MCL_INI DEVELOPPER'S KIT

Making MapInfo tools Multi Lingually Compatible with the MLC_INI[1] solution

Documentation and resources for the MapBasic programmer

Jacques Paris, July 2000

jakesp@total.net

[1] The Multi-Lingual Compatibility Project is an initiative of Jacques PARIS (jakesp@total.net). The INI SOLUTION is the brain child of Bill THOEN (bthoen@ctmap.com); it was implemented will the help of Mats Elfström (mats.elfstrom@telia.com) and Carlos Montalvillo Gómez (Carlos.Montalvillo@sgsmap.com)

The MlcInKit.zip file contains a variety of documents and examples. They include in particular the resource libraries that are required to compile and link a project that respects the MLC standard solution, documented examples and this document.

MLC_INI libraries

mlc_ini.def	defines the sub/functions of the mlc_ini.mb library
mlc_ini.mb	contains the sub/functions corresponding to calls made in the application and handle all “language” related functions
profile.def	defines the API related functions to read/write the ini file.
profile.mb	necessary in its MBO form for the linking the project

These files should be placed in a “central” directory in order to keep a unique copy of them; calls to these files should include the complete path to this directory.

PROJECT FILE

Every application requires a project file (*application*.mbp) in order to link the resources libraries to the main application module (and the application modules together if they exist) should read as follows:

[LINK]

application=*application*.mbx	application mbx name
module=*application*.mbo	application main module
module= MLC_INI.mbo	language choice, text “extractor” sub/functions
module=profile.mbo	Windows API Profile sub/functions

PROGRAMMING ELEMENTS

1 – Include statement(s)

> ***note****: specify the complete path to these files if they do not reside in the directory in which the application is developed*

Always required

```
include "mlc_ini.def"
```

Required only if direct calls to the profile sub/functions (ini_GetIni, ini_WriteINI) are made

```
include "profile.def"
```

2 – Declare statements

Minimal set of statements

(Goodbye and About under any other form are recommended, not required)

```
declare sub main
declare sub MenuSetup
declare sub option_lang
declare sub goodbye
declare sub about
declare Sub BuildLanguage (byval sProfile
                           as string)
```

3 – Language initialization

This sub detects if an ini file exists; if not, it will create one with the `BuildLanguage()` subroutine.

It registers the language to use (the original language if a new ini is created or the previously selected one if an ini file exists) and calls for setting the menu

If other parameters are kept in the ini file, it is also the place to read in their values, using the ini_GetIni() function.

```
      dim fileini as string
'=============================================
      sub main

      fileini=Applicationdirectory$()+
                           "APPLICATION.ini"
      if not FileExists (fileini) then
           call BuildLanguage (fileini)
      end if
      call mlc_InitLanguage (fileini)
      call MenuSetup

      end sub
```

4 – Menu creation

In this document, message numbers are written as space fillers only. Installing the menu in the Tool Menu (ID 4) prevents the cluttering of the main menu bar by the menus of all the loaded applications. That option is available only with versions starting with 4.5.

This sub should also contain the definitions of shortcut menus, the modifications to toolbars (addition of icons) and the creation of new tool pads, if any is needed.

```
      sub MenuSetup

      Create Menu msg(8) as
           msg(14) calling option_lang,
           msg(15) calling options,
           "(-",
           msg(9)  Calling About,
           msg(10) Calling Goodbye
      if systeminfo(3)<450 then
           Alter Menu Bar Add msg(8)
        else
           Alter Menu ID 4 Add msg(8) As msg(8)
      end if
      end sub
```

5 – Change of language

The **`mlc_SetLanguageDlg()`** function allows choosing among the languages present in the ini file.

If any other menus or pads are implied in the change of languages, they should be dealt with in this subroutine. Item menus can be simply removed, button pads must be “destroyed”

```
      sub option_lang

      dim sPrevMenu,a as string

      sPrevMenu = msg(8)
      if mlc_SetLanguageDlg () then
           if systeminfo(3)<450 then
                Alter menu bar remove sPrevMenu
             else
                a="Alter menu ID 4 remove
                           """+sprevmenu+"""
                run command a
           end if
           call MenuSetup
      end if
      end sub
```

6 – Writing the strings

The strings corresponding to the text markers (calls to the msg(nn) function) are stored in a special subroutine that must respect the structure and the contents described below.

The size of the array must be adjusted to the exact number of strings.

