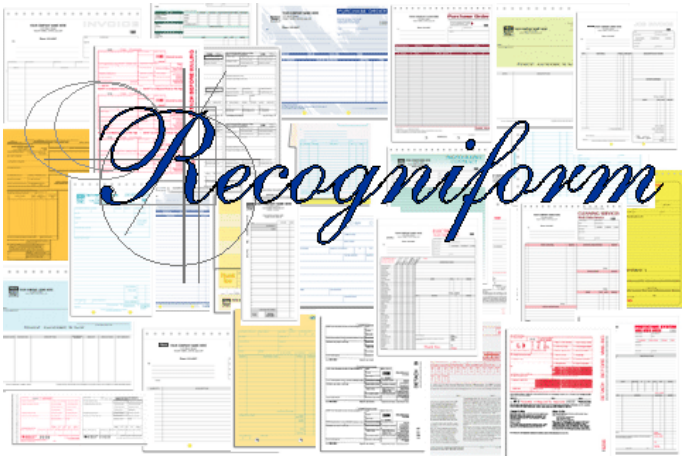


Software Development Kit

ICR

Recogniform Technologies SpA

HOW TO CONTACT US

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

Phone : +39 0984 404174

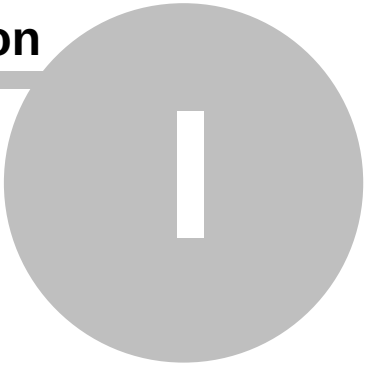
Fax : +39 0984 830299

Internet : www.recogniform.com

E-Mail : info@recogniform.com

Table of contents

Introduction



1 Introduction

1.1 Copyright

The software and the documentation are property of:

Recogniform Technologies SpA
Contrada Concistocchi
87036 Rende (CS)
Italy
www.recogniform.com
info@recogniform.com

1.2 License

1.3 Overview

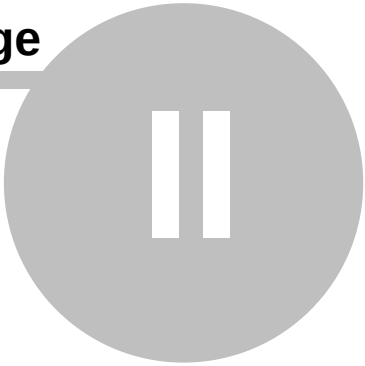
The library allows to recognize handwritten text from images acquired using a scanner. The basic character set supported can be different in function of recognizer used. A recognizer is a .rec file containing the knowledgement base used by the engine to read a character set: currently are available the following recognizers:

`digit.rec` allows to recognize digits written in international style
`eurodigit.rec` allows to recognize digits written in european style
`lower.rec` allows to recognize lowercase characters written in european and international style
`upper.rec` allows to recognize uppercase characters written in european and international style
`upperlow.rec` allows to recognize lowercase and uppercase characters written in european and international style
`alphanum.rec` allows to recognize digits, lowercase and uppercase characters written in european and international style

We can create specialized and customized recognizers for specific use.

The recognition process is fast and accurate and you can get too the confidence and the rectangle area of each recognized character.

Usage



2 Usage

2.1 Visual C++

You have to include the RECOICRAPI.C in your program. Before to execute your application make sure the *RECOICR.DLL* is available in your same .exe directory or in windows\system directory.

2.2 C#

You have to include the RECOICRAPI.CS in your program. Before to execute your application make sure the *RECOICR.DLL* is available in your same .exe directory or in windows\system directory.

2.3 Visual Basic

You have to include the RECOICRAPI.BAS in your program. Before to execute your application make sure the *RECOICR.DLL* is available in your same .exe directory or in windows\system directory.

2.4 Visual Basic .NET

You have to include the RECOICRAPI.VB in your program. Before to execute your application make sure the *RECOICR.DLL* is available in your same .exe directory or in windows\system directory.

