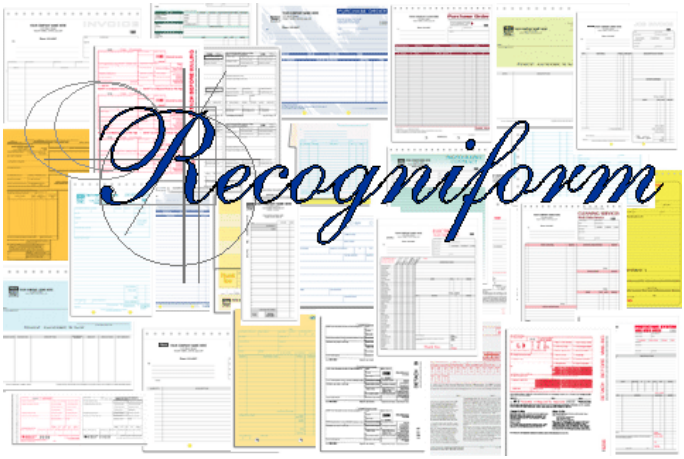


Software Development Kit

OCRA

Copyright 2020

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 5

- Copyright 5
- License 5
- Overview 5

Usage 8

- Visual C++ 8
- C# 8
- Visual Basic 8
- Visual Basic .NET 8
- Java 8
- Delphi 8

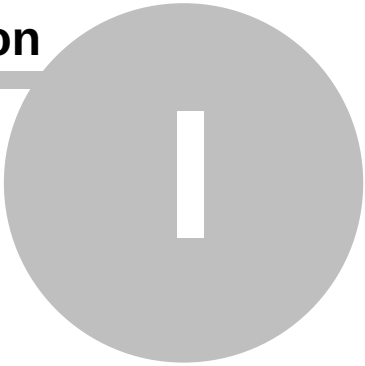
API References 10

- OCRA_Init 10
- OCRA_Done 11
- OCRA_Read 12
- OCRA_ResultLinesCount 13
- OCRA_ResultLineLength 14
- OCRA_ResultLineChars 16
- OCRA_ResultLineConfidences 17
- OCRA_ResultLineRects 18
- OCRA_SetParameter 19

Sample 23

- Code Sample 23

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

It is illegal to copy or reproduce this manual, or any part thereof, in any shape or form. The information contained in this manual is subject to change without notice and does not present a commitment on the part of Recogniform Technologies SpA.

Recogniform Technologies SpA shall not be held liable for technical or editorial errors and/or omissions made here, nor for incidental or consequential damages resulting from the furnishing, performance, or use of the software and documentation.

Recogniform Technologies SpA reserves the right to make changes to the software and documentation without notice.

Product names mentioned here are used for identification purposes only and may be tradenames and/or registered trademarks of their respective companies.

YOU CANNOT DISTRUBUTE SOFTWARE INCLUDING THIS SDK LIBRARY UNLESS YOU HAVE A WRITTEN AGREEMENT (ROYALTIES FREE OR ROYALTIES BASED) WITH RECOGNIFORM TECHNOLOGIES SPA

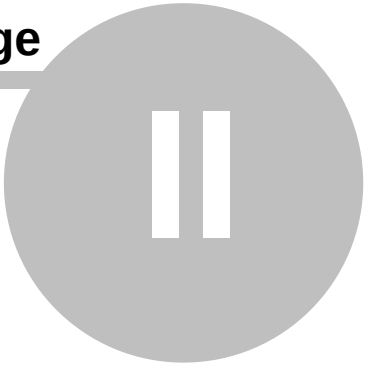
1.3 Overview

The library allows to recognize OCR-A codelines from images acquired from scanner. The basic character set supported contains numeric and symbols characters:

