

Table of contents:

- [1 - Info](#)
- [2 - KiCad tree for translations:](#)
 - [2.1 - Dictionary tree:](#)
 - [2.2 - Search path:](#)
 - [2.3 - Files:](#)
- [3 - Using poedit](#)
 - [3.1 - Installation](#)
 - [3.2 - KiCad preparation](#)
 - [3.3 - Poedit Configuration](#)
 - [3.4 - Project Configuration:](#)
 - [3.5 - Path and files Configuration:](#)
 - [3.6 - Keyword Configuration:](#)
 - [3.7 - Save the project:](#)
- [4 - Create or edit a dictionary:](#)
- [5 - Adding a new language entry in KiCad source code](#)
 - [5.1 - Steps:](#)
 - [5.1.1 - Adding a new id in include/id.h.](#)
 - [5.1.2 - Adding a new icon \(aesthetic purpose only\)](#)
 - [5.1.3 - Editing common/edaappl.cpp](#)
 - [5.1.4 - Recompiling](#)

1 - Info

The different menus and tool tips in KiCad are internationalized, and can be easily translated into a local language **without source code modifications**.

The rules are:

- They are written in english.
- All strings which must be translated are written like: _("hello world"), and displayed "hello world" but if a dictionary is found translated into the locale language before displaying.
- A dictionary **english->locale** handle translation (one dictionary by language).

The easier way to create and maintain the dictionary **english->locale** is to use, **poedit**. (www.poedit.org).

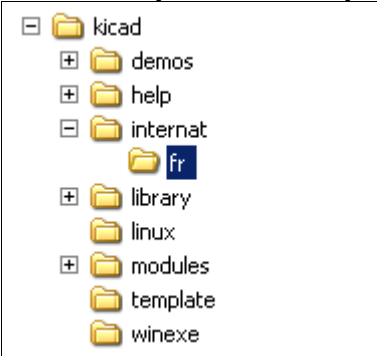
Poedit scans KiCad sources and allows you to enter translations.

You must download KiCad sources and set poedit in order to create translations.

2 - KiCad tree for translations:

2.1 - Dictionary tree:

The dictionary will be found by Kicad only if it is in a suitable path:

	<p>The suitable path is kicad/internat/xx, or kicad/internat/xx_yy with: xx = normalised locale indicator (short form) like:</p> <ul style="list-style-type: none"> • fr = france • en = english • es = spanish • pt = portuguese <p>or: xx_yy = normalised locale indicator (long form) like:</p> <ul style="list-style-type: none"> • fr_FR • en_GB • en_US
---	--

2.2 - Search path:

Dictionaries and on-line help files are searched in this order:

- In the path in normalised locale indicator (long form) (kicad/internat/xx_yy)
- In the path in normalised locale indicator (short form) (kicad/internat/xx)

And for on-line help files search is made in

- In the path in normalised locale indicator (long form) (kicad/help/xx_yy)

GUI Translation HOWTO

- In the path in normalised locale indicator (short form) (kicad/help/xx)
- kicad/help/en
- kicad/help/fr

Note:

The main KiCad path is retrieved from the binary path, or (if not found):

under windows:

- c:\kicad
- d:\kicad
- c:\Program Files\kicad

or under linux:

- /usr/share/kicad
- /usr/local/share/kicad
- /usr/local/kicad/share/kicad
- /usr/local/kicad

2.3 - Files:

In each directory there are 2 files **kicad/internat/xx**:

- internat.po (the dictionary file)
- internat.mo (the poedit work file)

3 - Using poedit

3.1 - Installation

Download and install poedit (www.poedit.org).
Poedit exists for Windows, Linux and Mac OS X.

Download and unzip KiCad sources.

3.2 - KiCad preparation

KiCad sources: in this example files are in [f:/kicad/](#).

All the strings to translate are tagged like **_("string to translate")**.

poedit must search the **_** (underscore) symbol to locate these strings.

One must add in KiCad the suitable directory for the dictionary (**kicad/internat/xx**).

