

- length
- data
- area
- orientation
- checksum flag

We can also create specialized and customized recognizers for specific use if you need.

Usage



2 Usage

2.1 Visual C++

You have to include the RECOBCRAPI.C in your program.

Before to execute your application make sure the *RECOBCR.DLL* is available in your same .exe directory or in windows\system directory.

2.2 C#

You have to include the RECOBCRAPI.CS in your program.

Before to execute your application make sure the *RECOBCR.DLL* is available in your same .exe directory or in windows\system directory.

2.3 Visual Basic

You have to include the RECOBCRAPI.BAS in your program.

Before to execute your application make sure the *RECOBCR.DLL* is available in your same .exe directory or in windows\system directory.

2.4 Visual Basic .NET

You have to include the RECOBCRAPI.VB in your program.

Before to execute your application make sure the *RECOBCR.DLL* is available in your same .exe directory or in windows\system directory.

2.5 Delphi

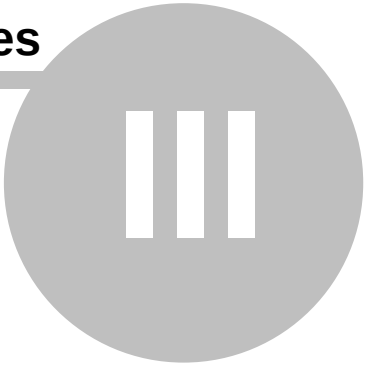
You have to include the RECOBCRAPI.PAS in your program.

Before to execute your application make sure the *RECOBCR.DLL* is available in your same .exe directory or in windows\system directory.

2.6 Java

You have to use 32 bit JVM and you have to include the RECOBCRAPI.JAVA in your program. Before to execute your application make sure the *RECOBCR.DLL* is available in your same .jar directory.

API References



3 API References

3.1 BCR_Init

C/C++ Declaration

```
__stdcall long BCR_Init(char* Name, char* Key);
```

C# Declaration

```
int BCR_Init(string Name, string Key);
```

Visual Basic Declaration

```
Function BCR_Init(ByVal Name As String, ByVal Key
As String) As Integer
```

Visual Basic .NET Declaration

```
Function BCR_Init(ByVal Name As String, ByVal Key
As String) As Integer
```

Delphi Declaration

```
function BCR_Init(Company:PAnsiChar;
LicenseKey:PAnsiChar):Integer; stdcall;
```

Java Declaration

```
int BCR_Init(String Name, String Key);
```

Description

This is the first function to call: initialize the library and returns a session handle to use in next calls. When you buy the library you receive an "user" and a "password" string necessary to initialize the library in normal mode: without this value or with wrong values the library is initialized in evaluation mode. The evaluation mode works exactly as normal mode but some time, when you call the recognition function, is displayed a warning dialog box remembering the evaluation state: you can close it and continue to work with no problems.

Parameters

User (in) - then user name string

Password (in) - then password string

Return values

The session handle if the library is initialized, 0 otherwise.

Note

You should use BCR_Init at application startup and BCR_Done at application closing and not every time you need to recognize an image.

The “status” string, shown in the windows dialog, is built by 1 and 0 meaning OK or KO for this checking in the order from left to right:

- SDK initialized with valid user/key
- License file integrity
- License file matching sdk version
- Computer verification done or not required
- Dongle verification done or not required
- Date expiration done or not required
- SDK option used unlocked

3.2 BCR_Done

C/C++ Declaration

```
__stdcall void BCR_Done(long Session);
```

C# Declaration

```
void BCR_Done(int Session);
```

Visual Basic Declaration

```
Sub BCR_Done(ByVal Session As Integer)
```

Visual Basic .NET Declaration

```
Sub BCR_Done(ByVal Session As Integer)
```

Delphi Declaration

```
procedure BCR_Done(SessionHandle:Integer);
stdcall;
```

Java Declaration

```
void BCR_Done(int SessionHandle);
```

Description

This is the last function to call when you don't need more services from the library: de-initialize the library and free all used resources.

Parameters

SessionHandle (in) - the session handle to free

Return values

n/a

Note

You should use BCR_Init at application startup and BCR_Done at application closing and not every time you need to recognize an image.