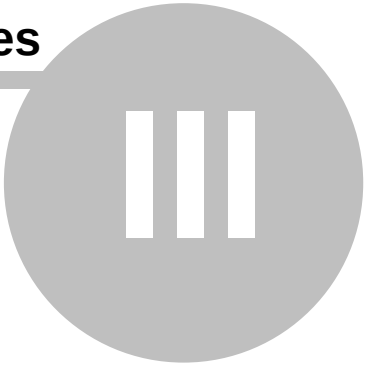
2.5 Java

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

2.6 Delphi

You have to include the RECOICRAPI.PAS in your program. Before to execute your application make sure the *RECOICR.DLL* is available in your same .exe directory or in windows\system directory.

API References



3 API References

3.1 ICR_Init

C/C++ Declaration

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

C# Declaration

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

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

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

Java Declaration

```
int ICR_Init(String User, String Password);
```

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 wrang 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

Passwors (in) - then password string

Return values

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

3.2 ICR_Done

C/C++ Declaration

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

C# Declaration

```
void ICR_Done(int Session);
```

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

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

Java Declaration

```
void ICR_Done(int Session);
```

Description

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

Parameters

SessionHandle (in) - the session handle to free

Return values

n/a

3.3 ICR_SelectRecognizer

C/C++ Declaration

```
__stdcall void ICR_SelectRecognizer(long Session,
char* RecFile);
```

C# Declaration

```
void ICR_SelectRecognizer(int Session, string
RecFile);
```

Visual Basic Declaration

```
Sub ICR_SelectRecognizer(ByVal Session As Integer,
ByVal RecFile As String)
```

Visual Basic .NET Declaration

```
Sub ICR_SelectRecognizer(ByVal Session As Integer,
ByVal RecFile As String)
```

Delphi Declaration

```
procedure
ICR_SelectRecognizer(SessionHandle:Integer;
RecognizerFile:PAnsiChar); stdcall;
```

Java Declaration

```
void ICR_SelectRecognizer (int Session, String
RecognizerFile);
```

Description

This function allows to select the recognizer to use: see the overview section for more details.

Parameters

SessionHandle (in) - the session handle to use
RecFileName (in) - the full name of the recognizer file (.rec)

Return values

n/a

3.4 ICR_RecognizeChar

C/C++ Declaration

```
__stdcall long ICR_RecognizeChar(long Session,
long DIBIn, long Options);
```

C# Declaration

```
int ICR_RecognizeChar(int Session, int DIBIn, int
Options);
```

Visual Basic Declaration

```
Function ICR_RecognizeChar(ByVal Session As
Integer, ByVal DIBIn As Integer, ByVal Options As
Integer) As Integer
```

Visual Basic .NET Declaration

```
Function ICR_RecognizeChar(ByVal Session As
Integer, ByVal DIBIn As Integer, ByVal Options As
Integer) As Integer
```

Delphi Declaration

```
function ICR_RecognizeChar(SessionHandle:Integer;
DIBHandle:Integer; Options:Integer):Integer;
stdcall;
```

Java Declaration

```
int ICR_RecognizeChar (int Session, int DIBHandle,
int Options);
```

Description

This is the function performing the recognition: you have to call this function after library initialization to perform the optical character recognition. The DIB handle haves to contain a single character.

Parameters

SessionHandle (in) - the session handle to use
DIBIn (in) - the handle of the monochrome DIB to recognize
Options (in) - the options to use (not used at the moment)

Return values

The number of characters recognized if success, 0 otherwise.

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.

3.5 ICR_RecognizeWord

C/C++ Declaration

```
__stdcall long ICR_RecognizeWord(long Session,
long DIBIn, long Options);
```

C# Declaration

```
int ICR_RecognizeWord(int Session, int DIBIn, int
Options);
```

Visual Basic Declaration

```
Function ICR_RecognizeWord(ByVal Session As
Integer, ByVal DIBIn As Integer, ByVal Options As
Integer) As Integer
```

Visual Basic .NET Declaration

```
Function ICR_RecognizeWord(ByVal Session As
Integer, ByVal DIBIn As Integer, ByVal Options As
Integer) As Integer
```

Delphi Declaration

```
function ICR_RecognizeWord(SessionHandle:Integer;
DIBHandle:Integer; Options:Integer):Integer;
stdcall;
```

Java Declaration

```
int ICR_RecognizeWord (int Session, int DIBHandle,
int Options);
```

Description

This is the function performing the recognition: you have to call this function after library initialization to perform the optical character recognition. The DIB handle haves to contain a single word.

Parameters

SessionHandle (in) - the session handle to use
DIBIn (in) - the handle of the monochrome DIB to recognize
Options (in) - the options to use (not used at the moment)

Return values

The number of characters recognized if success, 0 otherwise.

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.

