



# ***USER GUIDE***

(ver. EN-1.7)



© 2006-2008 ACTIVA di Gismondi Roberto - Italy. All rights reserved.  
[www.activasoft.it](http://www.activasoft.it)

FXInterpreter is a product of ACTIVA.

iFix is a trademark of GE Fanuc International Inc..  
Microsoft, MS, ActiveX, Visual Basic, Visual Basic for Applications,  
Windows, Windows XP are trademarks of Microsoft Corporation.  
Other company and product names mentioned in this document  
may be trademarks of their respective owners.

ACTIVA has no commercial or technical relationship with Microsoft and GE Fanuc.

## INDEX

SAVE YOUR TIME.....	3
INTRODUCTION.....	4
COMPARISON.....	5
SETUP.....	7
ADD TO THE PROJECT.....	7
TO BEGIN.....	8
TIPS.....	9
SINTAX.....	10
Translation of a picture.....	10
Change of language with translation of all visible pictures.....	11
Translation of a UserForm (VBA).....	11
Change of language with translation of all visible pictures and forms.....	12
Translation of MsgBox and InputBox.....	13
Translation of real-time alarms (AlarmSummary).....	13
Translation of historical alarms.....	13
Dictionary update.....	15
Dictionary global update and check for all pictures.....	15
Change of development language in pictures.....	16
Configuration file.....	16
LICENSE.....	16
WEB RESOURCES.....	17

**SAVE YOUR TIME**

**Save hours of work and cut costs !**

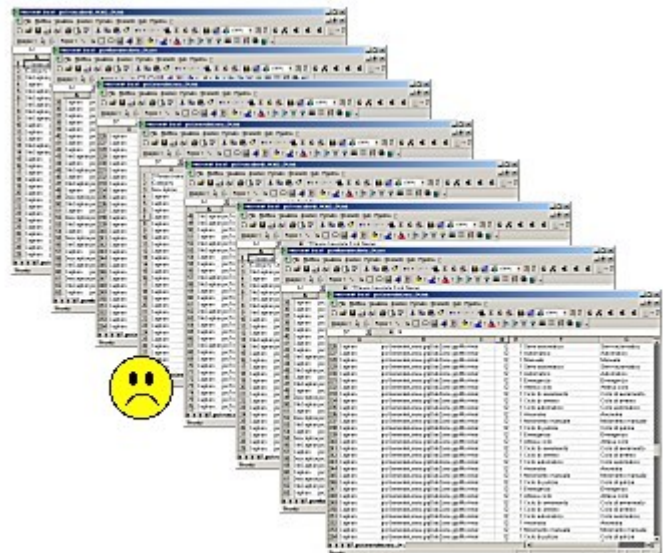
**FX**Interpreter

Multilanguage management module for **GE Fanuc iFix**

**A single Excel file for the whole project  
and all the languages !**

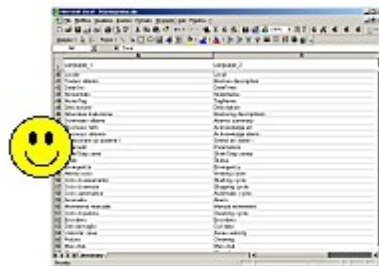
For a medium-sized project the multilanguage management of iFix works with **TENS of CSV files with strings that are NOT UNIQUE!!!**

A lot of working hours are required to compile or modify all these files...



With FXInterpreter you work with only **A SINGLE Excel file for all the languages (dictionary) with UNIQUE strings!!!**

Translations can also be typed or modified by the end user and the dictionary is reusable.



Moreover many **additional functions** are available for a superior flexibility.

**ACTIVA** develops software applications and supplies analytical and advice services. To implement SCADA systems (Supervisory Control And Data Acquisition) **iFix of GE Fanuc** is the preferred environment.

The idea for a tool to speed up and simplify multilanguage management comes from programming experience and market demands.

ACTIVA FXInterpreter is the low cost answer to requests from developers and end users who don't want to complicate their lives!