The call to **`mlc_PrimeLanguage()`** registers the "original" language strings in the ini file. Adjust "English" to the appropriate name if different.

```
Sub BuildLanguage(byval sProfile as string)
dim sMsg(23) as string

sMsg(1) = "\nDirect Access to Preferences"
sMsg(2) = "\nSystem Settings"
sMsg(3) = "\nMap Window"
...
sMsg(22) = "Effective only at the next
                                    loading"
sMsg(23) = "of the application"

call mlc_PrimeLanguage ("English", sMsg,
                                    sProfile)
End Sub
```

7 – Direct calls to the ini_ sub/functions: using the ini file for application parameters

The developer will have to make direct calls to the **Profile functions** necessary to write in or read out a string in this ini file only if more parameters must be maintained in the ini file. The Profile.def file must be "included" in the MB of the application (see 1-). These functions have the following syntax:

subroutine	call **ini_WriteIni** (Group, Keyname, Value, Profile)
and	
function	Value = **ini_GetIni** (Group, Keyname, Default, Profile)

Profile	is the 'name' (string) of the ini file
Value	a string that represents the string after an '=' in the ini file.
Group	the text wrapped in square brackets (e.g. Language1)
Keyname	the identifier before the '=' (e.g. msg7).

for ini_GetIni() only

Default	is the value returned if the Keyname in Group is not found. The provided MB libraries use "?"

Note also that these Profile functions will work in either 16-bit or 32-bit Windows.

An example of their use is provided in the PREFER demo application.

8 – Original Language other than English

It the developer wants to use a language other than English as the original language of the "strings", he must translate the three strings [sMsg(1) to sMsg(3)] of the `mlc_BuildLanguage()` subroutine in the mcl_ini.mb library to this language

```
sub mlc_BuildLanguage (byval sProfile as string)

dim sMsg(3) as string
dim i as smallint
sMsg(1) = "Language option"
sMsg(2) = "Your choice will become active when
you close this dialog"
sMsg(3) = "OK"
for i = 1 to UBound(sMsg)
      call ini_WriteIni ("MLC1", "msg"+Str$(i),
```

```
                                sMsg(i), sProfile)
    next
    end sub
```

9 – Recommendations for designing dialogs

As the translation of a string into another language results almost systematically in a string of different length, it becomes difficult to layout dialog boxes in a way they are perfect in all circumstances. To a visually pleasing layout, to a compact single frame dialog, the developer should prefer safe layouts and multiple sequential dialogs.

Exploding a dialog in several frames is probably easy to understand and implement. Safety in dialogs is harder to imagine; it means that the essential elements of a dialog should not be altered by strings of uncontrollable lengths or that text strings should not be constrained by fixed elements. Here are some basic ideas:

- it is preferable for an EditText box (or any fixed width control) to be placed on the same line before a StaticText rather than after

- do not specify width, particularly for a dialog; it will allow automatic adjustment to any string length without truncation

- use left alignment only; forget about any centering or right alignment.

- specify the position of any element only if it must be given;

- height and varying lengths are generally not related excepted in the case a multiple line StaticText: the translated string length may have more lines than the original. It would be better to use several single line StaticText boxes to avoid this problem.

WARNING:

Each time the application is run, a new INI file is created. If changes are made to the text strings, they will be “activated” only if the **existing INI is deleted before running the program**.

MAIN APPLICATION MODULE : an example PREFER.MB

The **main application module** must contain some specific calls and must handle some situations in a specific way. The following documented example shows the key elements.

Most generic information is detailed in “Programming elements”. Comments included in this listing relates mainly to the PREFER application.

required

```
include "mlc_ini.def"
```

the profile.def is required in this application because of the direct calls to the Profile library for storing/retrieving parameters in the ini file

```
include "profile.def"
```

general sub structure

```
declare sub main
declare sub MenuSetup
declare sub option_lang
declare sub goodbye
declare sub about
declare Sub BuildLanguage (byval sProfile as string)
```

Specific subroutine to manage the extra option

```
declare sub options
```

The irow,ipos are the parameters added to the ini file for this application

```
dim irow,ipos as smallint
```

Fileini is a way to simplify the writing of the ini_ calls; useful particularly when using extra parameters

```
dim fileini as string

'================================================
sub main
fileini=Applicationdirectory$()+"PREFER.ini"
if not FileExists (fileini) then
     call BuildLanguage (fileini)
```

Write extra parameter values (set here at 6)

```
call ini_WriteIni("General","DockedPadRow","6",
                                                fileini)
call ini_WriteIni("General","DockedPadPos","6",
                                                fileini)
end if
call mlc_InitLanguage (fileini)
call MenuSetup
end sub

'================================================
sub MenuSetup
dim sCmd as string
```

