mikroPascal to mikroBasic Convertor

2015-02-08

Content

1 Introduction

This tool is a simple mikroPascal to mikroBasic source code convertor. The tool can convert complete **mP units** to **mB modules** and complete **mP programs** to **mB programs**.
Additionally the possibility exists to convert loose definitions and/or loose code (not embedded in a program or unit environment).
There are some mP constructs that can not be translated to mB, or some loose code that can not be translated correctly to mB, see Considerations and Still to do manually.

2 Usage

The tool is a stand alone tool, it can not be started from within the compiler's IDE.
There are 2 possible usages of the tool: interactive mode and batch mode.

2.1 Interactive mode

After starting the tool the user must take following steps:

- Copy some Pascal code to the clipboard (select the code, press control C),
- Paste that pascal code in the "Pascal" field of the tool (press Control V in the Pascal field),
- Press ">> TransLate >>" , and
- Copy the translated code in the "Basic" field of the tool to the clipboard (press control A, Control C in the "Basic" field,
- Past the clipboard contents to where the translated code is to be used (select the destination and press Control V).

2.2 Batch mode

Here the tool is called in a batch file. The command line(s) in that file contains (besides the name of the tool executable) 2 parameters separated by a comma: the mP source file and the mB destination file.
If the parameters contain spaces they will have to be enclosed in double quotes. Both parameters and the name of the tool executable can contain a path (directory info), see example below.

Example of file “BatchConversion.bat”:

```
@echo off
@echo Running Convertor Test
Pascal_to_Basic_Convertor.exe TestBasics.mpas,_TestBasics.mbas
Pascal_to_Basic_Convertor.exe Test.mpas, _Test.mbas
Pascal_to_Basic_Convertor.exe TestBasics2.mpas, _TestBasics2.mbas
Pascal_to_Basic_Convertor.exe ..\..\Test_IFDEF.mpas, ..\_Test_IFDEF.mbas
```

When executing above batchfile (by double clicking it) 4 conversions are done subsequently.

3 Considerations

- Currently the tool can only convert mP for PIC source code to mB for PIC source code, but the tool can most probably also be used with other mP source code.

- The mP code submitted to the tool is assumed to be compilable with an mP compiler. If non compilable code is submitted then also the generated mB code will be wrong. The tool does no syntax checking (that is the task of the compilers).

- The tool can not convert the mP “**with**” statements. There is no equivalent in mB for them. The user has to do the necessary additions in the mB code to cope with this, or avoid the “with” statement in the mP code.
  An extra comment is added in the basic code to signal a “with” statement. See section Convert the “With” statement.

- Some identifiers (e.g. names of variables) that are valid in mP are not in mB. They generate an “invalid identifier” error when trying to compile the converted code. See section Rename “invalid” identifiers.

- Loose code that is preceded by definitions (types, constants, variables etc.) has to be embedded in “begin” “end” statements, otherwise the tool will treat the code as if it is also a definition as the one preceding it. See section Converting “Loose” code.

- The tool does not take note of ***compiler directives***, it only translates them into their mB syntax. This means that (to the tool) the mP code looks like it there are no directives present at all, the tool can not know e.g. which values of the compiler directive are mutual exclusive. So, make sure the mP code also looks OK if the compiler directives are not there.

  There are a few exceptions to this principle, it is allowed to do the following with compiler directives:

  - Make double routine definitions, e.g.
    ```
     {$IFDEF purePascal}
    function uGLCD_Char_Ptr(ch: byte): ^const byte;
     {$ELSE}
    ```

```
procedure uGLCD_Char_Ptr(ch: byte);
{$ENDIF}
begin
  if FontFixed then
```

- Make double record definitions, e.g.

```
{$ifdef abd}
type xxx = record
{$else}
type fff = record
{$endif}
  a,b,c: byte;
end;
```

In all other conditions, where the presence of the compiler directive doesn't matter for the validity of the code (= translatable from the tool's point of view), compiler directives can be of course used freely, e.g.:

```
{$IFDEF abc}
var aa: byte;
{$ELSE}
var aa: dword;
{$ENDIF}
```

4 Still to do manually

4.1 Convert the "With" statement

The translation of the "with" statement existing in mP is not done during the conversion. In stead a comment is added: **`'?? <-- With statement not supported in mB`**

The user himself must manually add the record variable name to each of its members.