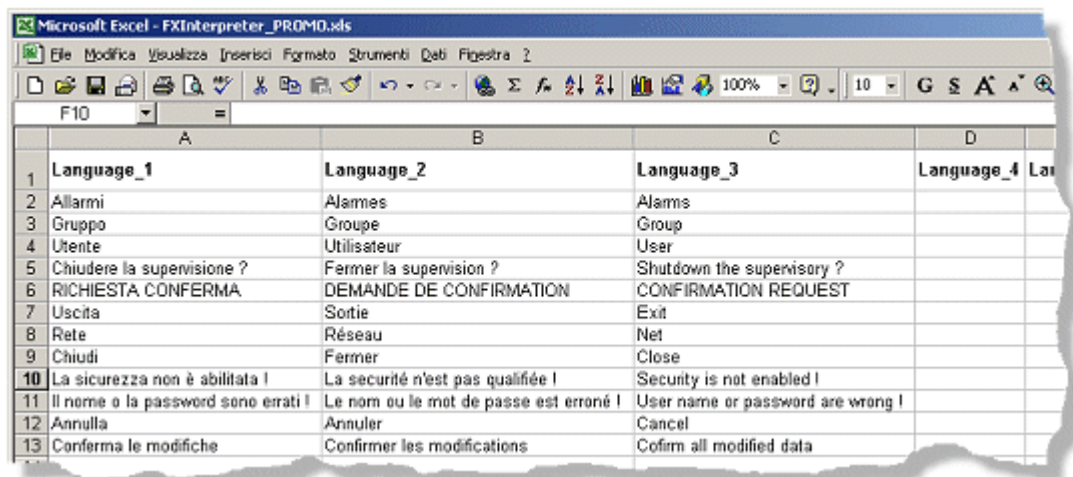
## INTRODUCTION

The development of applications for the foreign market almost always involves the need to address the management of one or more optional languages.

iFix 4.0 has added features that allow you to export and manage the textual content of pictures in specific CSV files (a file for every picture and every language). Each string in a text object (Fix2Dtext) or in a button object (CommandButton) corresponds to a line in the exported file. So in each file there are as many rows as there are repetitions of the same string in a single picture. For example, to translate or modify a single string repeated 10 times in each of the 30 pictures of an application with 2 foreign languages, it is necessary to edit 600 lines of CSV text.

ACTIVA FXInterpreter is a tool integrated in iFix that can greatly simplify multilanguage management and adds functionalities not available in iFix.

**The dictionary for all the pictures and all the languages consists of a single Excel file with unique strings.** The compilation of strings in the development language is automatic and translations must be manually typed in a single line.



	A	B	C	D
1	Language_1	Language_2	Language_3	Language_4 Lan
2	Allarmi	Alarmes	Alarms	
3	Gruppo	Groupe	Group	
4	Utente	Utilisateur	User	
5	Chiudere la supervisione ?	Fermer la supervision ?	Shutdown the supervisory ?	
6	RICHIESTA CONFERMA	DEMANDE DE CONFIRMATION	CONFIRMATION REQUEST	
7	Uscita	Sortie	Exit	
8	Rete	Réseau	Net	
9	Chiudi	Fermer	Close	
10	La sicurezza non è abilitata !	La sécurité n'est pas qualifiée !	Security is not enabled !	
11	Il nome o la password sono errati !	Le nom ou le mot de passe est erroné !	User name or password are wrong !	
12	Annulla	Annuler	Cancel	
13	Conferma le modifiche	Confirmer les modifications	Cofirm all modified data	

The main **immediate advantages** are as follows:

- multilingual implementation time reduced by approximately 90%
- multilingual maintenance time reduced by approximately 99%
- no need to compile a document with unique strings for a translation service
- translation is extended to other objects
- possibility of translations or quick changes by the end user (no risk to the application)
- dictionary is reusable and expandable over time

## COMPARISON

iFix versions prior to 4.0 do not have native multilanguage management.

