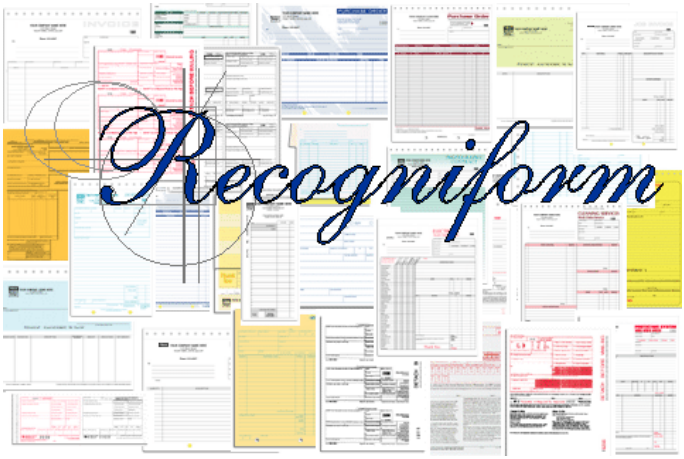


Software Development Kit

OMR

Recogniform Technologies SpA

HOW TO CONTACT US

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

Phone : +39 0984 404174

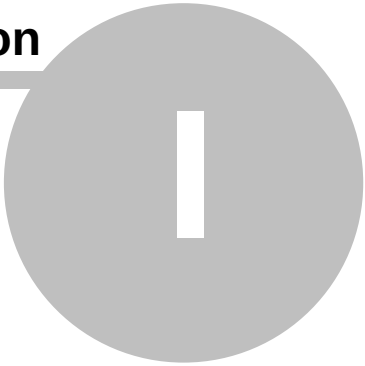
Fax : +39 0984 830299

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

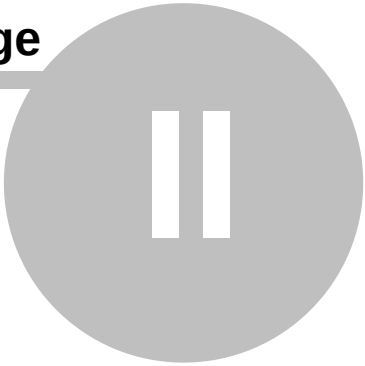
1.2 License

1.3 Overview

The library allows to recognize check-boxes from images acquired using a scanner. Its supported the recognition by pixel density and check extension.

The recognition process is fast and accurate and you can get the percentage of ink in the box as well as the check size.

Usage



2 Usage

2.1 Visual C++

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

2.2 C#

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

2.3 Visual Basic

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

2.4 Visual Basic .NET

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

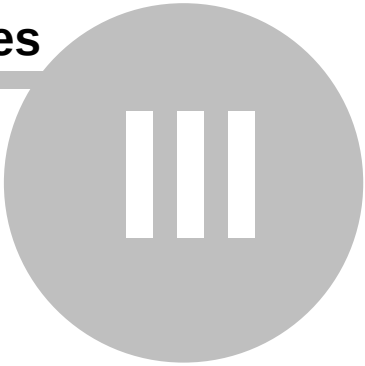
2.5 Delphi

You have to include the RECOOMRAPI.PAS in your program. Before to execute your application make sure the *RECOOMR.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 RECOOMRAPI.java in your program. Before to execute your application make sure the *RECOOMR.DLL* is available in your same .jar directory.