3.3 BCR_Recognize

C/C++ Declaration

```
__stdcall long BCR_Recognize(long SessionHandle,
long DIBIn, long CodeType, long ExpectedCodes,
long MinCodeLen, long MaxCodeLen, long
VirtualLaserSize, long VirtualLaserStep);
```

C# Declaration

```
int BCR_Recognize(int SessionHandle, int DIBIn,
int CodeType, int ExpectedCodes, int MinCodeLen,
int MaxCodeLen, int VirtualLaserSize, int
VirtualLaserStep);
```

Visual Basic Declaration

```
Function BCR_Recognize(Byval SessionHandle As
integer, Byval DIBIn As integer, Byval CodeType As
integer, Byval ExpectedCodes As integer, Byval
MinCodeLen As integer, Byval MaxCodeLen As
integer, Byval VirtualLaserSize As integer, Byval
```

```
VirtualLaserStep As integer) As integer
```

Visual Basic .NET Declaration

```
Function BCR_Recognize(Byval SessionHandle As
integer, Byval DIBIn As integer, Byval CodeType As
integer, Byval ExpectedCodes As integer, Byval
MinCodeLen As integer, Byval MaxCodeLen As
integer, Byval VirtualLaserSize As integer, Byval
VirtualLaserStep As integer) As integer
```

Delphi Declaration

```
Function BCR_Recognize(SessionHandle, DIBIn,
CodeType, ExpectedCodes, MinCodeLen, MaxCodeLen,
VirtualLaserSize, VirtualLaserStep):Integer;
stdcall;
```

Java Declaration

```
int BCR_Recognize(int SessionHandle,int DIBIn, int
CodeType, int ExpectedCodes,int MinCodeLen, int
MaxCodeLen, int VirtualLaserSize, int
VirtualLaserStep);
```

Note

See BCR_RecognizeEx
This function is available only for compatibility with old version.

3.4 BCR_RecognizeArea

C/C++ Declaration

```
long BCR_RecognizeArea(long Session, long
DIBHandle, long Left, long Top, long Right, long
Bottom, long CodeType, long MaxCodes, long MinLen,
long MaxLen, long LaserSize , long LaserStep, long
Quality, long Orientation, long VerifyCheckDigit);
```

C# Declaration

```
int BCR_RecognizeArea(int Session, int DIBHandle,
int Left, int Top, int Right, int Bottom, int
CodeType, int MaxCodes, int MinLen, int MaxLen,
int LaserSize , int LaserStep, int Quality, int
```

```
Orientation, int VerifyCheckDigit);
```

Visual Basic Declaration

```
Function BCR_RecognizeArea(ByVal Session As
Integer, ByVal DIBHandle As Integer, ByVal Left As
Integer, ByVal Top As Integer, ByVal Right As
Integer, ByVal Bottom As Integer, ByVal CodeType
As Integer, ByVal MaxCodes As Integer, ByVal
MinLen As Integer, ByVal MaxLen As Integer, ByVal
LaserSize As Integer, ByVal LaserStep As Integer,
ByVal Quality As Integer, ByVal Orientation As
Integer, ByVal VerifyCheckDigit As Integer) As
Integer
```

Visual Basic .NET Declaration

```
Function BCR_RecognizeArea(ByVal Session As
Integer, ByVal DIBHandle As Integer, ByVal Left As
Integer, ByVal Top As Integer, ByVal Right As
Integer, ByVal Bottom As Integer, ByVal CodeType
As Integer, ByVal MaxCodes As Integer, ByVal
MinLen As Integer, ByVal MaxLen As Integer, ByVal
LaserSize As Integer, ByVal LaserStep As Integer,
ByVal Quality As Integer, ByVal Orientation As
Integer, ByVal VerifyCheckDigit As Integer) As
Integer
```

Delphi Declaration

```
function BCR_RecognizeArea(Session, DIBHandle,
Left, Top, Right, Bottom, CodeType, MaxCodes,
MinLen,  MaxLen, LaserSize , LaserStep, Quality,
Orientation, VerifyCheckDigit):Integer; stdcall;
```

Java Declaration

```
int BCR_RecognizeArea(int Session, int DIBHandle,
int Left, int Top, int Right, int Bottom, int
CodeType, int MaxCodes, int MinLen, int MaxLen,
int LaserSize , int LaserStep, int Quality, int
Orientation, int VerifyCheckDigit);
```

Description

This is the function performing the recognition of the barcode in a defined area: you have to call this function after library initialization to perform the barcode recognition. The DIB handle can contain a single barcodes or multiple barcodes.