**FXInterpreter has been tested with iFix 3.5, 4.0 and 4.5.**

	iFix <b>without</b> FXInterpreter	iFix <b>with</b> FXInterpreter	Notes
Multilanguage management	✓	✓	It is possible to manage multiple foreign languages.
Global language	✓	✓	For all pictures.
Local language	✓	✓	It is possible to translate a picture in a local language, different from the global one.
Picture text and captions translation (Fix2DText and CommandButton)	✓	✓	Tooltips are included.
Picture objects translation in "development mode".	✓	✓	Allows you to replace the development language of pictures for easier maintenance by the end user.
A text CSV file for each picture and each language	☹		Eg. 30 pictures with 2 foreign languages → 60 files !
NOT unique strings in each CSV file	☹		Eg. 10 identical strings in 30 picture with 2 foreign languages → 600 rows to edit *
Manual export	☹		For each new picture or change (in a existing picture) or new language, it is necessary to export all the strings and to type all the translations.
Single Excel file		✓	A single file for <u>ALL</u> pictures and <u>ALL</u> languages.
Unique strings		✓	Each string appears once in the dictionary. Eg. 10 identical strings in 30 pictures with 2 foreign languages → only 1 row to edit.
Original dictionary is compiled automatically		✓	The list of the original strings to translate is simply generated by browsing the project in runtime or with a specific global command.
Dictionary is reusable in other projects		✓	Strings already in the dictionary are immediately translated without any further intervention. The dictionary can be expanded over time.

User global language		✓	The global language can be automatically changed on the basis of the user name at login.
User local language		✓	The local language of a picture can be automatically set on the basis of the registered user name.
Translation of default VBA objects (all)		✓	Tooltips are included.
Strings translation in MsgBox and InputBox		✓	It is also possible to translate strings with variables inside.
Historical alarms translation		✓	It is possible to translate historical alarms and to restore descriptions if they are partially cut off by log process.
Translation without pictures reopening		✓	It is NOT necessary to close and to reopen the pictures after language change.
Selection of the objects to be translated on the basis of an optional prefix		✓	The prefix can be set by the developer (eg. "TR_"). The dictionary is further reduced.
Third part ActiveX translation		✓	Quotation on request. Textual properties must be accessible.
Recognition of spaces before and after the strings		✓	The dictionary contains only "clean" strings, while in runtime it is possible to see all the spaces added for aesthetic reasons. **
Implementation time – <b>90%</b>		✓	Realistic estimate.
Maintenance time – <b>99%</b>		✓	Realistic estimate.
<b>NOT</b> necessary to prepare a document for a translation service		✓	It is not necessary to extrapolate a unique document to send to an external translation service.
Simplification of changes by the end user (without any risk to the application)		✓	The uniqueness of the dictionary makes any changes both fast and easy.

(\*) With iFix CSV files it is NOT possible to use the automatic function "Search and replace" of a textual editor (like Notepad), because translations don't replace original strings, but they have to be added.

(\*\*) For example the string " Historical archive" is registered as "Historical archive", translated in the Italian dictionary as "Archivio storico" and shown as " Archivio storico".

## SETUP

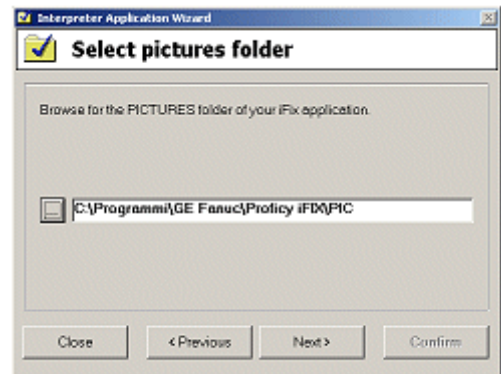
- Run "Setup.exe" and follow instructions.
- Read the EULA file (End User License Agreement) and this User Guide.