API References



3 API References

3.1 OMR_Init

C/C++ Declaration

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

C# Declaration

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

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

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

Java Declaration

```
int OMR_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 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
Passwors (in) - then password string

Return values

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

3.2 OMR_Done

C/C++ Declaration

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

C# Declaration

```
void OMR_Done(int Session);
```

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

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

Java Declaration

```
void OMR_Done(int SessionHandle);
```

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 OMR_Recognize

C/C++ Declaration

```
__stdcall long OMR_Recognize(long Session, long
DIBIn, long Left, long Top, long Right, long
Bottom);
```

C# Declaration

```
int OMR_Recognize(int Session, int DIBIn, int
Left, int Top, int Right, int Bottom );
```

Visual Basic Declaration

```
Function OMR_Recognize(ByVal Session As Integer,
ByVal DIBIn As Integer, ByVal Left As Integer,
ByVal Top As Integer, ByVal Right As Integer,
ByVal Bottom As Integer) As Integer
```

Visual Basic .NET Declaration

```
Function OMR_Recognize(ByVal Session As Integer,
ByVal DIBIn As Integer, ByVal Left As Integer,
ByVal Top As Integer, ByVal Right As Integer,
ByVal Bottom As Integer) As Integer
```

Delphi Declaration

```
function OMR_Recognize(SessionHandle, DIBHandle,
Left, Top, Right, Bottom:Integer):Integer;
stdcall;
```

Java Declaration

```
int OMR_Recognize(int SessionHandle, int
DIBHandle, int Left, int Top, int Right, int
Bottom);
```

Description

This is the function performing the recognition on a check box. you have to call this function after library initialization to perform the optical mark recognition. The DIB handle can contain the image with the check box to recognize.

Parameters

SessionHandle (in) - the session handle to use
DIBIn (in) - the handle of the DIB to recognize
Left (in) - the left coordinate of the check box to recognize
Top(in) - the top coordinate of the check box to recognize
Right (in) - the right coordinate of the check box to recognize
Bottom (in) - the bottom coordinate of the check box to recognize

Return values

0 if success, -1 if the session is invalid, -2 if the DIB handle is invalid, -3 if some internal error occurred.

Notes

The library works with standard DIBs images. Refer to Microsoft documentation for additional info about Device Independent Bitmaps. You have to use bitonal images. If you need to recognize multiple check boxes, you can call several time this function on the same images, changing just the box coordinates.

3.4 OMR_RecognizeSignature

C/C++ Declaration

```
__stdcall long OMR_RECOGNIZESIGNATURE(long Session
,long DIBHandle, long* SignatureLeft, long*
SignatureTop, long* SignatureRight, long*
SignatureBottom, long* SignatureStrokeLengthPixel,
long* SignatureStrokeAreaPixel, long*
SignaturePrePrintedCharactersCount, long*
SignaturePrePrintedLinePixel, long*
SingatureNoisePixel);
```

C# Declaration

```
int OMR_RecognizeSignature(int SessionHandle, int
DIBHandle, ref int SignatureLeft, ref int
SignatureTop, ref int SignatureRight, ref int
SignatureBottom, ref int
SignatureStrokeLengthPixel, ref int
SignatureStrokeAreaPixel,ref  int
SignaturePrePrintedCharactersCount,ref  int
SignaturePrePrintedLinePixel,ref int
SingatureNoisePixel);
```

Visual Basic Declaration

```
Function OMR_RecognizeSignature (ByVal
SessionHandle As Long, ByVal DIBHandle As Long,
ByRef SignatureLeft As Long, ByRef SignatureTop As
Long, ByRef signatureRight As Long, ByRef
SignatureBottom As Long, ByRef
SignatureStrokeLengthPixel As Long, ByRef
SignatureStrokeAreaPixel As Long, ByRef
SignaturePrePrintedCharactersCount As Long, ByRef
SignaturePrePrintedLinePixel As Long, ByRef
SingatureNoisePixel As Long) As Long
```

Visual Basic .NET Declaration

```
Function OMR_RecognizeSignature(ByVal
SessionHandle As Integer, ByVal DIBHandle As
Integer, ByRef SignatureLeft As Integer, ByRef
SignatureTop As Integer, ByRef signatureRight As
Integer, ByRef SignatureBottom As Integer, ByRef
SignatureStrokeLengthPixel As Integer, ByRef
SignatureStrokeAreaPixel As Integer, ByRef
SignaturePrePrintedCharactersCount As Integer,
ByRef SignaturePrePrintedLinePixel As Integer,
ByRef SingatureNoisePixel As Integer) As Integer
```

Delphi Declaration