3.6 ICR_RecognizeLine

C/C++ Declaration

```
__stdcall long ICR_RecognizeLine(long Session,
long DIBIn, long Options);
```

C# Declaration

```
int ICR_RecognizeLine(int Session, int DIBIn, int
Options);
```

Visual Basic Declaration

```
Function ICR_RecognizeLine(ByVal Session As
Integer, ByVal DIBIn As Integer, ByVal Options As
```

```
Integer) As Integer
```

Visual Basic .NET Declaration

```
Function ICR_RecognizeLine(ByVal Session As
Integer, ByVal DIBIn As Integer, ByVal Options As
Integer) As Integer
```

Delphi Declaration

```
function ICR_RecognizeLine(SessionHandle:Integer;
DIBHandle:Integer; Options:Integer):Integer;
stdcall;
```

Java Declaration

```
int ICR_RecognizeLine (int Session, int DIBHandle,
int Options);
```

Description

This is the function performing the recognition: you have to call this function after library initialization to perform the optical character recognition. The DIB handle haves to contain a single line of text.

Parameters

SessionHandle (in) - the session handle to use
DIBIn (in) - the handle of the monochrome DIB to recognize
Options (in) - the options to use (not used at the moment)

Return values

The number of characters recognized if success, 0 otherwise.

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.

3.7 ICR_RecognizeParagraph

C/C++ Declaration

```
__stdcall long ICR_RecognizeParagraph(long
Session, long DIBIn, long Options);
```

C# Declaration

```
int ICR_RecognizeParagraph(int Session, int DIBIn,
int Options);
```

Visual Basic Declaration

```
Function ICR_RecognizeParagraph(ByVal Session As
Integer, ByVal DIBIn As Integer, ByVal Options As
Integer) As Integer
```

Visual Basic .NET Declaration

```
Function ICR_RecognizeParagraph(ByVal Session As
Integer, ByVal DIBIn As Integer, ByVal Options As
Integer) As Integer
```

Delphi Declaration

```
function
ICR_RecognizeParagraph(SessionHandle:Integer;
DIBHandle:Integer; Options:Integer):Integer;
stdcall;
```

Java Declaration

```
int ICR_RecognizeParagraph (int Session, int
DIBHandle, int Options);
```

Description

This is the function performing the recognition: you have to call this function after library initialization to perform the optical character recognition. The DIB handle haves to contain a text paragraph.

Parameters

SessionHandle (in) - the session handle to use
DIBIn (in) - the handle of the monochrome DIB to recognize

Options (in) - the options to use (not used at the moment)

Return values

The number of characters recognized if success, 0 otherwise.

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.

3.8 ICR_GetResultLen

C/C++ Declaration

```
__stdcall long ICR_GetResultLen(long Session);
```

C# Declaration

```
int ICR_GetResultLen(int Session);
```

Visual Basic Declaration

```
Function ICR_GetResultLen(ByVal Session As
Integer) As Integer
```

Visual Basic .NET Declaration

```
Function ICR_GetResultLen(ByVal Session As
Integer) As Integer
```

Delphi Declaration

```
function
ICR_GetResultLen(SessionHandle:Integer):Integer;
stdcall;
```

Java Declaration

```
int ICR_GetResultLen (int Session);
```

Description

This is the function to call after the recognition process to get the number of characters recognized in the image, spaces included

Parameters

SessionHandle (in) - the session handle to use

Return values

The numbers of recognized characters, spaces included

3.9 ICR_GetResultConfidence

C/C++ Declaration

```
__stdcall long ICR_GetResultConfidence(long
Session);
```

C# Declaration

```
int ICR_GetResultConfidence(int Session);
```

Visual Basic Declaration

```
Function ICR_GetResultConfidence(ByVal Session As
Integer) As Integer
```

Visual Basic .NET Declaration

```
Function ICR_GetResultConfidence(ByVal Session As
Integer) As Integer
```

Delphi Declaration

```
function
ICR_GetResultConfidence(SessionHandle:Integer):Int
eger; stdcall;
```

Java Declaration

```
int ICR_GetResultConfidence (int Session);
```

Description

This is the function to call after the recognition process to get the confidence attributed to the last recognition, in the range 0..100.

