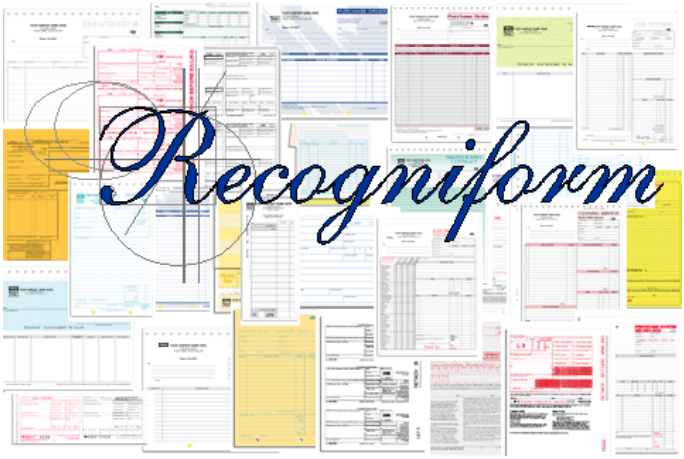


Software Development Kit

Form Identification

Copyright 2019

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 6

- Copyright ... 6
- License ... 6
- Overview ... 6

Usage 9

- Visual C++ ... 9
- C# ... 9
- Visual Basic ... 9
- Visual Basic .NET ... 9
- Java ... 9
- Delphi ... 9

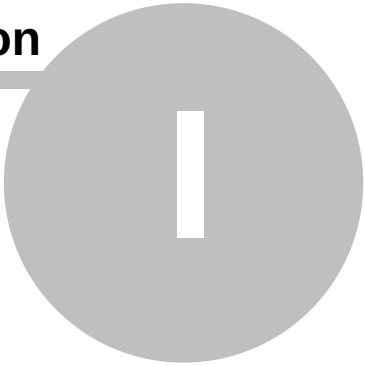
API References 11

- FORMID_Init ... 11
- FORMID_Done ... 12
- FORMID_Load ... 13
- FORMID_Save ... 14
- FORMID_Clear ... 15
- FORMID_AddFile ... 16
- FORMID_AddImage ... 17
- FORMID_Delete ... 18
- FORMID_Count ... 20
- FORMID_GetTemplateInfo ... 21
- FORMID_ProcessFile ... 22
- FORMID_ProcessImage ... 23
- FORMID_GetResult ... 25
- LoadFormIdLibrary ... 26
- FreeFormIdLibrary ... 26

Sample 29

- Code Sample ... 29

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.

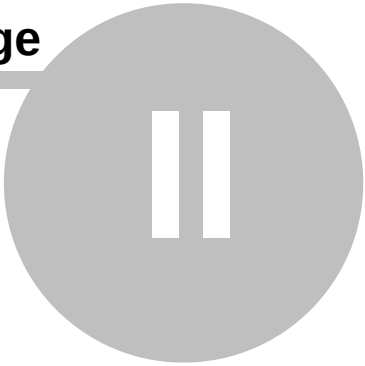
YOU CANNOT DISTRUBUTE SOFTWARE INCLUDING THIS SDK LIBRARY UNLESS YOU HAVE A WRITTEN AGREEMENT (ROYALTIES FREE OR ROYALTIES BASED) WITH RECOGNIFORM TECHNOLOGIES SPA

1.3 Overview

The library allows to identify the type of a form from a set of known forms types. After the identification, if this function was purchased and enabled, the library performs also the registration of the form on the matched template: this means that its calculated the horizontal and vertical displacement between the form and the matched

template as well as the skew, the resolution ratio and the orientation difference, if any. In this way its possible to identify with great accuracy the region of interest on the target form also if they was measured on the template form, compensating for the differences introduced by the scanning process.

