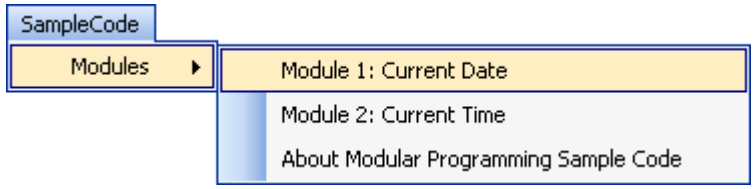
Modular Programming in MapBasic

This document describes how to break down the code of a larger, more complex MapBasic application into several modules.

If you are developing a large, complex application, your program could eventually contain thousands of lines of code. You could type the entire program into a single file, i.e. in one MapBasic Program Source (*.mb) file. However, you can also decide to split the application in several modules. The practice of breaking large programs down into smaller, more manageable pieces is known as modular programming.

What are the benefits of modular programming?

- You can divide your program into numerous, small files. Modular programs are generally easier to maintain in the long run.
- When more programmers are working on a project at the same time, each programmer can work on a separate module.
- You can easily reuse modules in multiple projects – this avoids the need to reinvent the wheel over and over again.



Sample Program

The description of the building blocks of a modular project in this document is based on a sample project: ***Project.MBX***.

You can download the source code of this ***Project.MBX*** from the following location: http://www.twiav.nl/php/mbsampleM2.php#mopr

This application demonstrates how to set up a MapBasic project consisting of several modules.

The actual application - showing current date and current time - is not that exciting at all.

It is the source files you should be after:

- *Project.mbp*
- *ProjectUI.mb*, *Functions.mb*, *ModuleOne.mb* and *ModuleTwo.mb*
- *Project.def*

You can compile the source files (***.mb**) to become object files (***.mbo**) and you can link the object files to become an executable application file (***.mbx**).

This little app is purely meant to be an example on how to set up your own projects.

How to set up a modular project?

The files which together make up a project are:

- The *MapBasic Project File* (***.mbp**). The project file tells the MapBasic linker how to combine separate modules into a single, executable application.

- One or more *MapBasic Program Source* files (***.mb**). These are the modules of the project. When you compile a single module that is part of a multiple-module project, the MapBasic compiler creates a *MapBasic Object File* (***.mbo**)

- One or more *MapBasic Include Files* (***.def**). This file contains information which is shared across modules, e.g. declarations of global variables; declarations of sub procedures; declarations of functions; define codes; etc.

Project File

A MapBasic Project file has the extension *.mbp.

Below you see an example of the content of a project file:

```
[Link]
Application=Project.mbx
Module=ProjectUI.mbo
Module=Functions.mbo
Module=ModuleOne.mbo
Module=ModuleTwo.mbo
```

- The first line should contain the keyword `[Link]`
- The second line should contain the text `Application=appfilename` (where *appfilename* specifies the file name of the executable file you want to create)
- All further lines should contain the text `Module=modulename` (where *modulename* specifies the name of a MapBasic object file).

If you add more modules to the project at a later date, remember to add appropriate `Module=` lines to the project file.

Source Files

A MapBasic Source file has the extension *.mb.

The source files do contain the actual code.

In general, when you compile a source file MapBasic will try to create an executable application file (*.mbx). If the source file is part of a larger project, MapBasic builds a MapBasic Object file (*.mbo) instead of an executable file (*.mbx).

MapBasic will build an object file whenever:

- the `Main` procedure is missing from the module that you are compiling. (Please note: an application can only have one `Main` procedure, so only one object file can hold this procedure; all others won't.)

- the module contains calls to functions and procedures that are not in the file. Calling a function or sub procedure located in another module is known as an external reference.

***For example*:**

1. In the file *ProjectUI.mb* some User Interface is created: a menu and an About box (with an Exit button to halt/remove the application). This module contains the `Main` procedure,

but it also calls two procedures - `ModuleOne` and `ModuleTwo` – which are located in other modules.

2. The module *ModuleOne* calls the function **LongDate** which is located in the module *Functions*.

Include files

Within a project you want to be able to i) call functions and procedures from other modules, and ii) to share variables with other modules.

To make all the functions, procedures and global variables known to all the modules in the project it is best to place all the relevant statements in one include file (also called a definitions file) and use the **Include** statement to incorporate these definitions in each module.

When MapBasic is compiling a program file and encounters an **Include** statement, the entire contents of the included file are inserted into the program file. The file specified by an **Include** statement should be a text file, containing only legitimate MapBasic statements.

For example:

1. Each of the modules in our sample project does contain the line: `Include "Project.def"`

2. The file *Project.def* does contain several statements (**Include**, **Declare Sub**, **Declare Function**, **Define** and **Global**).

Compiling and Linking a Project

To create an executable application file from your project you have to follow these steps:

1. Compile each module that is used in the project.

2. Link your application. MapBasic reads the object (*.mbo) files listed in the project file. If there are no link errors, MapBasic builds an executable (*.mbx) file. If there are link errors, MapBasic displays an error message.

(The object files created by the MapBasic compiler cannot be linked using any other linker, such as a C-language linker. Only the MapBasic linker can link MapBasic object modules.)

Using Notepad++?

If you are using Notepad++ to develop your MapBasic applications, you can configure your text editor to compile all object files, link the project file and run the resulting application, all with one mouse click or keystroke.

For instructions, see:
http://www.twiav.nl/php/mapbasic2.php#link