Parameters

SessionHandle (in) - the session handle to use

Return values

The confidence level from 0 to 100

3.10 ICR_GetResultValue

C/C++ Declaration

```
__stdcall long ICR_GetResultValue(long Session,
char* Buffer);
```

C# Declaration

```
void ICR_GetResultValue(int Session, StringBuilder
Buffer);
```

Visual Basic Declaration

```
Sub ICR_GetResultValue(ByVal Session As Integer,
ByVal Buffer As String)
```

Visual Basic .NET Declaration

```
Sub ICR_GetResultValue(ByVal Session As Integer,
ByVal Buffer As StringBuilder)
```

Delphi Declaration

```
function ICR_GetResultValue(SessionHandle:Integer;
Buffer:PAnsiChar):Integer; stdcall;
```

Java Declaration

```
int ICR_GetResultValue (int Session, Memory
Buffer);
```

Description

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

Parameters

SessionHandle (in) - the session handle to use
CharBuffer (out) - the buffer where will be moved the characters recognized. You have to allocate enough space.

Return values

n/a

3.11 ICR_GetCharacterConfidence

C/C++ Declaration

```
__stdcall void ICR_GetCharacterConfidence(long
Session, long CharIndex, long CharAlt, long*
Buffer);
```

C# Declaration

```
void ICR_GetCharacterConfidence(int Session, int
CharIndex, int CharAlt, ref int Buffer);
```

Visual Basic Declaration

```
Sub ICR_GetCharacterConfidence(ByVal Session As
Integer, ByVal CharIndex AS Integer, ByVal CharAlt
As Integer, ByRef Buffer As Integer)
```

Visual Basic .NET Declaration

```
Sub ICR_GetCharacterConfidence(ByVal Session As
Integer, ByVal CharIndex As Integer, ByVal CharAlt
As Integer, ByRef Buffer As Integer)
```

Delphi Declaration

```
procedure
ICR_GetCharacterConfidence(SessionHandle:Integer;
CharIndex:Integer; CharAlternative:Integer; Var
Confidence:Integer); stdcall;
```

Java Declaration

```
void ICR_GetCharacterConfidence (int Session, int
CharIndex, int CharAlternative, int Confidence);
```

Description

Retrieves the confidence of one of possible character alternative recognized. For each character you can get 3 alternatives.

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).
Confidence (out) - the buffer where will be movede the confidence of characters. The value is in the range 0 (low) to 100 (hi).

Return values

n/a

3.12 ICR_GetCharacterValue

C/C++ Declaration

```
__stdcall void ICR_GetCharacterValue(long Session,
long CharIndex, long CharAlt, char* Buffer);
```

C# Declaration

```
void ICR_GetCharacterValue(int Session, int
CharIndex, int CharAlt, string Buffer);
```

Visual Basic Declaration

```
Sub ICR_GetCharacterValue(ByVal Session As
Integer, ByVal CharIndex As Integer, ByVal CharAlt
As Integer, ByVal Buffer As String)
```

Visual Basic .NET Declaration

```
Sub ICR_GetCharacterValue(ByVal Session As
```

```
Integer, ByVal CharIndex As Integer, ByVal CharAlt
As Integer, ByVal Buffer As StringBuilder)
```

Delphi Declaration

```
procedure
ICR_GetCharacterValue(SessionHandle:Integer;
CharIndex:Integer; CharAlternative:Integer; Var
Value:Char); stdcall;
```

Java Declaration

```
void ICR_GetCharacterValue (int Session, int
CharIndex, int CharAlternative, String Value);
```

Description

Retrieves the ascii value of one of possible character alternative recognized. For each character you can get 3 alternatives.

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).
Value (out) - the buffer where will be movede the ascii value of character.

Return values

n/a

3.13 ICR_GetCharacterRect

C/C++ Declaration

```
__stdcall void ICR_GetCharacterRect(long Session,
long CharIndex, long* Left, long* Top, long*
Right, long* Bottom);
```

C# Declaration

```
void ICR_GetCharacterRect(int Session, int
CharIndex, ref int Left, ref int Top, ref int
Right, ref int Bottom);
```

Visual Basic Declaration

```
Sub ICR_GetCharacterRect(ByVal Session As Integer,
ByVal CharIndex As Integer, ByRef Left As Integer,
ByRef Top As Integer, ByRef Right As Integer,
ByRef Bottom As Integer)
```

Visual Basic .NET Declaration

```
Sub ICR_GetCharacterRect(ByVal Session As Integer,
ByVal CharIndex As Integer, ByRef Left As Integer,
ByRef Top As Integer, ByRef Right As Integer,
ByRef Bottom As Integer)
```

Delphi Declaration

```
procedure
ICR_GetCharacterRect(SessionHandle:Integer;
CharIndex:Integer; Var
Left,Top,Right,Bottom:Integer); stdcall;
```

Java Declaration

```
void ICR_GetCharacterRect (int Session, int
CharIndex,int Left,int Top,int Right,int Bottom);
```

Description

Retrieves the rectangle coordinates of one of possible character alternative recognized. For each character you can get 3 alternatives.