Parameters

SessionHandle (in) - the session handle to use
DIBIn (in) - the handle of the monochrome DIB to recognize
Left (in) - the left barcode coordinates
Top (in) - the top barcode coordinates
Right (in) - the right barcode coordinates
Bottom (in) - the bottom barcode coordinates
CodeType (in) - the types of barcodes to recognize: can be a combinantion (or) from 1 =2of5Interlaved, 2=2of5IATA, 4=2of5Industrial, 8=2of5Matrix, 16=2of5DataLogic, 32=Code39, 64=PharmaCode, 128=EAN13, 256=EAN8, 512=UPCA, 1024=UPCE, 2048=EXT2, 4096=EXT5, 8192=CodaBar, 16384=MSI, 32768=Code11, 65536=Code128, 131072=Code93, 262144=Code39Extended
MaxCodes (in) - the maximum number of expected barcodes: the system stop to search more codes if this number of codes is recognized
MinLen (in) - the minumum length in characters (including start/stop characters and check digit) expected for a barcode.
MaxLen (in) - the maximum length in characters (including start/stop characters and check digit) expected for a barcode.
LaserSize(in) - the height of virtual laser used to scan the image. The system will process together a number of contiguous rows equal to this value. Use 3 as default.
LaserStep(in) - the step of virtual laser used to scan the image. The system will skip between to consecutive scans a number of rows equal to this value. Use 32 as default.
Quality(in) - the quality expected for the processing image: can be 1 = Normal, 2=Dark, 3=Light
Orientation(in) - the expected orientation of barcodes: can be a combination (or) from 1 = LeftToRight, 2=RightToLeft, 3=TopToBottom, 4=BottomToTop
VerifyCheckDigit (in) - the flag allowing to enable (1) or disable (0) the verification of barcode check digit. When disabled, if the barcode has a check digit, it will be returned with the barcode data.

Return values

The number of barcodes recognized if success, 0 otherwise.

Notes

The library works with standard DIBs images. Refer to Microsoft documentation for additional info about Device Indipendent Bitmaps. The barcode recognition is executed on b&w images, so if you use a grayscale or color image the library automatically binarize it before to perform processing. You could also consider to use our *Dynamc Thresholding Library* to get better result in conversion.

3.5 BCR_RecognizeEx

C/C++ Declaration

```
__stdcall long BCR_RecognizeEx(long SessionHandle,
long DIBIn, long CodeType, long ExpectedCodes,
long MinCodeLen, long MaxCodeLen, long
VirtualLaserSize, long VirtualLaserStep, long
ImageQuality, long Orientation, long
CheckDigitVerification);
```

C# Declaration

```
int BCR_RecognizeEx(int SessionHandle, int DIBIn,
int CodeType, int ExpectedCodes, int MinCodeLen,
int MaxCodeLen, int VirtualLaserSize, int
VirtualLaserStep, int ImageQuality, int
Orientation, int CheckDigitVerification);
```

Visual Basic Declaration

```
Function BCR_RecognizeEx(Byval SessionHandle As
Integer, Byval DIBIn As Integer, Byval CodeType As
Integer, Byval ExpectedCodes As Integer, Byval
MinCodeLen As Integer, Byval MaxCodeLen As
Integer, Byval VirtualLaserSize As Integer, Byval
VirtualLaserStep As Integer, Byval ImageQuality As
Integer, Byval Orientation As Integer, Byval
CheckDigitVerification)As Integer
```

Visual Basic .NET Declaration

```
Function BCR_RecognizeEx(Byval SessionHandle As
Integer, Byval DIBIn As Integer, Byval CodeType As
Integer, Byval ExpectedCodes As Integer, Byval
MinCodeLen As Integer, Byval MaxCodeLen As
```

```
Integer, Byval VirtualLaserSize As Integer, Byval
VirtualLaserStep As Integer, Byval ImageQuality As
Integer, Byval Orientation As Integer, Byval
CheckDigitVerification)As Integer
```

Delphi Declaration

```
function BCR_RecognizeEx(SessionHandle, DIBIn,
CodeType, ExpectedCodes, MinCodeLen, MaxCodeLen,
VirtualLaserSize, VirtualLaserStep,  ImageQuality,
Orientation, CheckDigitVerification):Integer;
stdcall;
```

Java Declaration

```
int BCR_RecognizeEx(int SessionHandle,int DIBIn,
int CodeType, int ExpectedCodes,int MinCodeLen,
int MaxCodeLen, int VirtualLaserSize, int
VirtualLaserStep, int   ImageQuality, int
Orientation, int  CheckDigitVerification);
```

Description

This is the function performing the recognition: you have to call this function after library initialization to perform the barcode recognition. The DIB handle can contain a single barcodes or multiple barcodes.