Example: the result after conversion can e.g. be:

```
' with RecordName do '?? <-- With statement not supported in mB
 'begin
   Field1 = Value1
   Field2 = Value2
 'end
```

This has to be changed by the user as follows:

```
   RecordName.Field1 = Value1
   RecordName.Field2 = Value2
```

Note:

Avoiding the non convertibility with the "with" statement to mBasic can also be done by using a pointer to the record in stead of using the "with" statement.

Example in mP:

```
type Txxx = record;
     x1: byte;
     x2: byte;
end;

procedure xxx(var yyy: Txxx);
var Ptr : ^Txxx; // <-- creation of pointer to the record
begin
  Ptr := @yyy;    // <-- initialisation of pointer
  Ptr^.x1 := 2;   // <-- usage of pointer
  Ptr^.X2 := 3;   // <-- usage of pointer
end;
```

in stead of (using the with statement)

```
procedure xxx(var yyy: Txxx);
begin
  with yyy do
  begin
    x1 := 2;
    X2 := 3;
  end;
end;
```

Gives in mB:

```
structure Txxx
dim x1 as byte
    x2 as byte
end structure

sub procedure xxx(dim byref yyy as Txxx)
dim
  Ptr = ^Txxx
'begin sub
  Ptr = @yyy
  Ptr^.x1 = 2
  Ptr^.X2 = 3
end sub
```

in stead of (the record members will not be found in mB)

```
sub procedure xxx(dim byref yyy as Txxx)
'begin sub
'   with yyy do '?? <-- With statement not supported in mB
  'begin
    x1 = 2
    X2 = 3
  'end
end sub
```

4.2 Rename “invalid” identifiers

Some identifiers (e.g. names of variables) that are valid in mP are not in mB. They generate an “invalid identifier” error when trying to compile the converted code. They have to be renamed manually in the generated mB source. An easy manner is just adding an underscore to the invalid name.

Known cases of invalid identifiers are: System, Command, Dir, Asc, Stop (while "start" is accepted as identifier), FileAttr, Sqr, Line, Int, Base, Version, Clear, CurDir and WeekDay.
Those identifier are seen by the compiler as reserved words, but they are not present in the help about reserved words.

4.3 Converting “Loose” code

Loose code (= code not embedded in a unit/module or in a program layout) that is preceded by definitions (types, constants, variables etc.) has to be embedded in “begin” “end” statements, otherwise the tool will treat the code as if it is also a definition as the one preceding it. If no definitions are preceding the mP source code then the additional “begin” “end” block statements are not needed.

Example (mP):
```
type a = byte; // <----------- some definition

for I := 0 to 10 do
begin
  // do some things
end;
```

mB output:
```
typedef a as byte ' <------------ some definition

typedef for I : as 0 to 10 do ' <----- messed up translation
'begin
  ' do some things
'end                          ' <----- messed up translation
```

mP code to be changed in: (adding “begin” and “end” around the code)
```
type a = byte;

begin // <--------------------- additionally
for I := 0 to 10 do
begin
  // do some things
end;
end; // <---------------------- additionally
```

correct mB output:
```
typedef a as byte

'begin <------------------------ can be removed
for I = 0 to 10
```

```
'begin
  ' do some things
next I
'end <--------------------------- can be removed
```

The user can remove the (commented out) “begin””end” lines from the generated mB code.

5 Acknowledgements

Thanks to Janni for the help with the testing of the tool, the problem reports and the improvement suggestions.

[end of document]