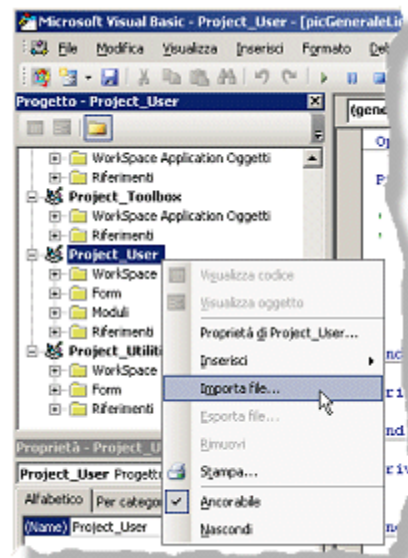
## ADD TO THE PROJECT

To add FXInterpreter to an iFix application, proceed as follow.

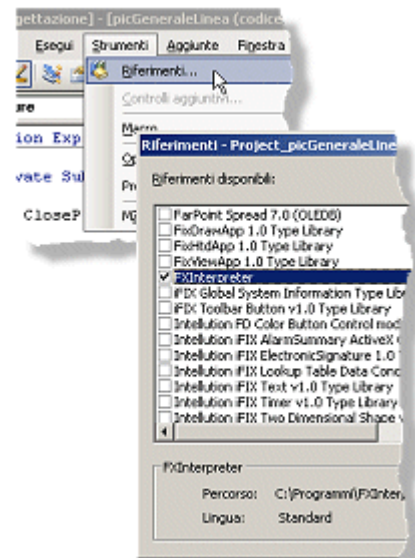
1. From the menu "Start / Programs / Activa / FXInterpreter" start the "Interpreter Application Wizard" to create the FXInterpreter files in the iFix application.



2. Open the iFix application, open the VBA environment and import the module "modInterpreter.bas" in the "Project\_User" (the module is in the "FXInterpreter" folder, within the pictures folder).



3. In the "Project\_User" add the reference to the "FXInterpreter" library.



4. Open and edit the configuration file "FXInterpreter.ini" in the "FXInterpreter" folder of the application (if necessary).

## TO BEGIN

If you don't have a license for FXInterpreter, the iFix hardware key must NOT be used: the module will run for **2 hours in demo mode** (just like iFix).

The multilanguage management for all the pictures can be achieved in a few simple steps:

1. Install and insert FXInterpreter in the application, as described above.
2. Read the chapter "TIPS". In particular it is recommended to use a prefix to identify objects to be translated and reduce the size of the dictionary.
3. Before opening a picture, insert the following code:

```
Interpreter.TranslatePicture "PictureName" (with quotes)
```

Example:

```
Interpreter.TranslatePicture "picMainView"
OpenPicture "picMainView"
```

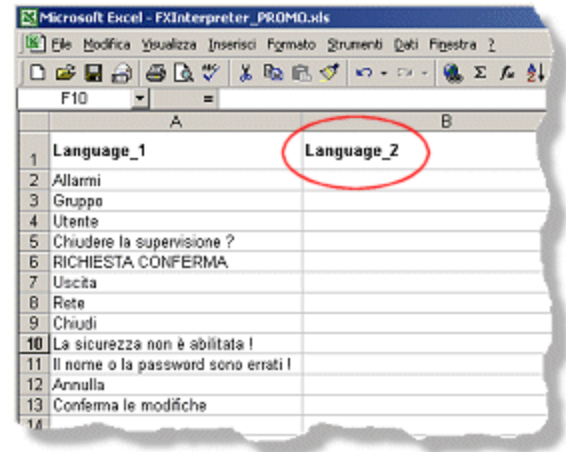
4. At the end of the "CFixPicture\_Initialize" event code of pictures that open automatically at startup, insert the following code (as written):

```
Interpreter.TranslatePicture Me.Name
```

5. Run the project and open pictures to automatically edit the development language column in the dictionary or look at "Dictionary global update and check for all pictures".



6. Close the Workspace and open the Excel file "FXInterpreter.xls" (in the "FXInterpreter" folder of the project) to type in translations.



7. Restart the Workspace and verify translations in graphics.

To disable the FXInterpreter initial splash screen, click on the "Startup configuration" button and set the parameter "ShowInfoPanelOnStartup=NO" in the configuration file "FXInterpreter.ini".