0 1 2 3 4 5 6 7 8 9 > < # + -

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

2.2 C#

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

2.3 Visual Basic

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

2.4 Visual Basic .NET

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

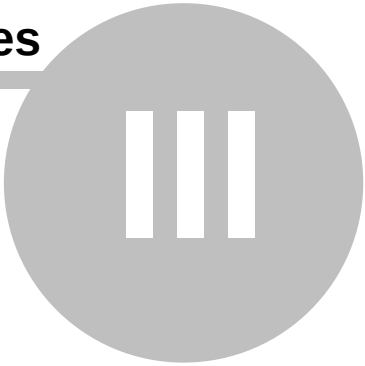
2.5 Java

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

2.6 Delphi

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

API References



3 API References

3.1 OCRA_Init

C/C++ Declaration

```
__stdcall long OCRA_Init (char* Name, char* Key,
long* SessionHandle);
```

C# Declaration

```
int OCRA_Init(string Name, string Key, ref int
SessionHandle);
```

Visual Basic Declaration

```
Function OCRA_Init (ByVal Name As String, ByVal
Key As String, ByRef SessionHandle As Long) As
Long
```

Visual Basic .NET Declaration

```
Function OCRA_Init(ByVal Name As String, ByVal Key
As String, ByRef SessionHandle As Integer) As
Integer
```

Delphi Declaration

```
function
OCRA_Init(User:PAnsiChar;Password:PAnsiChar;Var
SessionHandle:THandle):Integer; stdcall;
```

Java Declaration

```
int OCRA_Init(String User, String Password,int[]
Session);
```

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
Passwords (in) - then password string
SessionHandle (out) - the session handle to use in next library calls

Return values

OCRA_OK : success, the library is initialized
OCRA_NOMEM : the system don't have enough memory
OCRA_NORESOURCE : some necessary resources don't are available

Note

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 OCRA_Done

C/C++ Declaration

```
__stdcall long OCRA_Done(long Session);
```

C# Declaration

```
int OCRA_Done(int Session);
```

Visual Basic Declaration

```
Function OCRA_Done(ByVal Session As Long) As Long
```

Visual Basic .NET Declaration

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

Delphi Declaration

```
function OCRA_Done(SessionHandle:THandle):Integer;
stdcall;
```

Java Declaration

```
int OCRA_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

OCRA_OK : success, the library is finalized
OCRA_INVALIDHANDLE : the session handle is not valid

3.3 OCRA_Read

C/C++ Declaration

```
__stdcall long OCRA_Read(long Session, long DIB,
byte SingleLine);
```

C# Declaration

```
int OCRA_Read(int Session,int DIB, byte
SingleLine);
```

Visual Basic Declaration

```
Function OCRA_Read (ByVal Session As Long, ByVal
DIB As Long, ByVal SingleLine As Byte) As Long
```

Visual Basic .NET Declaration

```
Function OCRA_Read(ByVal Session As Integer, ByVal
DIB As Integer, ByVal SingleLine As Byte) As
Integer
```

Delphi Declaration

```
function
OCRA_Read(SessionHandle:THandle;DIB:THandle;Single
Line:Boolean):Integer; stdcall;
```

Java Declaration

```
int OCRA_Read(int Session, int DIB, int
SingleLine);
```

Description

This function performs the recognition: you have to call this function after library initialization.

Parameters

SessionHandle (in) - the session handle to use
DIBIn (in) - the handle of the monochrome DIB to recognize
SingleLine (in) - flag to help the recognition: if "true" (1) the engine search for a single codeline, if "false" (0) the engine search for multiple codelines.

Return values

OCRA_OK : success, the recognition is done correctly
OCRA_INVALIDHANDLE : the session handle is not valid
OCRA_INVALIDDIB : the input DIB is not valid, ie. is not a monochrome image

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.

3.4 OCRA_ResultLinesCount

C/C++ Declaration

```
__stdcall long OCRA_ResultLinesCount(long Session,
long* Count);
```

C# Declaration

```
int OCRA_ResultLinesCount(int Session, ref int
Count);
```

Visual Basic Declaration

```
Function OCRA_ResultLinesCount (ByVal Session As
Long, ByRef Count As Long) As Long
```

Visual Basic .NET Declaration

```
Function OCRA_ResultLinesCount(ByVal Session As
Integer, ByRef Count As Integer) As Integer
```

Delphi Declaration