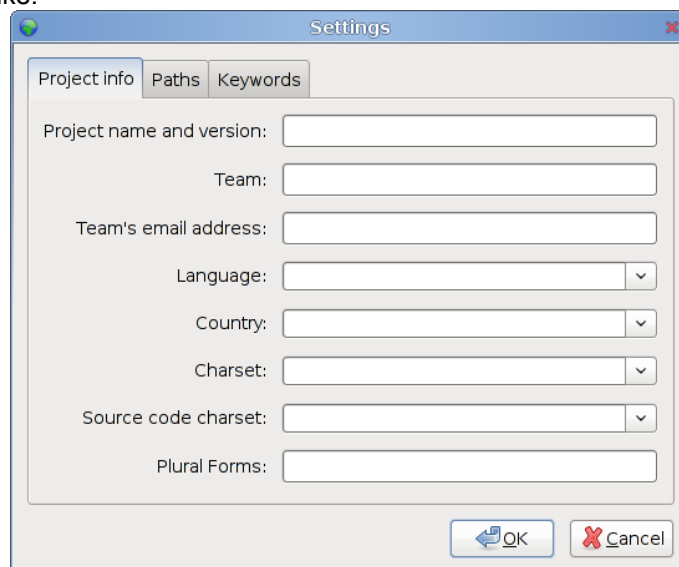
In this example, the directory is **kicad/internat/fr**.

3.3 - Poedit Configuration

Run poedit.

Run **File/New catalog...**

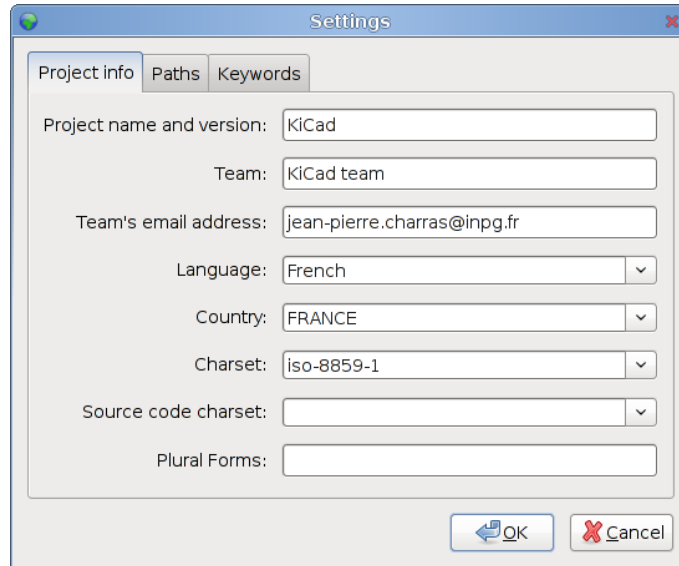
You should see something like:



GUI Translation HOWTO

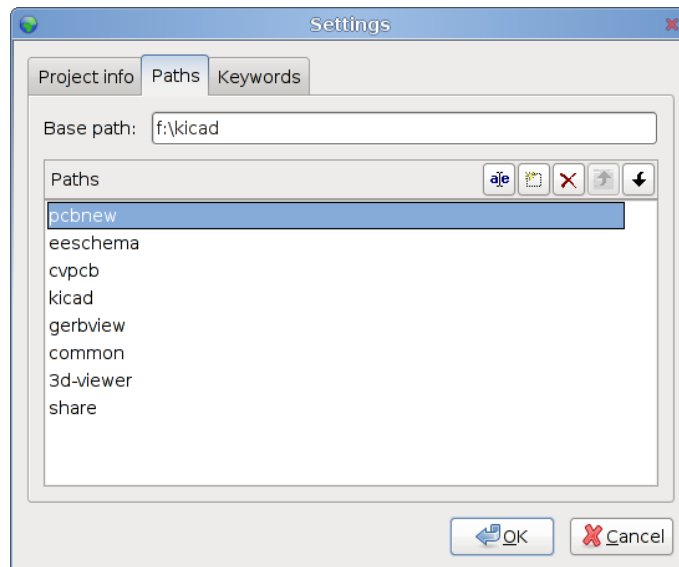
3.4 - Project Configuration:

Here is the configuration for the french translation:

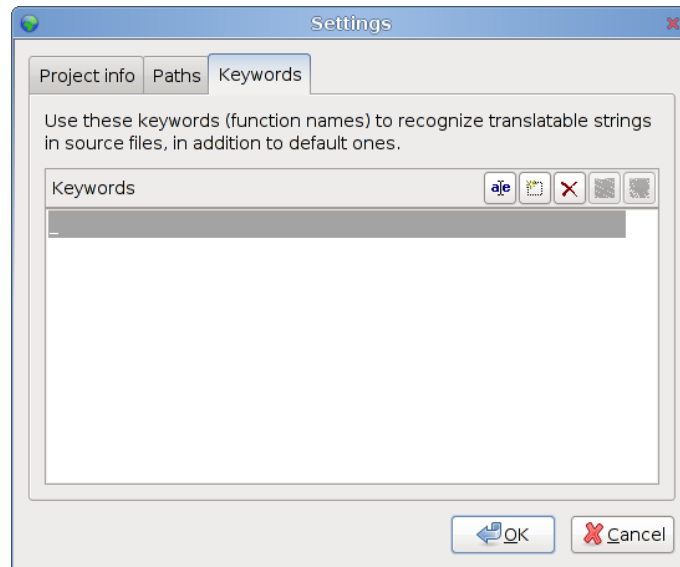


The source files are in english, so no need to choose something for source code.

3.5 - Path and files Configuration:



3.6 - Keyword Configuration:



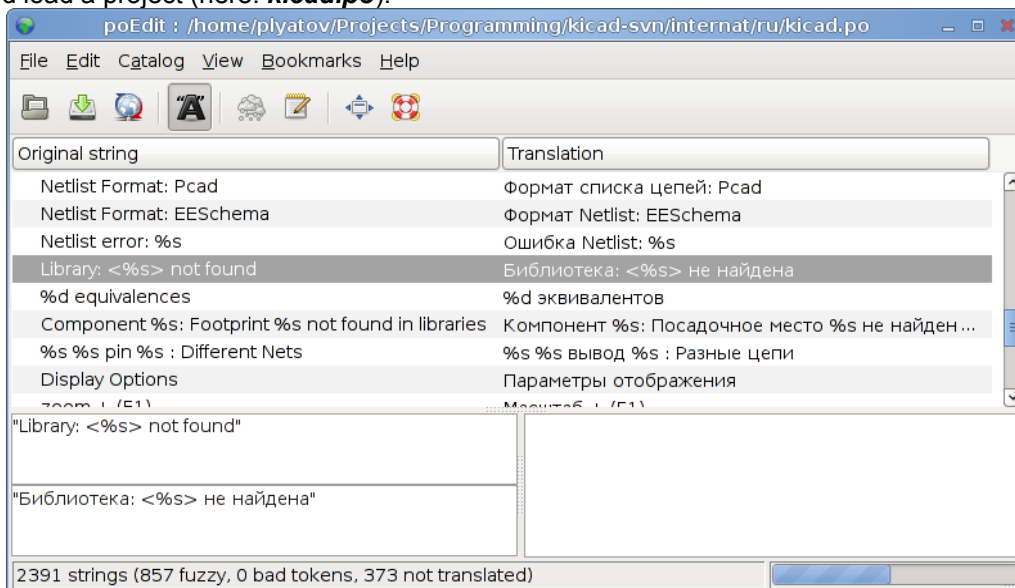
Only one keyword to enter: _ (underscore) used as tag in source files

3.7 - Save the project:

Save the new project in *kicad/internat/xx* with the name *kicad.po*

4 - Create or edit a dictionary:

Run poedit and load a project (here: *kicad.po*).



Run the command **Catalog/update from sources**.

New strings (not yet translated) will be displayed on the top of the window.

5 - Adding a new language entry in KiCad source code

This step is NOT required.

It is usefull only for developpers, and for testing purpose only

In KiCad we can force the used language.

It is highly recommended to use the default language.

GUI Translation HOWTO



But because developers must test translations, a new entry in the language list can be useful for testing purposes.

5.1 - Steps:

5.1.1 - Adding a new id in include/id.h.