To test the multilanguage management functions, it is also possible to use the files in the folder "Example", in the installation path:

1. Copy the pictures "FXInterpreterTest\_Main" and "FXInterpreterTest\_Popup" in a new iFix application.
2. Add FXInterpreter to the application, as described in the previous chapter.
3. Overwrite the file "FXInterpreter.xls" in the folder "FXInterpreter" of the application with the one you can find in the folder "Example".
4. Open the picture "FXInterpreterTest\_Main" and run the application.

## TIPS

- Translations included in the dictionary should not be longer than strings used in graphics, for obvious reasons of space.
- Translations should be unique just like all the strings in the development language column of the dictionary; in this way it is possible to obtain a unique relationship between strings in different languages.
- Just as in the multilanguage management of iFix, text animations must be implemented through the visibility of more overlapping Fix2DText objects, i.e. direct animation of the caption of a single object should not be used (with a local or global strings table).
- To further restrict the dictionary to include only strings of interest, you can identify items to be translated using a prefix in the name of the objects. To obtain this, it is enough to open the "FXInterpreter.ini" file (in the "FXInterpreter" folder of the project) and type "GetObjectsPrefix = YES"; the default prefix "ObjectsTargetPrefix" is "TR\_" (TRAnslate), but it can be changed as desired.

If "GetObjectsPrefix=NO" then all the Fix2DText and commandButton objects (with tooltips) will be translated, as well as all the VBA default objects. In this case, as in multilanguage management of iFix, it is essential that, for viewing numeric tags, the developer uses DataLink objects and not Fix2DText.

- Apart from the use of FXInterpreter, it is a good rule that the names of pictures, forms and objects do not contain accented letters or graphic signs characteristic of particular languages.

- To prevent errors in displaying translations, verify any system settings in the Control Panel of Windows ("Control panel \ Regional and language options \ Advanced \ Language for non-Unicode programs") and provide for changes BEFORE creating graphics. For example: to display translations in Bulgarian with English version of Windows, it is recommended that you select "Bulgarian" from the drop down list.

**The issues related to the language of the operating system, the fonts and system settings do NOT depend on FXInterpreter; in these cases it is useful to read the online help of iFix. In particular, note that Windows localization limits the languages you can switch between!**

- NOTE:**  
The property "Comments" of pictures is used by FXInterpreter, then it cannot be set by the iFix developer.  
The property "Tag" of VBA forms is used by FXInterpreter, then it cannot be set by the iFix developer.

## SINTAX

### Translation of a picture

To translate the contents of a picture you can take two different ways.

- The code for the translation of a picture should be add before opening the picture itself (OpenPicture, OpenTGDPicture, etc.).

To translate to the global language specified in the configuration file or through the property "Interpreter.GlobalLangNum":

```
Interpreter.TranslatePicture "PictureName"           (with quotes)
or
Interpreter.TranslatePicture "PictureName.grf"      (with quotes)
```

Example:

```
Interpreter.TranslatePicture "picMainView"
OpenPicture "picMainView"
```

To translate to a specific language (local):

```
Interpreter.TranslatePicture "PictureName", LanguageNumber
```

Example:

```
Interpreter.TranslatePicture "picMainView", 3
OpenPicture "picMainView"
```

To translate to the language of a specific user (listed in the configuration file):

```
Interpreter.TranslatePicture "PictureName",, "UserName" (with quotes)
```

Example:

```
Interpreter.TranslatePicture "picMainView",, "Bob"
OpenPicture "picMainView"
```

- To translate a picture you can also use a single instruction (always the same for all pictures) at the end of the script "CFixPicture\_Initialize":

```
Interpreter.TranslatePicture Me.Name
```

or

```
Interpreter.TranslatePicture Me.Name, LanguageNumber
```

or

```
Interpreter.TranslatePicture Me.Name,, "UserName" (with quotes)
```

#### NOTES

- The method n.2 should be used only in pictures that open at startup of the project, while it is not recommended in other cases because the "CFixPicture\_Initialize" event is NOT prior to the opening of the picture and then the change of language is visible at run-time.
- Instead of writing the explicit name of the user, obviously it is possible to use a variable *strUserName* (string type) containing the user name.