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

3.14 ICR_GetCharactersCount

C/C++ Declaration

```
__stdcall long ICR_GetCharactersCount(long
Session);
```

C# Declaration

```
int ICR_GetCharactersCount(int Session);
```

Visual Basic Declaration

```
Function ICR_GetCharactersCount(ByVal Session As
Integer) As Integer
```

Visual Basic .NET Declaration

```
Function ICR_GetCharactersCount(ByVal Session As
Integer) As Integer
```

Delphi Declaration

```
function
ICR_GetCharactersCount(SessionHandle:Integer):Inte
ger; stdcall;
```

Java Declaration

```
int ICR_GetCharactersCount (int Session);
```

Description

This is the function to call after the recognition process to get the number of characters recognized in the image, spaces excluded.

Parameters

SessionHandle (in) - the session handle to use

Return values

The numbers of recognized characters, spaces excluded.

3.15 ICR_FreeImage

C/C++ Declaration

```
__stdcall void ICR_FreeImage(long Session,long
DIBHandle);
```

C# Declaration

```
void ICR_FreeImage(int Session, int DIBHandle);
```

Visual Basic Declaration

```
Sub ICR_FreeImage(ByVal Session As Integer, ByVal
DIBHandle As Integer)
```

Visual Basic .NET Declaration

```
Sub ICR_FreeImage(ByVal Session As Integer, ByVal
DIBHandle As Integer)
```

Delphi Declaration

```
procedure ICR_FreeImage(Session:Integer;
DIBHandle:Integer);  stdcall;
```

Java Declaration

```
void ICR_FreeImage(int Session, int DIBHandle);
```

Description

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

Parameters

Session (in) - the session handle to use
DIBHandle (in) - the handle of the DIB to free

Return values

n/a

3.16 ICR_LoadImage

C/C++ Declaration

```
__stdcall long ICR_LoadImage(long Session, char*
FileName);
```

C# Declaration

```
int ICR_LoadImage(int Session, string FileName);
```

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

```
function ICR_LoadImage(Session:Integer;
FileName:PAnsiChar):Integer;  stdcall;
```

Java Declaration

```
int ICR_LoadImage(int Session, 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.

3.17 LoadICRLibrary

C/C++ Declaration

```
long LoadICRLibrary();
```

Description

Load the ICR DLL library: you have to use this function one time before to use other API functions

3.18 FreeICRLibrary

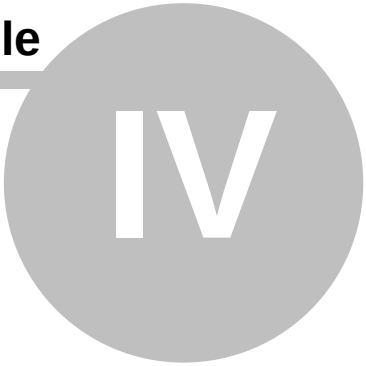
C/C++ Declaration

```
void FreeICRLibrary();
```

Description

Unload the ICR DLL library: you have to use this function one time before to exit from your application.

Sample



4 Sample

4.1 Code Sample

```
#include "stdafx.h"
#include "recoicr.c"

int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE
hPrevInstance,LPSTR lpCmdLine, int nCmdShow)
{

        // Load dynamically the library
        LoadICRLibrary();

        // Init the ICR session
        int Session= ICR_Init("demo", "demo");

        // Select the recognizer
        ICR_SelectRecognizer(Session, ".\\digit.rec");

        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 a line of text from the DIB
                int nChar= ICR_RecognizeLine(Session, (long)
hBitmap, 0);

                // Close the Clipboard
                ::CloseClipboard();

                // Check if there are recognized chars
                if (nChar>0) {

                        // Retrieve the recognized characters
                        nChar = ICR_GetResultValue(Session,
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 data recognized !", "ERROR",
MB_OK);


        }
        // Show an error message
        else MessageBox(NULL, "Unable to paste DIB", "ERROR",
MB_OK);


        // Close the session
        ICR_Done(Session);

        // Unload the library
        FreeICRLibrary();

        return 0;
}
```

Annotation