Usage



2 Usage

2.1 Visual C++

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

2.2 C#

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

2.3 Visual Basic

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

2.4 Visual Basic .NET

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

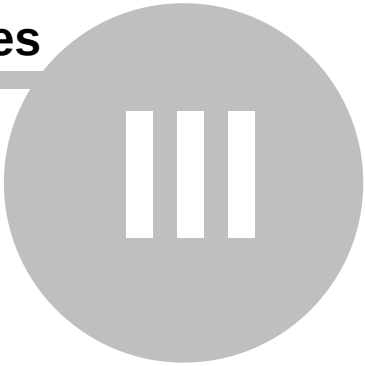
2.5 Java

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

2.6 Delphi

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

API References



3 API References

3.1 FORMID_Init

C/C++ Declaration

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

C# Declaration

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

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

```
function FORMID_Init(NAme:PAnsiChar;
LicenseKey:PAnsiChar):Integer; stdcall;
```

Java Declaration

```
int FORMID_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 production mode but, when you call the form identification 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.

3.2 FORMID_Done

C/C++ Declaration

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

C# Declaration

```
void FORMID_Done(int Session);
```

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

```
procedure FORMID_Done(Session: Integer); stdcall;
```

Java Declaration

```
void FORMID_Done(int Session);
```

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

3.3 FORMID_Load

C/C++ Declaration

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

C# Declaration

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

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

```
function
FORMID_Load(Session:Integer;FileName:PAnsiChar):In
teger; stdcall;
```

Java Declaration

```
int FORMID_Load(int Session, String FileName);
```

Description

This function allows to load a file, previously saved, with the data of templates forms set used for identification.

Parameters

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

Return values

The number of templates included in the set if success, -1 otherwise.

Notes

The file format is a binary proprietary and cannon be edited or created using other software: you can create it only using the FORMID_Save function of this SDK.

3.4 FORMID_Save

C/C++ Declaration

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

C# Declaration

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

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

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

Java Declaration

```
int FORMID_Save(int Session, String FileName);
```

Description

This function allows to save in a file the set of templates currently available in the session so that can be used in the future without to recreate the set.

Parameters

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

Return values

The number of templates included in the set if success, -1 otherwise.

Notes

You should save a set file only after the addition of one or more templates using the FORMID_Add function.

3.5 FORMID_Clear

C/C++ Declaration

```
__stdcall long FORMID_Clear(long Session);
```

C# Declaration

```
int FORMID_Clear(int Session);
```

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

```
function FORMID_Clear(Session:Integer):Integer;
stdcall;
```

Java Declaration

```
int FORMID_Clear(int Session);
```

Description

This function allows to delete the full set of templates currently available in the session.

Parameters

SessionHandle (in) - the session handle to use

Return values

0 if success, -1 otherwise.

Notes

None.

3.6 FORMID_AddFile

C/C++ Declaration

```
__stdcall void FORMID_AddFile(long Session, char*
FileName, long PageNumber, char* Description, long
Tag);
```

C# Declaration

```
int FORMID_AddFile(int Session, string FileName,
int PageNumber, string Description, int Tag);
```

Visual Basic Declaration

```
Function FORMID_AddFile(ByVal Session As Integer,
ByVal FileName As String, ByVal PageNumber As
Integer, ByVal Description As String, ByVal Tag As
Integer) As Integer
```

Visual Basic .NET Declaration

```
Function FORMID_AddFile(ByVal Session As Integer,
ByVal FileName As String, ByVal PageNumber As
Integer, ByVal Description As String, ByVal Tag As
Integer) As Integer
```

Delphi Declaration

```
function FORMID_AddFile(Session:Integer;
FileName:PAnsiChar; PageNumber: Integer;
Description:PAnsiChar; Tag: Integer):Integer;
stdcall;
```

Java Declaration

```
int FORMID_AddFile(int Session, String FileName,
int PageNumber,String  Description, int Tag);
```

Description

This function allows to add a new template form to the set available in the session.

Parameters

SessionHandle (in) - the session handle to use
FileName(in) - the name of the empty form file, in TIF format, to be used as template.
PageNumber (in) – the number of page to load from the file (0 = first page)
Description(in) - the description to be associated to the form template
Tag(in) - the numeric tag to be associated to the form template

Return values

The numeric index of the template into the set (first index is 0) or -1 if some error occurred.

Notes

None.

3.7 FORMID_AddImage

C/C++ Declaration

```
__stdcall void FORMID_AddImage(long Session, long
DIBHandle, char* Description, long Tag);
```

C# Declaration

```
int FORMID_AddImage(int Session, int DIBHandle,
string Description, int Tag);
```

Visual Basic Declaration