➔ In *include/id.h*, locate the sequence like:

```
ID_LANGUAGE_CHOICE,  
ID_LANGUAGE_DEFAULT,  
ID_LANGUAGE_ENGLISH,  
ID_LANGUAGE_FRENCH,  
ID_LANGUAGE_SPANISH,  
ID_LANGUAGE_GERMAN,  
ID_LANGUAGE_RUSSIAN,  
ID_LANGUAGE_PORTUGUESE,  
ID_LANGUAGE_ITALIAN,  
ID_LANGUAGE_SLOVENIAN,  
ID_LANGUAGE_HUNGARIAN,  
ID_LANGUAGE_POLISH,  
ID_LANGUAGE_KOREAN,  
ID_LANGUAGE_CATALAN,  
ID_LANGUAGE_UNUSED3,  
ID_LANGUAGE_UNUSED4,  
ID_LANGUAGE_CHOICE_END,
```

and add a new entry in list (which will be used later in menus) like:

`ID_LANGUAGE_MY_LANGUAGE` (one can replace a line like `ID_LANGUAGE_UNUSED3` if exists to do that).

5.1.2 - Adding a new icon (aesthetic purpose only)

➔ Create a new icon in xpm format: usually the country flag.

Others language icons are in *common/bitmaps*

This is text like:

GUI Translation HOWTO

```
/* XPM */
static const char * lang_fr_xpm[] = {
"16 16 5 1",
"   c None",
".   c #000000",
"+   c #0000D2",
"@   c #FFFFFF",
"#   c #F00000",
"
",
"
",
".....",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".++++@@@@####.",
".....",
"
",
"
",
"
"};
```

This is a new file like *common/bitmaps/Lang_My_language.xpm*, starting by something like:

```
/* XPM */
static const char * lang_my_language_xpm[] = {
"16 16 5 1",
"   c None",
```

5.1.3 - Editing common/edaappl.cpp

➔ locate the text:

```
#ifdef __WINDOWS__
/* Icons for language choice (only for Windows)*/
#include "Lang_Default.xpm"
#include "Lang_En.xpm"
#include "Lang_Es.xpm"
#include "Lang_Fr.xpm"
#include "Lang_Pt.xpm"
#include "Lang_It.xpm"
#include "Lang_De.xpm"
#include "Lang_Sl.xpm"
#include "Lang_Hu.xpm"
#include "Lang_Po.xpm"
#include "Lang_Ko.xpm"
#include "Lang_Ru.xpm"
#include "Lang_Catalan.xpm"
#endif
```

and add a line to include the new icon file.

➔ Locate:

```
struct LANGUAGE_DESCR
{
    int          m_WX_Lang_Identifier;           // wxWidget locale
    identifier (see wxWidget doc)
    int          m_KI_Lang_Identifier;         // kicad identifier used in
    menu selection (see id.h)
    const char** m_Lang_Icon;                 // the icon used in menus
    const wxChar* m_Lang_Label;              // Label used in menus
```

GUI Translation HOWTO

```
    bool        m_DoNotTranslate;           // set to true if the
m_Lang_Label must not be translated
};

#define LANGUAGE_DESCR_COUNT 14
static struct LANGUAGE_DESCR s_Language_List[LANGUAGE_DESCR_COUNT] =
{
    {
        wxLANGUAGE_DEFAULT,
        ID_LANGUAGE_DEFAULT,
        lang_def_xpm,
        _( "Default" )
    },
    {
        wxLANGUAGE_ENGLISH,
        ID_LANGUAGE_ENGLISH,
        lang_en_xpm,
        wxT( "English" ),
        true;
    },
    {
        wxLANGUAGE_FRENCH,
        ID_LANGUAGE_FRENCH,
        lang_fr_xpm,
        _( "French" )
    },

```

and add a new entry like:

```
{
    wxLANGUAGE_MY_LANGUAGE,
    ID_LANGUAGE_MY_LANGUAGE,
    lang_my_language_xpm,
    _( "My_language" )
},
```

`wxLANGUAGE_MY_LANGUAGE` is the wxWidgets language identifier for the country (see wxWidget doc).

5.1.4 - Recompiling

Obviously, this is the next and last step.