### **Change of language with translation of all visible pictures**

To set a specific global language and immediately translate all visible pictures, insert the following code in the script of a command button:

```
Interpreter.GlobalLangNum = LanguageNumber  
Interpreter.TranslateAll
```

Example:

```
Interpreter.GlobalLangNum = 2  
Interpreter.TranslateAll
```

To set the language of a specific user (listed in the configuration file) and immediately translate all visible pictures, insert the following code in the script of a command button:

```
Interpreter.SetUserGlobalLang "UserName" (with quotes)  
Interpreter.TranslateAll
```

Example:

```
Interpreter.SetUserGlobalLang "Bob"  
Interpreter.TranslateAll
```

#### NOTES

Instead of writing the explicit name of the user, obviously it is possible to use a variable *strUserName* (string type) containing the user name.

### **Translation of a UserForm (VBA)**

To translate the contents of a form you can take two different ways.

1. The code for the translation of a form can be inserted before opening the form itself ("FormName Show").

To translate to the global language specified in the configuration file or through the property "Interpreter.GlobalLangNum":

```
Interpreter.TranslateForm FormName (without quotes)
```

Example:

```
Interpreter.TranslateForm frmDataList  
frmDataList.Show
```

To translate to a specific language (local):

```
Interpreter.TranslateForm FormName, LanguageNumber
```

Example:

```
Interpreter.TranslateForm frmDataList, 3  
frmDataList.Show
```

To translate to the language of a specific user (listed in the configuration file):

```
Interpreter.TranslateForm FormName,, "UserName" (with quotes)
```

Example:

```
Interpreter.TranslateForm frmDataList,, "Bob"  
frmDataList.Show
```

2. To translate a form you can also use a single instruction (always the same for all forms) at the end of the script "UserForm\_Initialize":

```
Interpreter.TranslateForm Me
```

or

```
Interpreter.TranslateForm Me, LanguageNumber
```

or

```
Interpreter.TranslateForm Me,, "UserName"
```

#### NOTES

- The method n.2 is recommended because the VBA event "UserForm\_Initialize" is prior to the opening of the form, then the change of language is not visible at runtime and the use of a single instruction in a single point of the form makes easier implementing multilanguage management.
- Instead of writing the explicit name of the user, obviously it is possible to use a variable *strUserName* (string type) containing the user name.

### **Change of language with translation of all visible pictures and forms**

To set a specific global language and immediately translate all visible pictures and forms, insert the following code in the script of a command button:

```
Interpreter.GlobalLangNum = LanguageNumber  
Interpreter.TranslateAll UserForms
```

Example:

```
Interpreter.GlobalLangNum = 2  
Interpreter.TranslateAll UserForms
```

To set the language of a specific user (listed in the configuration file) and immediately translate all visible pictures and forms, insert the following code in the script of a command button:

```
Interpreter.SetUserGlobalLang "UserName" (with quotes)  
Interpreter.TranslateAll UserForms
```

Example:

```
Interpreter.SetUserGlobalLang "Bob"  
Interpreter.TranslateAll UserForms
```

## NOTES

- FXInterpreter will translate only forms of the “calling” picture (containing the above code).
- Instead of writing the explicit name of the user, obviously it is possible to use a variable *strUserName* (string type) containing the user name.

## Translation of MsgBox and InputBox

To translate a string in the language of the calling picture or form, use the following code:

```
Interpreter.TranslateString("String", Me)
```

To translate a string in a specific language, use the following code:

```
Interpreter.TranslateString("String", , LanguageNumber)
```

Example:

the message

```
MsgBox "The max value " & _
      lngMaxVal & _
      " has been exceeded!", _
      vbExclamation, "Attention"
```

can be translated with the following code

```
MsgBox Interpreter.TranslateString("The max value ", Me) & _
      lngMaxVal & _
      Interpreter.TranslateString(" has been exceeded!", Me), _
      vbExclamation, Interpreter.TranslateString("Attention", Me)
```

equivalent (for example) to the following code (Italian)

```
MsgBox "Il valore massimo " & _
      lngMaxVal & _
      " è stato superato!"
      vbExclamation, "Attenzione"
```

## Translation of real-time alarms (AlarmSummary)

The content of "AlarmSummaryOCX" is not accessible, but FXInterpreter can simulate the translation through the visibility of several objects of the same type (overlapping) on the basis of the language entered in the property "Description".