```
function
OMR_RecognizeSignature(SessionHandle:Integer;DIBHa
ndle:Integer;SignatureLeft, SignatureTop,
SignatureRight, SignatureBottom,
SignatureStrokeLengthPixel,
SignatureStrokeAreaPixel,
SignaturePrePrintedCharactersCount,
SignaturePrePrintedLinePixel ,
SingatureNoisePixel:PInteger):Integer; stdcall;
external 'RECOOMR.DLL';
```

Java Declaration

```
int OMR_RecognizeSignature(int SessionHandle, int
DIBHandle, Memory SignatureLeft, Memory
SignatureTop, Memory SignatureRight, Memory
SignatureBottom, Memory
SignatureStrokeLengthPixel, Memory
SignatureStrokeAreaPixel, Memory
SignaturePrePrintedCharactersCount, Memory
```

```
SignaturePrePrintedLinePixel , Memory
SingatureNoisePixel);
```

Description

This function allows to recognize if there is a signature on a part of document

Parameters

SessionHandle (in) - the session handle to use
DIBHandle (in) –bitmap handle
SignatureLeft (out) - left coordinate of signature position
SignatureTop (out) - top coordinate of signature position
SignatureRight (out) - right coordinate of signature position
SignatureBottom (out) - bottom coordinate of signature position
SignatureStrokeLengthPixel (out) - stroke length in pixel
SignatureStrokeAreaPixel (out) - signature pixel
SignaturePrePrintedCharactersCount (out) - number of located printed character
SignaturePrePrintedLinePixel (out) - number of pixel of recognized horizontal line
SingatureNoisePixel: (out) noise in pixel

Return values

It return 0 if success, -1 if there is an error

3.5 OMR_GetMarkDensityPercent

C/C++ Declaration

```
__stdcall long OMR_GetMarkDensityPercent(long
Session, double* Value);
```

C# Declaration

```
int OMR_GetMarkDensityPercent(int Session, ref
double Value);
```

Visual Basic Declaration

```
Function OMR_GetMarkDensityPercent(ByVal Session
As Integer, ByRef Value As Double) As Integer
```

Visual Basic .NET Declaration

```
Function OMR_GetMarkDensityPercent(ByVal Session
As Integer, ByRef Value As Double) As Integer
```

Delphi Declaration

```
function
OMR_GetMarkDensityPercent(SessionHandle:Integer;
Var Value:Double):Integer; stdcall;
```

Java Declaration

```
int OMR_GetMarkDensityPercent(int SessionHandle,
Memory Value);
```

Description

This is the function to get the percentage of ink found in the last recognition. You can compare the returned value with a threshold to decide if a box is filled or not.

Parameters

SessionHandle (in) - the session handle to use
Value (out) – the percentage of black pixels recognized

Return values

0 if success, -1 if the session is invalid, -3 if some internal error occurred

3.6 OMR_GetMarkSizePercent

C/C++ Declaration

```
__stdcall long OMR_GetMarkSizePercent(long
Session, double* Value);
```

C# Declaration

```
internal static extern int
OMR_GetMarkSizePercent(int Session, ref double
Value );
```

Visual Basic Declaration

```
Function OMR_GetMarkSizePercent(ByVal Session As
```

```
Integer, ByRef Value As Double) As Integer
```

Visual Basic .NET Declaration

```
Function OMR_GetMarkSizePercent(ByVal Session As
Integer, ByRef Value As Double) As Integer
```

Delphi Declaration

```
function
OMR_GetMarkSizePercent(SessionHandle:Integer; Var
Value:Double):Integer; stdcall;
```

Java Declaration

```
int OMR_GetMarkSizePercent(int SessionHandle,
Memory Value);
```

Description

This is the function to get the size of mark in percentage respect to the box diagonal size.You can compare the returned value with a threshold to decide if a box is checked or not.

Parameters

SessionHandle (in) - the session handle to use
Value (out) – the size of mark in percentage respect to the box diagonal

Return values

0 if success, -1 if the session is invalid, -3 if some internal error occurred

3.7 OMR_ImportDDB

C/C++ Declaration