```
function
OCRA_ResultLinesCount(SessionHandle:THandle; Var
Count:Integer):Integer; stdcall;
```

Java Declaration

```
int OCRA_ResultLinesCount(int SessionHandle, int[]
Count);
```

Description

This is the function to call after the recognition process to get the number of codelines found in the image.

Parameters

SessionHandle (in) - the session handle to use
LinesCount (out) - the number of codelines recognized in the image

Return values

OCRA_OK : success, the lines count is retrieved
OCRA_INVALIDHANDLE : the session handle is not valid

3.5 OCRA_ResultLineLength

C/C++ Declaration

```
__stdcall long OCRA_ResultLineLength(long Session,
long LineIndex,long* Count);
```

C# Declaration

```
int OCRA_ResultLineLength(int Session, int
LineIndex, ref int Count);
```

Visual Basic Declaration

```
Function OCRA_ResultLineLength (ByVal Session As
Long, ByVal LineIndex As Long, ByRef Count As
Long) As Long
```

Visual Basic .NET Declaration

```
Function OCRA_ResultLineLength(ByVal Session As
Integer, ByVal LineIndex As Integer, ByRef Count
As Integer) As Integer
```

Delphi Declaration

```
function
OCRA_ResultLineLength(SessionHandle:THandle;
Line:Integer; Var Length:Integer):Integer;
stdcall;
```

Java Declaration

```
int OCRA_ResultLineLength(int Session, int Line,
int [] Length);
```

Description

This is the function to call after the recognition process to get the length of a recognized codeline.

Parameters

SessionHandle (in) - the session handle to use
LineIndex (ind) - the number of the codeline to retrieve the length. First codeline is 0.
LineLength (out) - the number of character recognized in the selected codeline

Return values

OCRA_OK : success, the line length is retrieved
OCRA_INVALIDHANDLE : the session handle is not valid

OCRA_INVALIDLINE : the line index is not valid

3.6 OCRA_ResultLineChars

C/C++ Declaration

```
__stdcall long OCRA_ResultLineChars(long Session,
long LineIndex, char* Buffer);
```

C# Declaration

```
int OCRA_ResultLineChars(int Session, int
LineIndex, StringBuilder Buffer);
```

Visual Basic Declaration

```
Function OCRA_ResultLineChars (ByVal Session As
Long, ByVal LineIndex As Long, ByVal Buffer As
String) As Long
```

Visual Basic .NET Declaration

```
Function OCRA_ResultLineChars(ByVal Session As
Integer, ByVal LineIndex As Integer, ByVal Buffer
As StringBuilder) As Integer
```

Delphi Declaration

```
function
OCRA_ResultLineChars(SessionHandle:THandle;
Line:Integer; Buffer:PAnsiChar):Integer; stdcall;
```

Java Declaration

```
int OCRA_ResultLineChars(int Session, int Line,
Memory Buffer);
```

Description

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

Parameters

SessionHandle (in) - the session handle to use
LineIndex (ind) - the number of the codeline to retrieve the recognized chars. First codeline is 0.

CharBuffer (out) - the buffer where will be movede the characters recognized in the selected codeline. You have to allocate enough space.

Return values

OCRA_OK : success, the line length is retrieved
OCRA_INVALIDHANDLE : the session handle is not valid
OCRA_INVALIDLINE : the line index is not valid

3.7 OCRA_ResultLineConfidences

C/C++ Declaration

```
__stdcall long OCRA_ResultLineConfidences(long
Session, long LineIndex,float* Buffer);
```

C# Declaration

```
int OCRA_ResultLineConfidences(int Session, int
LineIndex, ref float Buffer);
```

Visual Basic Declaration

```
Function OCRA_ResultLineConfidences(ByVal Session
As Long, ByVal LineIndex As Long, ByRef Buffer As
Single) As Long
```

Visual Basic .NET Declaration

```
Function OCRA_ResultLineConfidences(ByVal Session
As Integer, ByVal LineIndex As Integer, ByRef
Buffer As Single) As Integer
```

Delphi Declaration