Example:

If the object AlarmSummaryOCX1 displays alarms in the language 1 (through the field AlarmUserField1" of DataBlocks) and object AlarmSummaryOCX2 displays alarms in the language 2 (through the field "AlarmUserField2"), then just use the following settings directly in the window "Properties" of the two objects:

```
AlarmSummaryOCX1.Description = "Language_1"
AlarmSummaryOCX2.Description = "Language_2"
```

## Translation of historical alarms

Alarms can be automatically archived by iFix in a database or in specific text files.

Whatever the method you choose, it is sufficient to obtain a recordset (via ADO) with the data you want to translate; then the translated recordset can be assigned to a grid for viewing.

The user may request a recordset translated in the local language of the calling container (Picture / UserForm) or he can specify a language for translation.

The original language of historical alarms descriptions can be different from the development language of graphics. If the original language is not specified (the language to translate from), it is assumed that it is identical to the development language of graphics.

The complete syntax is quite complex, but the following examples clarify the use (parameters between square brackets are optional):

```
Set rsAlm = Interpreter.TranslateAlarmsRecordset(RecordseToTranslate,
FieldToTranslate, [Me], [RequestedLanguageNumber], [OriginalLanguageNumber],
[RestoreOriginalStrings])
```

where

*RecordseToTranslate* is the ADO recordset containing alarms.

*FieldToTranslate* is the ordinal number (zero based) of the field you want to translate.

*Me* (alternative to "RequestedLanguageNumber") is a reference to the picture or form that requires the translation to its current language; it must be entered as written.

*RequestedLanguageNumber* (alternative to "Me") is the number of the specific requested language.

*OriginalLanguageNumber* is the number of language from which you want to translate, to be entered only if other than the development language of graphics.

*RestoreOriginalStrings* (True / False) indicates the optional request for "translation" even when the language is the same, in order to get the completion of any strings cut off by the log service.

You can repeat the instruction to translate more fields.

In order to exploit the possibility of strings completion, all original complete strings should be included in the dictionary by hand, in addition to translations, preventing automatic insertion by FXInterpreter (with incomplete strings).

#### ATTENTION

To simplify the dictionary for graphics and to avoid slowing down translations in real time, **dictionary for alarms is contained in the Excel file "FXInterpreter\_Alm.xls"**.

#### Example 1

To translate the field 4 of the recordset "rsAlm" from the development language to the current language of the picture:

```
Set rsAlm = Interpreter.TranslateAlarmsRecordset(rsAlm, 4, Me)
```

#### Example 2

To translate the fields 4 and 5 of the recordset "rsAlm" from the development language to the current language of the picture:

```
Set rsAlm = Interpreter.TranslateAlarmsRecordset(rsAlm, 4, Me)
Set rsAlm = Interpreter.TranslateAlarmsRecordset(rsAlm, 5, Me)
```

### Example 3

To translate the field 4 of the recordset "rsAlm" from the language 2 (different from the development language of graphics) to the current language of the picture:

```
Set rsAlm = Interpreter.TranslateAlarmsRecordset(rsAlm, 4, Me, , 2)
```

### Example 4

To translate the field 4 of the recordset "rsAlm" from the language 2 (different from the development language of graphics) to the language 1:

```
Set rsAlm = Interpreter.TranslateAlarmsRecordset(rsAlm, 4, , 1, 2)
```

### Example 5

To obtain completion of the field 4 in the same language of the original strings (i.e. to force translation even when the requested language is the same as the log language), you can use the following instruction:

```
Set rsAlm = Interpreter.TranslateAlarmsRecordset(rsAlm, 4, Me, , , True)
```

## Dictionary update

To obtain automatic filling of the list of strings in the development language column or verify the presence of translations in a specific language, simply browse the application in run mode.