Parameters

SessionHandle (in) - the session handle to use
DIBIn (in) - the handle of the monochrome DIB to recognize
CodeType (in) - the types of barcodes to recognize: can be a combinantion (or) from 1 =2of5Interlaved, 2=2of5IATA, 4=2of5Industrial, 8=2of5Matrix, 16=2of5DataLogic, 32=Code39, 64=PharmaCode, 128=EAN13, 256=EAN8, 512=UPCA, 1024=UPCE, 2048=EXT2, 4096=EXT5, 8192=CodaBar, 16384=MSI, 32768=Code11, 65536=Code128, 131072=Code93, 262144=Code39Extended
ExpectedCodes (in) - the maximum number of expected barcodes: the system stop to search more codes if this number of codes is recognized
MinCodeLen (in) - the minumum length in characters (including start/stop characters and check digit) expected for a barcode.
MaxCodeLen (in) - the maximum length in characters (including start/stop characters and check digit) expected for a barcode.
VirtualLaserSize(in) - the height of virtual laser used to scan the

image. The system will process together a number of contiguous rows equal to this value. Use 3 as default.
VirtualLaserStep(in) - the step of virtual laser used to scan the image. The system will skip between to consecutive scans a number of rows equal to this value. Use 32 as default.
ImageQuality(in) - the quality expected for the processing image: can be 1 = Normal, 2=Dark, 3=Light
Orientation(in) - the expected orientation of barcodes: can be a combination (or) from 1 = LeftToRight, 2=RightToLeft, 3=TopToBottom, 4=BottomToTop
CheckDigitVerification (in) - the flag allowing to enable (1) or disable (0) the verification of barcode check digit. When disabled, if the barcode has a check digit, it will be returned with the barcode data.

Return values

The number of barcodes recognized if success, 0 otherwise.

Notes

The library works with standard DIBs images. Refer to Microsoft documentation for additional info about Device Indipendent Bitmaps. The barcode recognition is executed on b&w images, so if you use a grayscale or color image the library automatically binarize it before to perform processing. You could also consider to use our *Dynamc Thresholding Library* to get better result in conversion.

3.6 BCR_GetCodesCount

C/C++ Declaration

```
__stdcall long BCR_GetCodesCount(long
SessionHandle);
```

C# Declaration

```
int BCR_GetCodesCount(int SessionHandle);
```

Visual Basic Declaration

```
Function BCR_GetCodesCount(ByVal SessionHandle As
Integer)As Integer;
```

Visual Basic .NET Declaration

```
Function BCR_GetCodesCount(ByVal SessionHandle As
Integer)As Integer;
```

Delphi Declaration

```
Function
BCR_GetCodesCount(SessionHandle:Integer):Integer;
stdcall;
```

Java Declaration

```
int BCR_GetCodesCount(int SessionHandle);
```

Description

This is the function to call after the recognition process to get the number of codes recognized in the image.

Parameters

SessionHandle (in) - the session handle to use

Return values

The numbers of recognized codes

3.7 BCR_GetCodeLen

C/C++ Declaration

```
__stdcall long BCR_GetCodeLen(long SessionHandle,
long CodeIndex);
```

C# Declaration

```
int BCR_GetCodeLen(int SessionHandle, int
CodeIndex);
```

Visual Basic Declaration

```
Function BCR_GetCodeLen(Byval SessionHandle As
Integer, Byval CodeIndex As Integer)As Integer;
```

Visual Basic .NET Declaration

```
Function BCR_GetCodeLen(Byval SessionHandle As
Integer, Byval CodeIndex As Integer)As Integer;
```

Delphi Declaration

```
Function BCR_GetCodeLen(SessionHandle:Integer,
CodeIndex:Integer):Integer;stdcall;
```

Java Declaration

```
int BCR_GetCodeLen(int SessionHandle,int
CodeIndex);
```

Description

Retrieves the ASCII text length of one of the codes recognized.

Parameters

SessionHandle (in) - the session handle to use
CodeIndex (in) - the index of the code . First code recognized is 0,last is the BCR_GetCodesCount-1.

Return values

The length of code text.