```
function
OCRA_ResultLineConfidences(SessionHandle:THandle;
Line:Integer; Buffer:Pointer):Integer; stdcall;
```

Java Declaration

```
int OCRA_ResultLineConfidences(int Session, int
Line, float[] Confidences);
```

Description

This is the function to call after the recognition process to get the confidences of recognized chars in a codeline. The confidence is the probability that a char was been recognized correctly.

Parameters

SessionHandle (in) - the session handle to use
LineIndex (ind) - the number of the codeline to retrieve the recognized chars. First codeline is 0.
ConfidenceBuffer (out) - the buffer where will be movede the confidences of characters recognized in the selected codeline. You have to allocate enough space. Each confidence value is rapresented by a single precision number in the range 0.0 - 1.0 where 0.0 is lowest confidence and 1.0 is the highest confidence level.

Return values

OCRA_OK : success, the line length is retrieved
OCRA_INVALIDHANDLE : the session handle is not valid
OCRA_INVALIDLINE : the line index is not valid

3.8 OCRA_ResultLineRects

C/C++ Declaration

```
__stdcall long OCRA_ResultLineRects(long Session,
long LineIndex,rect* Buffer);
```

C# Declaration

```
int OCRA_ResultLineRects(int Session, int
LineIndex, ref Rect Buffer);
```

Visual Basic Declaration

```
Function OCRA_ResultLineRects(ByVal Session As
Long, ByVal LineIndex As Long, ByRef Buffer As
Rect) As Long
```

Visual Basic .NET Declaration

```
Function OCRA_ResultLineRects(ByVal Session As
Integer, ByVal LineIndex As Integer, ByRef Buffer
As Rect) As Integer
```

Delphi Declaration

```
function OCRA_ResultLineRects
(SessionHandle:THandle; LineIndex:Integer; Var
Buffer:TRect): Integer; stdcall;
```

Java Declaration

```
int OCRA_ResultLineRects(int Session, int Line,
Memory Buffer);
```

Description

This is the function to call after the recognition process to get the rectagle areas filled by each character in a codeline.

Parameters

SessionHandle (in) - the session handle to use
LineIndex (ind) - the number of the codeline to retrieve the recognized chars. First codeline is 0.
RectsBuffer (out) - the buffer where will be movede the rects of characters recognized in the selected codeline. You have to allocate enough space. Each rect value is rapresented by four long numbers rapresenting left, top, right and bottom posizion of the rectangle.

Return values

OCRA_OK : success, the line length is retrieved
OCRA_INVALIDHANDLE : the session handle is not valid
OCRA_INVALIDLINE : the line index is not valid

3.9 OCRA_SetParameter

C/C++ Declaration

```
__stdcall long OCRA_SetParameter(long Session,
long ParameterID, long ParameterValue);
```

C# Declaration

```
int OCRA_SetParameter(int Session, int
ParameterID, int ParameterValue);
```

Visual Basic Declaration

```
Function OCRA_SetParameter (ByVal Session As Long,
ByVal ParameterID As Long, ByVal ParameterValue As
```

```
Long) As Long
```

Visual Basic .NET Declaration

```
Function OCRA_SetParameter(ByVal Session As
Integer, ByVal ParameterID As Integer, ByVal
ParameterValue As Integer) As Integer
```

Delphi Declaration

```
function OCRA_SetParameter(Session:Integer;
ParameterID :Integer; ParameterValue:Integer):
Integer; stdcall;
```

Java Declaration

