mikroPascal to mikroBasic Convertor

2013-04-21

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.

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).

3 Considerations

- Currently the tool can only convert mP for PIC source code to mP for PIC source code.
- The mP code submitted to the tool is assumed to be compilable in 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 it. The user has to make the necessary changes to cope with this.
- The tool can not convert “downto” in an mP “for” statement. The user himself has to insert a negative step.
- 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 as if it is also a definition as the one preceding it.

4 Still to do manually

4.1 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 add 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
```

4.2 Downto (in for statement)

The “downto” part of the “for” statement is not known in mB, the user himself must make the necessary change: replace the “downto” by “to” and add a negative step to the for statement.

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

```
for I = 10 downto 0
'begin
  ' do some things
next I
```

This has to be changed by the user as follows:

```
for I = 10 to 0 step - 1 '<--------- here
'begin
  ' do some things
next I
```

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

[end of document]