3.8 BCR_GetCodeText

C/C++ Declaration

```
__stdcall long BCR_GetCodeText(long SessionHandle,
long CodeIndex, char *CharBuffer);
```

C# Declaration

```
int BCR_GetCodeText(int Session, int CodeIndex,
Buffer As String)
```

Visual Basic Declaration

```
Function BCR_GetCodeText(ByVal Session As Integer,
ByVal CodeIndex As Integer, ByVal Buffer As
String) As Integer
```

Visual Basic .NET Declaration

```
Function BCR_GetCodeText(ByVal Session As Integer,
ByVal CodeIndex As Integer, ByVal Buffer As
String) As Integer
```

Delphi Declaration

```
function
BCR_GetCodeText(SessionHandle:Integer;ResultIndex:
Integer;Buffer:PChar):Integer; stdcall;
```

Java Declaration

```
int BCR_GetCodeText(int SessionHandle,int
CodeIndex, Memory CharBuffer);
```

Description

This is the function to call after the recognition process to get the data recognized.

Parameters

SessionHandle (in) - the session handle to use
CodeIndex (in) - the index of the code . First code recognized is 0, last is the BCR_CodesCount-1.
CharBuffer (out) - the buffer where will be moved the characters recognized. You have to allocate enough space.

Return values

The number of characters moved in the buffer.

Notes

The library works with standard DIBs images. Refer to Microsoft documentation for additional info about Device Indipendent Bitmaps. If you have to recognize a gray-scale image, please consider to use our *Dynamc Thresholding Library* to get better result in conversion.

3.9 BCR_GetCodeArea

C/C++ Declaration

```
__stdcall void BCR_GetCodeRect(long SessionHandle,
long CodeIndex, long *Left, long *Top, long
*Right, long *Bottom);
```

C# Declaration

```
void BCR_GetCodeRect(int SessionHandle, int
CodeIndex, int *Left, int *Top, int *Right, int
*Bottom);
```

Visual Basic Declaration

```
Sub BCR_GetCodeRect (ByVal Session As Integer,
ByVal CodeIndex As Integer, ByRef Left As Integer,
ByRef Top As Integer, ByRef right As Integer,
ByRef bottom As Integer)
```

Visual Basic .NET Declaration

```
Sub BCR_GetCodeRect (ByVal Session As Integer,
ByVal CodeIndex As Integer, ByRef Left As Integer,
ByRef Top As Integer, ByRef right As Integer,
ByRef bottom As Integer)
```

Delphi Declaration

```
Procedure BCR_GetCodeRect
(Session:Integer,CodeIndex:Integer, var
Left:Integer, var Top:Integer, var right:Integer,
var bottom:Integer);stdcall;
```

Java Declaration

```
void     BCR_GetCodeRect(int     SessionHandle,int
CodeIndex,  int  Left,  int  Top,  int  Right,int
Bottom);
```

Description

Retrieves the rectangle coordinates of one of the codes recognized.

Parameters

SessionHandle (in) - the session handle to use
CodeIndex (in) - the index of the code . First code recognized is 0, last is the BCR_GetCodesCount-1.
Left (out) - the buffer where will be moved the left position of the rectangle containing the barcode.
Top (out) - the buffer where will be moved the top position of the rectangle containing the barcode.
Right (out) - the buffer where will be moved the right position of the rectangle containing the barcode.
Bottom (out) - the buffer where will be moved the bottom position of the rectangle containing the barcode.

Return values

None

3.10 BCR_GetCodeCheck

C/C++ Declaration

```
__stdcall long BCR_GetCodeCheck(long
SessionHandle, long CodeIndex);
```

C# Declaration

```
int BCR_GetCodeCheck(int SessionHandle, int
CodeIndex);
```

Visual Basic Declaration

```
Function BCR_GetCodeCheck(ByVal SessionHandle As
Integer, ByVal CodeIndex As Integer)As Integer;
```

Visual Basic .NET Declaration

```
Function BCR_GetCodeCheck(ByVal SessionHandle As
Integer, ByVal CodeIndex As Integer)As Integer;
```

Delphi Declaration

```
Function BCR_GetCodeCheck(SessionHandle:Integer,
CodeIndex:Integer):Integer;stdcall;
```

Java Declaration

```
int BCR_GetCodeCheck(int SessionHandle, int
CodeIndex);
```

Description

Retrieves the status of the check digit present in the barcode.

Parameters

SessionHandle (in) - the session handle to use
CodeIndex (in) - the index of the code . First code recognized is 0, last is the BCR_GetCodesCount-1.

Return values

0 if check digits invalid or was not found, 1 if check digit found and valid.