```
__stdcall long OMR_ImportDDB(long Session, long
BitmapHandle, long PaletteHandle);
```

C# Declaration

```
int OMR_ImportDDB(int Session, int BitmapHandle,
int PaletteHandle);
```

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

```
function OMR_ImportDDB(SessionHandle:Integer;
BitmapHandle:Integer;
PaletteHandle:Integer):Integer;  stdcall;
```

Java Declaration

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

Description

This is the function to import a Device Dependent Bitmap building an Device Independent Bitmap handle.

Parameters

SessionHandle (in) - the session handle to use
BitmapHandle (out) –bitmap handle
PaletteHandle (in)- Palette handle

Return values

Bitmap handle

3.8 OMR_LoadImage

C/C++ Declaration

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

C# Declaration

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

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

```
function OMR_LoadImage(SessionHandle:Integer;
FileName:PAnsiChar):Integer;  stdcall;
```

Java Declaration

```
int OMR_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.

3.9 OMR_FreeImage

C/C++ Declaration

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

C# Declaration

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

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

```
procedure OMR_FreeImage(SessionHandle:Integer;
DIBHandle:Integer);  stdcall;
```

Java Declaration

```
OMR_FreeImage(int SessionHandle,int DIBHandle);
```

Description

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

Parameters

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

Return values

n/a

3.10 LoadOMRLibrary

C/C++ Declaration

```
long LoadOMRLibrary();
```

Description

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

3.11 FreeOMRLibrary

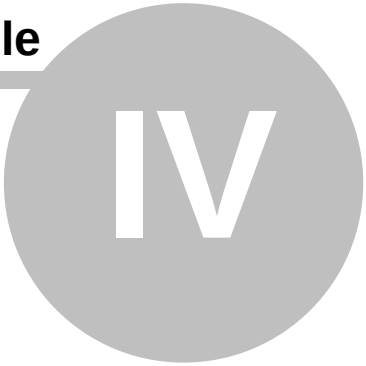
C/C++ Declaration

```
void FreeOMRLibrary()
```

Description

Unload the OMR 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 "recoomr.c"
int APIENTRY WinMain(HINSTANCE hInstance, HINSTANCE
hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{
 // Load dynamically the library
 LoadOMRLibrary();
 // Init the OMR session
 int Session= OMR_Init("demo", "demo");
 long hBitmap;
 // 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 check box in fixed position
  OMR_Recognize (Session, (long) hBitmap, 20, 50, 40, 70);
  // Close the Clipboard
  ::CloseClipboard();
  // Retrieve the ink percentage (0 – 100)
  double dInk;
  OMR_GetMarkDensityPercent (Session, &dInk);
  // Retrieve the check percentage (0 – 100)
  double dCheck;
  OMR_GetMarkSizePercent (Session, &dCheck);
  double dInkThreshold = 10F;
  double dCheckThreshold = 60F;
  // Compare the percentages with the thresholds
  if ((dInk>=dInkThreshold) && (dCheck>=dCheckThreshold))
  {
   // The check box is checked
   MessageBox(NULL, "Checked ! ", "RESULT", MB_OK);
  } else
    {
     if ((dInk<dInkThreshold) && (dCheck<dCheckThreshold))
     {
       // The check box is unchecked
       MessageBox(NULL, "Unchecked ! ", "RESULT", MB_OK);
     } else
         {
          // The check box status is unknown
          MessageBox(NULL, "Unknown! ", "RESULT", MB_OK);
         }
     }
  }
  // Show an error message
  else MessageBox(NULL, "Unable to paste DIB", "ERROR",
MB_OK);
  // Close the session
  OMR_Done(Session);
  // Unload the library
```

```
  FreeOMRLibrary();
  return 0;
}
```

Annotation