```
Function FORMID_AddImage(ByVal Session As Integer,
ByVal DIBHandle As Integer, ByVal Description As
String, ByVal Tag As Integer) As Integer
```

Visual Basic .NET Declaration

```
Function FORMID_AddImage(ByVal Session As Integer,
ByVal DIBHandle As Integer, ByVal Description As
String, ByVal Tag As Integer) As Integer
```

Delphi Declaration

```
function FORMID_AddImage(Session:Integer;
DIBHandle:Integer; Description: PAnsiChar; Tag:
Integer):Integer; stdcall;
```

Java Declaration

```
int FORMID_AddImage(int Session, int DIBHandle,
String Description, int Tag);
```

Description

This function allows to add a new template form to the set available in the session.

Parameters

SessionHandle (in) - the session handle to use
DIBHandle (in) - the DIB memory handle of image to be added as template.
Description(in) - the description to be associated to the form template
Tag(in) - the numeric tag to be associated to the form template

Return values

The numeric index of the template into the set (first index is 0) or -1 if some error occurred.

Notes

None.

3.8 FORMID_Delete

C/C++ Declaration

```
__stdcall long FORMID_Delete(long Session, long
Index);
```

C# Declaration

```
int FORMID_Delete(int Session , int Index);
```

Visual Basic Declaration

```
Function FORMID_Delete(ByVal Session As Integer,
ByVal Index As Integer) As Integer
```

Visual Basic .NET Declaration

```
Function FORMID_Delete(ByVal Session As Integer,
ByVal Index As Integer) As Integer
```

Delphi Declaration

```
function FORMID_Delete(Session:Integer;
Index:Integer):Integer; stdcall;
```

Java Declaration

```
int FORMID_Delete(int Session,int Index);
```

Description

This function allows to delete a template form from the set available in the session.

Parameters

SessionHandle (in) - the session handle to use
Index (in) – the template index to delete

Return values

The number of forms still available in the set or -1 if some error occurred.

Notes

n/a

3.9 FORMID_Count

C/C++ Declaration

```
__stdcall long FORMID_Count(long Session);
```

C# Declaration

```
int FORMID_Count(int Session);
```

Visual Basic Declaration

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

Visual Basic .NET Declaration

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

Delphi Declaration

```
function FORMID_Count(Session:Integer):Integer;
stdcall;
```

Java Declaration

```
int FORMID_Count(int Session);
```

Description

This function allows to retrieve the number of form templates included in the set available in the session.

Parameters

SessionHandle (in) - the session handle to use

Return values

The number of form templates in the set, -1 if some error occurred.

Notes

n/a

3.10 FORMID_GetTemplateInfo

C/C++ Declaration

```
__stdcall long FORMID_GetTemplateInfo(long
Session, long Index, char* Description, long*
Tag);
```

C# Declaration

```
int FORMID_GetTemplateInfo(int Session, int Index,
string Description, ref int Tag);
```

Visual Basic Declaration

```
Function FORMID_GetTemplateInfo(ByVal Session As
Integer, ByVal Index As Integer, ByVal Description
As String, ByRef Tag As Integer) As Integer
```

Visual Basic .NET Declaration

```
Function FORMID_GetTemplateInfo(ByVal Session As
Integer, ByVal Index As Integer, ByVal Description
As String, ByRef Tag As Integer) As Integer
```

Delphi Declaration

```
function FORMID_GetTemplateInfo(Session:Integer;
Index: Integer; Description: PAnsiChar; Tag:
Integer):Integer; stdcall;
```

Java Declaration

```
int FORMID_GetTemplateInfo(int Session, int
Index,Memory Description, int Tag);
```

Description

This function allows to retrieve the description and the tag associated to a template form.

Parameters

SessionHandle (in) - the session handle to use
Index (in) – the template index to use
Description (out) – the description associated to the template

Tag (out) – the numeric tag associated to the template

Return values

The length of description associated to the template, or -1 if some error occurred.

Notes

n/a

3.11 FORMID_ProcessFile

C/C++ Declaration

```
__stdcall long FORMID_ProcessFile(long Session,
char* FileName, long PageNumber, bool
FindOrientation, bool FindOffset, bool
FindStretch);
```

C# Declaration