3.11 BCR_GetCodeType

C/C++ Declaration

```
long BCR_GetCodeType(long SessionHandle, long
CodeIndex);
```

C# Declaration

```
int BCR_GetCodeType(int  Session , int  CodeIndex
) ;
```

Visual Basic Declaration

```
Function BCR_GetCodeType (ByVal Session As Long,
ByVal CodeIndex As Long) As Long
```

Visual Basic .NET Declaration

```
Function BCR_GetCodeType(ByVal Session As Integer,
ByVal CodeIndex As Integer) As Integer
```

Delphi Declaration

```
function
BCR_GetCodeType(SessionHandle:Integer;ResultIndex:
Integer):Integer;
```

Java Declaration

```
int BCR_GetCodeType(int SessionHandle, int
ResultIndex);
```

Description

retrieve the type of recognized barcode

Parameters

SessionHandle (in) - the session handle to use
CodeIndex (in) - the index of the code . First code recognized is 0, last is the BCR_GetCodesCount-1.

Return values

The code type is defined as follow:
1 = BCR_2of5Interlaved
2 = BCR_2of5IATA
4 = BCR_2of5Industrial
8 = BCR_2of5Matrix
16 = BCR_2of5DataLogic
32 = BCR_Code39
64 = BCR_PharmaCode
128 = BCR_EAN13
256 = BCR_EAN8
512 = BCR_UPCA
1024 = BCR_UPCE
2048 = BCR_EXT2
4096 = BCR_EXT5
8192 = BCR_CodaBar
16384 = BCR_MSI
32768 = BCR_Code11
65536 = BCR_Code128
131072 = BCR_Code93
262144 = BCR_Code39Extended

3.12 BCR_GetCodeOrientation

C/C++ Declaration

```
__stdcall long BCR_GetCodeOrientation(long
SessionHandle, long CodeIndex);
```

C# Declaration

```
int BCR_GetCodeOrientation(int SessionHandle, int
CodeIndex);
```

Visual Basic Declaration

```
Function BCR_GetCodeOrientation(ByVal
SessionHandle As Integer, ByVal CodeIndex As
Integer)As Integer;
```

Visual Basic .NET Declaration

```
Function BCR_GetCodeOrientation(ByVal
SessionHandle As Integer, ByVal CodeIndex As
Integer)As Integer;
```

Delphi Declaration

```
Function
BCR_GetCodeOrientation(SessionHandle:Integer,
CodeIndex:Integer):Integer;stdcall
```

Java Declaration

```
int BCR_GetCodeOrientation(int SessionHandle, int
CodeIndex);
```

Description

Retrieves the orientation of the recognized barcode.

Parameters

SessionHandle (in) - the session handle to use
CodeIndex (in) - the index of the code . First code recognized is 0, last is the BCR_GetCodesCount-1.

Return values

1 if code is oriented LeftToRight, 2 if oriented RighToLeft, 4 if oriented TopToBottom, 8 if oriented BottomToTop.

3.13 BCR_LoadImage

C/C++ Declaration

```
__stdcall long BCR_LoadImage(long SessionHandle,
char* FileName);
```

C# Declaration

```
int BCR_LoadImage(int SessionHandle, string
FileName);
```

Visual Basic Declaration

```
Function BCR_LoadImage(ByVal Session As Integer,
ByVal FileName As String) As Integer
```

Visual Basic .NET Declaration

```
Function BCR_LoadImage(ByVal Session As Integer,
ByVal FileName As String) As Integer
```

Delphi Declaration

```
Function BCR_LoadImage(Session:Integer,
FileName:PAnsiChar):Integer;stdcall;
```

Java Declaration

```
int BCR_LoadImage(int SessionHandle,String
FileName);
```

Description

This function allows to load an image from file obtaining a DIB handle. Supported files formats are TIF, JPG, PNG and BMP

Parameters

SessionHandle (in) - the session handle to use
FileName (in) - the name of the file to load

Return values

The DIB handle with the image inside the file, 0 if an error occurred.

3.14 BCR_ImportDDB

C/C++ Declaration

```
__stdcall long BCR_ImportDDB (long SessionHandle,
HBITAMP BitmapHandle, HPALETTE PaletteHandle);
```

C# Declaration

```
int BCR_ImportDDB (int SessionHandle, int
BitmapHandle, int PaletteHandle);
```

Visual Basic Declaration

```
Function BCR_ImportDDB(ByVal Session As Integer,
ByVal BitmapHandle As Integer, ByVal PaletteHandle
As Integer) As Integer
```

Visual Basic .NET Declaration

```
Function BCR_ImportDDB(ByVal Session As Integer,
ByVal BitmapHandle As Integer, ByVal PaletteHandle
As Integer) As Integer
```

Delphi Declaration

```
Function BCR_ImportDDB(Session:Integer,
BitmapHandle:Integer,
PaletteHandle:Integer):Integer;stdcall;
```

Java Declaration

```
int BCR_ImportDDB (int SessionHandle,int
BitmapHandle, int PaletteHandle);
```

Description

This function allows to import a DDB (Device Dependant Bitmap) from file obtaining a DIB handle.