```
int OCRA_SetParameter(int Session, int
ParameterID, int ParameterValue);
```

Description

This function allows to modify default parameters for recognition: you can use it for special requirements before perform the recognition, but be aware you can compromise the result with no correct settings.

Parameters

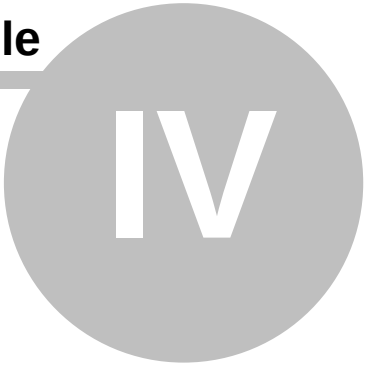
SessionHandle (in) - the session handle to use
ParameterId (in) - the parameter index to modify. Currently are supported this parameters:
OCRAPARAM_RECOGNIZESPACE : enable or disable the recognition of spaces. If Value is 1 the space recognition is enabled, if 0 is disabled. Default is 1.
OCRAPARAM_SPACEWIDTH : set the medium space width. Value is the space width expressed in pixels at 200 DPI. Default is 18.
OCRAPARAM_CHARMINW : set the minimum character width. Value is the width expressed in pixels at 200 DPI. Default is 6.
OCRAPARAM_CHARMINH : set the minimum character height. Value is the height expressed in pixels at 200 DPI. Default is 12.
OCRAPARAM_CHARMAXW : set the maximum character width. Value is the width expressed in pixels at 200 DPI. Default is 25.
OCRAPARAM_CHARMAXH : set the maximum character height. Value is the height expressed in pixels at 200 DPI. Default is 30.
OCRAPARAM_NOISESIZE : set the max noise size. Value is the

noise size expressed in pixels at 200 DPI. Default is 6.
OCRAPARAM_MINUSMINW : set the minimum "minus" width. Value is the width expressed in pixels at 200 DPI. Default is 10.
OCRAPARAM_MINUSMINH : set the minimum "minus" height. Value is the height expressed in pixels at 200 DPI. Default is 2.
OCRAPARAM_MINUSMAXW : set the maximum "minus" width. Value is the width expressed in pixels at 200 DPI. Default is 20.
OCRAPARAM_MINUSMAXH : set the maximum "minus" height. Value is the height expressed in pixels at 200 DPI. Default is 8.
OCRAPARAM_TECHNOLOGY : set the primary technology to use. Value can be 0 for neural network, 1 for statistical analysis. Default is 0.
OCRAPARAM_MAXRETRIES : set the maximum number of attempt to recognize a character: Value can range from 1 (faster) to 8 (slower). Default is 8.
ParameterValue (in) - the new parameter value. See parameter Id for admitted values.

Return values

OCRA_OK : success, the binarization is done correctly
OCRA_INVALIDHANDLE : the session handle is not valid
OCRA_INVALIDPARAMETER : the parameter index is not valid

Sample



4 Sample

4.1 Code Sample

```
// Demo user function
long RecognizeMyCodeline(long CodelineDIB)
 {

  // Declare local variables
  long Session,Error,Lines,LineLen;
  char Chars[1024]; // We expect to have codelines with less
then 1024 chars !
  single Confidences[1024];
  rect Rects[1024]

  // Init the library
  Error=OCRA_Init("demo","demo", &Session);

  // Check if any error occurred
  if (Error!=OCRA_OK)
   {

    // Perform the recognition
    Error=OCRA_Read(Session, CodelineDIB,  FALSE);

    // Check if any error occurred
    if (Error!=OCRA_OK)
     {

      // Retrieve the number of recognize lines
      OCRA_ResultLinesCount(Session, &Lines);

      // For each recognized text line...
      for (i=0; i<Lines; i++)
       {

        // Retrieve the codeline length
        OCRA_ResultLineLength(Session,i,&LineLen);

        //
==========================================================
=
        // You could allocate here memory for Chars,
Confidences and Rects
        // using LineLen value instead to use fixed size
buffers...
        //
==========================================================
=

        // Retrieve the codeline chars
        OCRA_ResultLineChars(Session,i,&Chars);

        // Retrieve the codeline char confidences...if
required
        OCRA_ResultLineConfidences(Session,i,&Confidences);

        // Retrieve the codeline char rects...if required
        OCRA_ResultLineRects(Session,i,&Rects);
```

```
            // ============================================
            // Add here your code to use the recognized data
            // ============================================


          }

        }

       // Deinitialize the library
       OCRA_Done(Session);
      }
     return (Error);
    }
```

Annotation