It is recommended that you provide during the development to the manual compilation of the original strings in MsgBox and InputBox (with translations), as these user interface elements are normally associated with contextual situations not easily reproducible in advance.

It is also recommended to close the Workspace before opening the Excel file to manually enter translations.

## Dictionary global update and check for all pictures

With regard to one or more pictures, to automatically fill the list of strings in the development language column and verify the presence of translations in a specific language, you can proceed as follows:

1. Close all pictures
2. Open the VBA environment
3. Open the "Project\_User"
4. Open the "Immediate" window
5. In the "Immediate" window, type the following code:  

```
Interpreter.UpdateAndVerifyDictionary
```

and push the key "Enter" on the keyboard
6. In the window that appears, select one or more pictures and enter the number of the language you wish to verify
7. Push "Ok" and wait for results

It is recommended to close the Workspace before opening the Excel file to manually enter translations.

**NOTE:** this function supports only pictures (strings in "tag group files", UserForms, MsgBox and InputBox are excluded).

## Change of development language in pictures

To automatically change the development language of ALL pictures, it is possible to proceed as follows:

1. Close all pictures
2. Open the VBA environment
3. Open the "Project\_User"
4. Open the "Immediate" window
5. In the "Immediate" window, type the following code:  
*Interpreter.ChangeDevLang*  
and push the key "Enter" on the keyboard
6. In the window that appears, select all pictures of your application and insert the number of the new desired development language.
7. Push "Ok" and wait for results.

It is recommended to close the Workspace before opening the Excel file to manually enter translations.

### NOTE:

- Always make a backup of the whole pictures folder before changing the development language.
- This function supports only pictures (strings in "tag group files", UserForms, MsgBox and InputBox are excluded).

## Configuration file

If the initial panel of FXInterpreter contains a command button that opens the configuration file "FXInterpreter.ini". If the initial panel does not appear at the startup of the project, to see the contents of the configuration file (without browsing the project folders, you can proceed as follows:

1. Open the VBA environment
2. Open the "Project\_User"
3. Open the "Immediate" window
4. In the "Immediate" window, type the following code:  
*Interpreter.ShowStartupConfig*  
and push the key "Enter" on the keyboard

## LICENSE

Each FXInterpreter license is combined with a specific iFix hardware key.

If the iFix hardware key is not detected at Workspace startup, FXInterpreter doesn't ask for any license and can be used for 2 hours in demo mode (just like iFix). Then if you buy a license, you accept all the features.

To purchase one or more licenses, you have to proceed as follows:

1. Send your request to [order@activasoft.it](mailto:order@activasoft.it) with following data :

*Name and surname*

*Company*

*Full address*

*Telephone*

*E-mail*

*License type (max pictures number)*

*Number of licenses*

*Serial number of each iFix hardware key*

*Complete data for invoice*



2. Wait for confirmation from ACTIVA with order code and bank data (IBAN) for payment.
3. Make the payment specifying the order code and wait for licenses and invoice.
4. Copy license files in the FXInterpreter installation folder (default is "C:\Program files\FXInterpreter") or in the "FXInterpreter" project folder.

In the case of a license for a development key, it is advisable to copy the file in the FXInterpreter installation folder.

In the case of licenses for runtime keys, it is advisable to include all license files in the "FXInterpreter" folder of the application, in order to avoid having to modify its content with different clients/servers or hardware keys.

## WEB RESOURCES

For information, prices and contacts:

Web site	<a href="http://www.activasoft.it">www.activasoft.it</a>
Informations	<a href="mailto:info@activasoft.it">info@activasoft.it</a>
Orders	<a href="mailto:order@activasoft.it">order@activasoft.it</a>
Technical support	<a href="mailto:support@activasoft.it">support@activasoft.it</a>



© 2006-2008 ACTIVA di Gismondi Roberto - Italy. All rights reserved.