Coding guidelines for the XPde project

Introduction

The purpose of this guide is to formalize the way to write code for the xpde project, it is specific to the Delphi/Kylix language and it differs in some cases with the standard Borland coding guidelines.

The main reason for these differences is that I usually write code in languages other than Kylix and I always try to find a common way to do it to prevent going crazy.

This guide it doesn't provide a high level of detail on the Delphi/Kylix language, it just documents the way to write code with the purpose to create readable code by all the developers of this project.

It doesn't include any pattern to develop code, but bear in mind that the best solution is always the easiest.

I will expand this guide as long as I see the common pitfalls when sending code to be integrated in the project.

I have tried not to make a big guide that makes writting code boring, so this guide doesn't cover all the aspects of development, but the most important parts. The rest is left to the common sense and the standard borland coding guidelines.

Source Files

There are several types of source files:

- .pas Source code files
- .xfm Forms
- .dpr Projects
- .bpg Project Groups
- .dpk Packages

Source file naming conventions

Source code files (.pas) must be named starting with lowercase u and with capital letters for each word in the name, that is uMyUnit.pas.

Project files (.dpr) must be named thinking that the output executable will be named accordingly, for example sysinfo.dpr to produce a sysinfo executable or taskmanager.dpr to create a taskmanager executable.

Xpde executable names must be all lowercase.

Project groups (.bpg) must be named starting with uppercase and with capital letters for each word, and also, ending with PG, for example MyProjectGroupPG.bpg.

Packages (.dpk) must be named starting with bpl all lowercase.

Indentation

The source code must not contain TABS, all indentation must be done using SPACES and 2 spaces each level, for example, 2,4,6,8, etc...

Whitespaces

Whitespaces must be used only when separating lists and colons, for example, parameter lists:

```
procedure myProcedure(param1: integer; param2: string);
```

But NOT use spaces to separate parenthesis:

```
procedure myProcedure( param1: integer; param2: string );
```

Source file organization

The source file organization must contain the following elements in the specified order:

1. Copyright
2. Unit name
3. Interface
4. Implementation
5. A closing end and a period

Copyright Block

The copyright block must be like this, including references to used/translated code where appropiate.

```
{ ************************************************************************** }
{                                                                            }
{ This file is part of the XPde project                                      }
{                                                                            }
{ Copyright (c) YYYY Author <mail@domain.com>                                }
{                                                                            }
{ Portions translated from XXXXXXXX Copyright (c) XXXXXXXXXXXXXXXX           }
{                                                                            }
{ This program is free software; you can redistribute it and/or              }
{ modify it under the terms of the GNU General Public                        }
{ License as published by the Free Software Foundation; either               }
{ version 2 of the License, or (at your option) any later version.           }
{                                                                            }
{ This program is distributed in the hope that it will be useful,            }
{ but WITHOUT ANY WARRANTY; without even the implied warranty of             }
{ MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU          }
{ General Public License for more details.                                   }
{                                                                            }
{ You should have received a copy of the GNU General Public License          }
{ along with this program; see the file COPYING.  If not, write to           }
{ the Free Software Foundation, Inc., 59 Temple Place - Suite 330,           }
{ Boston, MA 02111-1307, USA.                                                }
{                                                                            }
{ ************************************************************************** }
```

Uses

Uses declaration must be this way:

```
uses
  QForms, QGraphics, Controls,
  QDialogs, uMyUnit;
```

That is, each unit separated by a comma and a space and no more than three units per line.

Classes/Interfaces

Classes must be named starting with T and intefaces must start with I.

```
TMyClass = class

IMyInteface = inteface
```

Naming conventions

Identifiers/Methods

Identifiers should not use underscores (_) unless you are translating header files.
Identifiers must start with lowercase and capitalize each word.

Example:

```
myVariable
myMethod
```

Except those generated by the IDE for set and get methods.

Private Fields

Private members of classes must start with F

```
TMyClass = class
private
  FMyPrivateField: integer;
end;
```

Types

Types must be named all in lowercase

```
myIdentifier: integer;
myVariable: string;
```

When declaring variables, each variable must be on it's own line, for example:

```
//Correct
var
  i: integer;
  b: integer;

//Incorrect
var
  i, b: integer;
```

Custom Types

Must start with T, like classes

```
TMyEnumeration=(meValue1, meValue2, meValue3);
```

Enumeration values must start with an abrevation of the type.

Begin...end blocks

Unlike the standard coding guidelines, begin MUST NOT BE in a new line, for example:

```
//Correct
if condition then begin
  doSomething;
end
else begin
  doSomething;
end;

//Incorrect
if condition then
begin
  doSomething;
end
else
begin
  doSomething;
end;
```

I don't document the rest of language structures, but bear in mind begin must not be in a new line.

Forms and controls/components

The only restriction on Forms naming is the naming convention on modal dialogs, that must end in Dlg, for example, AboutDlg.

To name controls and components on a dialog, there is a golden rule, which says that controls must be named with a prefix (2-3 lowercase letters) that specifies the class of the component, for example:

Prefix	Component
pm	TPopupMenu
lb	TLabel
btn	TButton
…	…

pmOptions, lbInformation, btnStart, etc...