```
int FORMID_ProcessFile(int Session, string
FileName, int PageNumber, bool FindOrientation,
bool FindOffset, bool FindStretch);
```

Visual Basic Declaration

```
Function FORMID_ProcessFile(ByVal Session As
Integer, ByVal FileName As String, ByVal
PageNumber As Integer, ByVal FindOrientation As
Boolean, ByVal FindOffset As Boolean, ByVal
FindStretch As Boolean) As Integer
```

Visual Basic .NET Declaration

```
Function FORMID_ProcessFile(ByVal Session As
Integer, ByVal FileName As String, ByVal
PageNumber As Integer, ByVal FindOrientation As
Boolean, ByVal FindOffset As Boolean, ByVal
FindStretch As Boolean) As Integer
```

Delphi Declaration

```
function FORMID_ProcessFile(Session:Integer;
FileName:PAnsiChar; PageNumber:Integer;
FindOrientation: Boolean; FindOffset:Boolean;
```

```
FindStretch:Boolean):Integer; stdcall;
```

Java Declaration

```
int FORMID_ProcessFile(int Session, String
FileName, int PageNumber,boolean
FindOrientation,boolean FindOffset, boolean
FindStretch);
```

Description

This function allows to process a form from a file.

Parameters

SessionHandle (in) - the session handle to use
FileName(in) - the name of TIFF file to process.
PageNumber (in) – the number of page to process from the file (0 = first page)
FindOrientation (in) - the flag allowing to enable or disable the matching with rotated forms.
FindOffset (in) - the flag allowing to enable or disable the estimation of horizontal and vertical offset.
FindStretch(in) - the flag allowing to enable or disable the estimation of horizontal and vertical stretch.

Return values

1 if the form was recognized, 0 if the form doesn't match with any template, -1 if some error occurred.

Notes

The resulting data can be retrieved using the function FORMID_GetResults.

3.12 FORMID_ProcessImage

C/C++ Declaration

```
__stdcall long FORMID_ProcessImage(long Session,
long DIBHandle, bool FindOrientation, bool
FindOffset, bool FindStretch);
```

C# Declaration

```
int FORMID_ProcessImage(int Session, int
DIBHandle, bool FindOrientation, bool FindOffset,
bool FindStretch);
```

Visual Basic Declaration

```
Function FORMID_ProcessImage(ByVal Session As
Integer, ByVal DIBHandle As Integer, ByVal
FindOrientation As Boolean, ByVal FindOffset As
Boolean, ByVal FindStretch As Boolean) As Integer
```

Visual Basic .NET Declaration

```
Function FORMID_ProcessImage(ByVal Session As
Integer, ByVal DIBHandle As Integer, ByVal
FindOrientation As Boolean, ByVal FindOffset As
Boolean, ByVal FindStretch As Boolean) As Integer
```

Delphi Declaration

```
function FORMID_ProcessImage(Session:Integer;
DIBHandle:Integer; FindOrientation: Boolean;
FindOffset:Boolean; FindStretch:Boolean): Integer;
stdcall;
```

Java Declaration

```
int FORMID_ProcessImage(int Session, int
DIBHandle, boolean FindOrientation,boolean
FindOffset, boolean FindStretch);
```

Description

This function allows to process an image from memory.

Parameters

SessionHandle (in) - the session handle to use
DIBHandle (in) – the memory handle of DIB to be processed
FindOrientation (in) - the flag allowing to enable or disable the matching with rotated forms.
FindOffset (in) - the flag allowing to enable or disable the estimation of horizontal and vertical offset.
FindStretch(in) - the flag allowing to enable or disable the estimation of horizontal and vertical stretch.

Return values

1 if the form was recognized, 0 if the form doesn't match with any template, -1 if some error occurred.

Notes

The resulting data can be retrieved using the function FORMID_GetResults.

3.13 FORMID_GetResult

C/C++ Declaration

```
__stdcall long FORMID_GetResult(long Session, long
ValueIndex, void* ValueBuffer);
```

C# Declaration

```
int FORMID_GetResult(int Session, int ValueIndex,
ref int ValueBuffer);
```

Visual Basic Declaration

```
Function FORMID_GetResult(ByVal Session As
Integer, ByVal ValueIndex As Integer, ByRef
ValueBuffer As Integer) As Integer
```

Visual Basic .NET Declaration