Parameters

SessionHandle (in) - the session handle to use
BitmapHandle (in) - the handle of DDB to import
PaletteHandle (in) - the handle of DDB palette to import. Passing 0 it will be used a standard palette.

Return values

The DIB handle with the image inside, 0 if an error occurred.

3.15 BCR_FreeImage

C/C++ Declaration

```
__stdcall void BCR_FreeImage(long SessionHandle,
long DIBHandle);
```

C# Declaration

```
void BCR_FreeImage(int SessionHandle, int
DIBHandle);
```

Visual Basic Declaration

```
Sub BCR_FreeImage(ByVal SessionHandle As Integer,
ByVal DIBHandle As Integer);
```

Visual Basic .NET Declaration

```
Sub BCR_FreeImage(ByVal SessionHandle As Integer,
ByVal DIBHandle As Integer);
```

Delphi Declaration

```
Procedure BCR_FreeImage(SessionHandle:Integer,
DIBHandle:Integer):Integer;stdcall;
```

Java Declaration

```
void      BCR_FreeImage(int      SessionHandle,int
DIBHandle);
```

Description

This function allows to remove from memory an image previously loaded from file or imported from DDB.

Parameters

SessionHandle (in) - the session handle to use
CharIndex (in) - the index of the character . First character is 0, last

is the ICR_GetCharactersCount-1.
CharAlternative (in) the index of the alternative, from 0 (hi confidence) to 2 (low confidence).
Left (out) - the buffer where will be moved the left position of the rect containg the character.
Top (out) - the buffer where will be moved the top position of the rect containg the character.
Right (out) - the buffer where will be moved the righr position of the rect containg the character.
Bottom (out) - the buffer where will be moved the bottom position of the rect containg the character.

Return values

n/a

3.16 LoadBCRLibrary

C/C++ Declaration

```
RECOBCR_API void LoadBCRLibrary(void)
```

Description

Load the BCR DLL library: you have to use this function one time before to use other API functions from Visual C++.

This function is not required using the API from Delphi or Visual Basic

3.17 FreeBCRLibrary

C/C++ Declaration

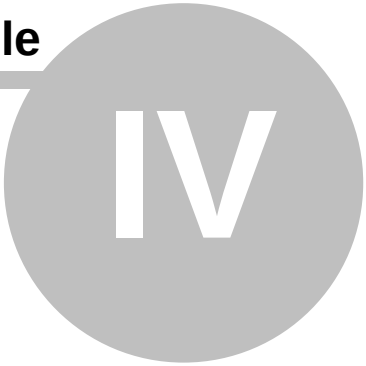
`RECOIO_API void FreeIOLibrary(void)`

Description

Unload the BCR DLL library: you have to use this function one time before to exit from your Visual C++ application.

This function is not required using the API from Delphi or Visual Basic

Sample



4 Sample

4.1 Samples

4.1.1 Visual C++

```
This sample code performs barcode recognition on image available on cli
int RecognizeBarcodesOnClipboard()
{
 // Load dynamically the library
 LoadBCRLibrary();
 // Init the BCR session
 int Session= BCR_Init("demo", "demo");
 long hBitmap;
 char RecognizedString[255];
 // Check if a DIB is available in clipboard
 bool bAvail=IsClipboardFormatAvailable(CF_DIB);
 hBitmap = 0;
 if (bAvail)
 {
  // Open the Clipboard
  ::OpenClipboard(NULL);
  // Retrieve the DIB from clipboard
  hBitmap = (long)
  GetClipboardData(CF_DIB);
  // Recognize barcodes Code39 or 2of5i, 1 per page, from 5 to 15 chara
  // with laser size 3, laser step 32,image quality normal, image orien
  int nCodes= BCR_RecognizeEx(Session,(long) hBitmap, 32 or 1, 1, 5, 15
  // Close the Clipboard
  ::CloseClipboard();
  // Check if there are recognized chars
  if (nCodes>0) {
    // Retrieve the first barcode recognized characters
    nChar = BCR_GetCodeText(Session, 0, RecognizedString);
    // Make the buffer a null terminated string
    RecognizedString[nChar]=0;
    // Show the recognized characters
    MessageBox(NULL,
    RecognizedString, "RESULT", MB_OK);
  }
  // Show an error message
  else MessageBox(NULL, "No barcodes recognized !", "ERROR", MB_OK);
 }
 // Show an error message
 else MessageBox(NULL, "Unable to paste DIB","ERROR", MB_OK);
 // Close the session
 BCR_Done(Session);
 // Unload the library
 FreeBCRLibrary();
 return 0;
}
```