Create initial menu or recreate it with a different language
In this example, the main part of the application is a simple menu+toolpad. Normally, there would be a menu item that will launch the operations.

```
Create Menu msg(8) as
    msg(14) calling option_lang,
    msg(15) calling options,
      "(-",
    msg(9)  Calling About,
    msg(10) Calling Goodbye

      if systeminfo(3)<450 then
            Alter Menu Bar Add msg(8)
      else
            Alter Menu ID 4 Add msg(8) As msg(8)
      end if
```

Get extra parameter values

```
irow=ini_GetIni("General","DockedPadrow",MSG_DEFAULT,
                                                fileini)
ipos=ini_GetIni("General","DockedPadPos",MSG_DEFAULT,
                                                fileini)
```

Create buttonpad docked with the irow, ipos parameters. It is the main purpose of this application.

```
create buttonpad msg(8) as
pushbutton calling about icon 230 helpmsg msg(1)
pushbutton calling 210 icon 116 HelpMsg msg(2)
pushbutton calling 212 icon 110 HelpMsg msg(3)
pushbutton calling 215 icon 109 HelpMsg msg(4)
pushbutton calling 211 icon 117 HelpMsg msg(5)
pushbutton calling 213 icon 98 HelpMsg msg(6)
pushbutton calling 214 icon 101 HelpMsg msg(7)
```

```
fixed toolbarposition (irow,ipos)
end sub

'================================================
sub option_lang
dim sPrevMenu,a as string
      This subroutine deals only with the language choice
sPrevMenu = msg(8)
if mlc_SetLanguageDlg () then
      if systeminfo(3)<450 then
            Alter menu bar remove sPrevMenu
        else
            a="Alter menu ID 4 remove
                              """+sprevmenu+""""
            run command a
      end if
      The buttonpad is destroyed here. Its definition cannot be
      saved as that of a menu. It will be recreated with the new
      language by the next call.
      Alter ButtonPad sPrevMenu Destroy
      call MenuSetup
end if
end sub

'================================================
Sub options
      specific subroutine to specify/modify the values of extra
      parameters and register the new choice(s) in the ini file
dim jrow,jpos as smallint
dialog title msg(16)
control staticText title msg(17)+str$(irow)
                              position 10,10
control staticText title msg(18)+str$(ipos)
                              position 20,20
control staticText title msg(19) position 10,35
control edittext value irow into jrow width 15
                              position 20,45 height 10
control edittext value ipos into jpos width 15
                              position 20,58 height 10
control staticText title msg(20) position 45,45
control statictext title msg(21) position 45,58
control statictext title msg(22) position 10,75
control statictext title msg(23) position 10,85
control okbutton
if not commandinfo(1) then exit sub end if
if jrow<>irow then
      call ini_WriteIni("General","DockedPadRow",jrow,
                                              fileini)
      irow=jrow
end if
if jpos<>ipos then
      call ini_WriteIni("General","DockedPadPos",jpos,
                                              fileini)
      ipos=jpos
end if
end sub

'================================================
Sub About
note
msg(11)+chr$(13)+chr$(13)+msg(12)+chr$(13)+chr$(13)+ms
g(13)
end sub

'================================================
Sub GoodBye
  End Program
End Sub

'================================================
Sub BuildLanguage(byval sProfile as string)
dim sMsg(23) as string
      All the strings required for the application. The numbering
      does not respect the relative place of a string in the listing.
      sMsg(1) = "\nDirect Access to Preferences"
      sMsg(2) = "\nSystem Settings"
      sMsg(3) = "\nMap Window"
      sMsg(4) = "\nLegend Window"
      sMsg(5) = "\nStartUp"
      sMsg(6) = "\nAddress Matching"
      sMsg(7) = "\nDirectories"
      sMsg(8) = "Preference"
      sMsg(9) = "About Preferences"
      sMsg(10) = "Remove Preferences"
      sMsg(11) = "Direct access to the various
                              Preferences requesters"
```

```
      sMsg(12) = "This ultra simple program is a
                  demonstration of a multi lingual
                  application using messages from the
                  ini file"
      sMsg(13) = "Jacques Paris under Mats Elfström's
                  influence and serious
                  rework by Bill Thoen, June 2000"
      sMsg(14) = "Language Choice"
      sMsg(15) = "ToolPad Position"
      sMsg(16) = "Position of docked ToolPad"
      sMsg(17) = "ToolPad is now docked on row "
      sMsg(18) = "and in the postion "
      sMsg(19) = "Enter the values you want for"
      sMsg(20) = "row (0 topmost row)"
      sMsg(21) = "position (0 rightmost position)"
      sMsg(22) = "Effective only at the next loading"
      sMsg(23) = "of the application"

call mlc_PrimeLanguage ("English", sMsg, sProfile)
End Sub
```