```
Function FORMID_GetResult(ByVal Session As
Integer, ByVal ValueIndex As Integer, ByRef
ValueBuffer As Integer) As Integer
```

Delphi Declaration

```
function FORMID_GetResult(Session:Integer;
ValueIndex:Integer; ValueBuffer: Integer):
Integer; stdcall;
```

Java Declaration

```
int FORMID_GetResult(int Session, int ValueIndex,
Memory ValueBuffer);
```

Description

This function allows to retrieve a value resulting from the last processing operation.

Parameters

SessionHandle (in) - the session handle to use
ValueIndex (in) – the value index to retrieve. Currently you can use 0 (Matching Template Index), 1 (Template Similarity Percentage x 100), 2 (Form Orientation Degrees), 3 (X Offset), 4 (Y Offset), 5 (X Stretch) 6 (Y Stretch), 7 (Skew Degrees x 100), 8 (Crop Border Left), 9 (Crop Border Top), 10 (Crop Border Right), 11 (Crop Border Bottom), 12 (X Resolution Percentage x 100), 13 (Y Resolution Percentage x 100)
ValueBuffer(out) – the numeric value retrieved

Return values

The length of value retrieved (4 if index is valid, else 0) if success, -1 if some error occurred.

Notes

n/a

3.14 LoadFormIdLibrary

C/C++ Declaration

```
long LoadFORMIDLibrary();
```

Description

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

3.15 FreeFormIdLibrary

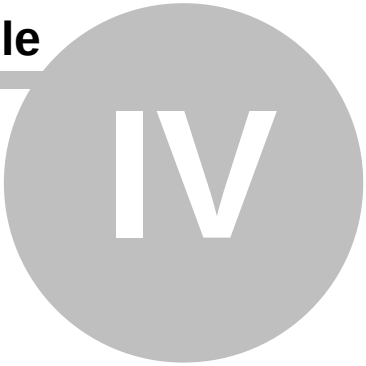
C/C++ Declaration

```
void FreeFORMIDLibrary();
```

Description

Unload the RECOFID.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 <stdio.h>
#include "recoformid.c"
int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE
hPrevInstance,LPSTR lpCmdLine,int nCmdShow)
{
 char buffer[200];
 // Load dynamically the library
 LoadFormIdLibrary();
 // Init the session
 int Session= FORMID_Init("demo", "demo");
 // You could also use FORMID_Load to load a saved form set
instead that add all the forms every time…
 // Add a first kind of form to learn by samples
 FORMID_AddFile(Session,".\\Templates\\Template_9452.tif", 0,”
Template_9452”,1);
 // Add a second kind of form to learn by samples
 FORMID_AddFile(Session,".\\Templates\\Template_SS4.tif", 0, ”
Template_SS4”,2);
 // Add a thirdy kind of form to learn by samples
 FORMID_AddFile(Session,".\\Templates\\Template_W7P.tif", 0, ”
Template_W7P”,3);
 // You could also use FORMID_Save to save the set and reuse
it next time…
 // Process a filled form
 if
(FORMID_ProcessFile(Session,".\\Filled\\Filled_SS4_1(Shifted).
tif ", 0, True, True, True)==1)
 {
  // Retrieve the form type and offsets
  int FormIndex, FormXOffset, FormYOffset;
  FORMID_GetResult (Session, 0, &FormIndex);
  FORMID_GetResult (Session, 3, &FormXOffset);
  FORMID_GetResult (Session, 4, &FormYOffset);
  // Retrieve the form name from id
  char TemplateName[200];
  int TemplateTag;
  FORMID_GetTemplateInfo (Session, FormIndex,
&TemplateName,&TemplateTag)
  MessageBox(NULL, TemplateName, "RESULT", MB_OK);
  sprintf( buffer, "X Offsets: %d – Y Offset: %d”,
FormXOffset,FormYOffset);
  MessageBox(NULL, buffer, "RESULT", MB_OK);
 }
  // Show an error message
 else MessageBox(NULL, "Unable to recognize the form !",
"ERROR",MB_OK);
 // Close the session
 FORMID_Done(Session);
 // Unload the library
 FreeFormIdLibrary();
 return 0;
}
```

Annotation