4.1.2 Visual Basic

```
This sample code performs barcode recognition on image available on cli
Private RecognizeBarcodesOnClipboard()
' Declare some variables
Dim DIBHandle As Long
Dim CodeStr As String
Dim Codes As Long
Dim Code As Long
Dim Dummy As Long
Dim Orientation As Long
Dim CodeString As String
Dim CodeLen As Long
Dim BCRSession As Long

' open the session
BCRSession = BCR_Init("demo", "demo")
' If there's a DIB on the clipboard, get it
If Clipboard.GetFormat(vbCFDIB) Then
OpenClipboard (hwnd)
DIBHandle = GetClipboardData(vbCFDIB)
' Try to recognize barcodes
' Code39 or 2of5i
' max 4 per page
' from 3 to 16 characters
' virtual laser size 2, virtual laser step 1
' normal image quality
' orientation left to right or right to left
' check digit verification enabled
Codes = BCR_RecognizeEx(BCRSession, DIBHandle,
BCR_Code39 Or BCR_Code2of5i, 4, 3, 16, 2, 10,
BCR_NORMAL_QUALITY, BCR_LeftToRight Or
BCR_RightToLef, BCR_CHECKDIGIT_ENABLED)
' Check if at least a code was recognized
If Codes > 0 Then
 For Code = 0 To Codes - 1
  ' retrieve the len of code
  CodeLen = BCR_GetCodeLen(BCRSession, Code)
  ' allocate space for the code in a string
  CodeStr = Space(CodeLen)
  ' retrieve the code data
  Dummy = BCR_GetCodeText(BCRSession, Code, CodeStr)
  Orientation = BCR_GetCodeOrientation(BCRSession, Code)
  ' show the code data
  MsgBox ("Recognized codes " & Str(Code + 1) & " of " & Str(Codes) & "
 Next Code
Else
 MsgBox ("No code recognized !")
End If
CloseClipboard
Else
MsgBox ("No BMP in clipboard !")
End If
' close the session
BCR_Done (BCRSession)
End Sub
```

4.1.3 Delphi

```
This sample code performs barcode recognition on image available on cli
procedure RecognizeBarcodesOnClipboard;
Var DIBHandle:Integer;
        BCRSession:Integer;
        BarCodes:Integer;
        Buffer:PChar;
        BarType:Integer;
        BarOrientation:Integer;
        i:integer;
begin
        // Open the BCR session
        BCRSession:=BCR_Init('demo','demo');
        // Check if an image is in clipboard
        if Clipboard.HasFormat(CF_DIB) then
        begin
          // Open the clipboard
          Clipboard.Open;
          // Retrieve the dib
          DIBHandle:=Clipboard.GetAsHandle(CF_DIB);
          // Recognize the barcode
          // Type: Code39, I2of5
          // Max expected codes: 4
          // Codes length: from 3 to 20 characters
          // Virtual laser step/size: 2/10
          // Image Quality: normal
          // Orientation: LeftToRight or RightToLeft
          // CheckDigit Verification: enabled
          BarCodes:=BCR_RecognizeEx(BCRSession,DIBHandle,BCR_Code39 or B
          // Show the number of recognized barcodes
          ShowMessage('Barcodes recognized:'+IntToStr(BarCodes));
          // Retrieve info for each barcode recognized
          for i:=0 to BarCodes-1 do
          begin
                // Create the buffer for the result
                GetMem(Buffer,255);
                // Retrieve the recognized text
                BCR_GetCodeText(BCRSession,i,Buffer);
                // Retrieve the code type
                BarType:=BCR_GetCodeType(BCRSession,i);
                // Retrieve the code orientation
                BarOrientation:=BCR_GetCodeOrientation(BCRSession,i);
                // Show the code index, data and orientation
                ShowMessage('Barcode '+IntToStr(i+1)+' of'+IntToStr(BarC
                // Free the buffer
                FreeMem(Buffer);
          end;
          // Close the clipboard
          Clipboard.Close;
        end else ShowMessage('No image available in clipboard !');
        // Close the BCR session
        BCR_Done(BCRSession);
end;
```

Annotation