Robot Maker
For Innovative Leisure and For Education

POB-BASIC Syntax

Table of contents

Document management

Filename	POB-BASIC US Syntax
Creation date	29.03.2006
Author	

POB-Technology contacts

Adress	POB-TECHNOLOGY 4, rue Nicéphore Niepce 69 680 CHASSIEU, FRANCE
Mail	contact@pob-technology.com
Phone	+33 (0)4 72 43 02 36
Fax	+33 (0)4 78 58 04 92

1   Introduction

The Basic syntax of the POB-Basic and all of documentations come from PureBasic: http://www.purebasic.com/

2   Syntaxes

Comments are marked by ' ;'
> *; A comment*

All functions must be followed by a '('
> *aFunction()*

All labels must be followed by ' :'
> *aLabel :*

Hexadecimal number must begin by '0x'
> *a.l = 0x123ABC*

3   Basic types

Basic type must be specified at the first use. Here is the list of the native supported types and a brief description:

Name	Extension	Memory	Range
Byte	.b	1 octet	-128 to +127
Word	.w	2 octets	-32768 to +32767
Long	.l	4 octets	-2147483648 to +2147483647
Float	.f	4 octets	Real
String	.s	Length of the string+1	String

Examples:

a.b ; Set a variable as Byte
aWord.w ; Set a variable as Word

4 Keywords

4.1 Global-Const-Static

Syntax

    Global <variable>

Description

    Global allows the variables to be used as global, ie: they can be accessed inside any basic file and in any procedure.

Syntax

    Const <variable>

Description

    Const allows to declare variables a constant.

  ; Variable 'FASTGRAPHIC' is constant and global for all basic file.
Global Const FASTGRAPHIC.l = 1

Syntax

    Static <variable>

Description

    Static allows the variables to be used only in the current basic file. They can't be accessed outside the file. All the procedures of the current file can access the static variables.

; Example: we are in "test.bas":

; Variable 'FORMFOUND' can be accessed in all procedure of "test.bas".
Static FORMFOUND.l

4.2 Break-Continue

Syntax

    Break

Description

    Break allows to exit at any moment of the following loops: Repeat, For and While.

Syntax

    Continue

Description

    Continue allows to skip the end of the current iteration in the following loops: Repeat, For et While.

4.3 For-Next

Syntax

> **For** <variable> = <expression1> **To** <expression2> [**Step** <constant>]
>> ...
> **Next**

Description

The **For** : **Next** function is used to cause a loop within a program within given parameters. At each loop the <variable> is increased by a factor of 1, (or of the "Step value" if a **Step** value is specified), when the <variable> value equals the <expression2> loop stop.

With the Break command its possible to exit the **For** : **Next** loop at any moment, with the Continue command the end of the current iteration can be skipped.

4.4 If-Else-Endif

Syntax

> **If** <expression>
>> ...
> [**ElseIf** <expression>]
>> ...
> [**Else**]
>> ...
> **EndIf**

Description

The **If** structure is used to achieve tests, and/or change the programmes direction, if the test is true or false.
ElseIf can be used to do any number of additional test if the previous one was not true.
The **Else** optional command is used to execute a part of code, if all previous tests were false. Any number of **If** structures can be nested together.

4.5 Repeat-Until

Syntax

> **Repeat**
>> ...
> **Until** <expression>

Description

This function will loop until the <expression> becomes true. Any number can be repeated.

4.6 Select-EndSelect

Syntax

Select <expression1>
    Case <expression2>
        ...
    [**Case** <expression3>]
        ...
    [**Default**]
        ...
EndSelect

Description

    Select allows a quick choice. The program will execute the <expression1> and keep its value in memory. It will compare this value to all of the "**Case** <expression> values" and if true it will execute the corresponding code and quit the **Select** structure.

    If none of the **Case** values are true, then the **Default** code, (if specified), will be executed.

Bob.I = 2

Select Bob
    Case 1
        …
    Case 2
        ...
    Default :
        …
EndSelect

4.7 While-Wend

Syntax

    While <expression>
        ...
    Wend

Description

    Wend will loop until the <expression> becomes false. A good point to keep in mind with a **While** test is that if the first test is false, then the program will never enter the loop and will skip this part. A Repeat loop is executed at least once, (as the test is performed after each loop).

4.8 Goto

Syntax

Goto <label>

Description

This command is used to transfer the program directly to the labels position. Be cautious when using this function, as incorrect use could cause a program to crash...

4.9 End

Syntax

End

Description

Ends the program execution correctly.

5

6 Structure and Array

Syntax

   Dim nom.<type>(<expression>, [<expression>], ...)

Description

   Dim is used to 'size' the new arrays. An array can be of any types, including Structure structured, and user defined types.

Warning: Array's elements are all of same type.

```
Dim AnArrayOfInteger.l(9)      ; Create an integer array size of 10

AnArrayOfInteger(0) = 123      ; At 0, set integer '123'
AnArrayOfInteger(2) = 12345  ; At 2, set integer '12345'
```

Syntax

   Structure <nom>
                   ...
   EndStructure

Description

   Structure is useful to define user type. It is very useful as you can group into the same object the informations which are common. Structures fields are accessed with the **** option. Structures can be nested.

```
; Declare a structure 'TestStruct' with 2 members: 'AnInteger' and 'AFloat'
Structure TestStruct
      AnInteger.l
      AFloat.f
EndStructure

MyStruct.TestStruct              ; Create a new structure with name 'MyStruct'

MyStruct\AnInteger = 12345    ; Set in 'AnInteger' of 'MyStruct' the integer '12345'
MyStruct\AFloat = 1.23456      ; Set in 'AFloat' of 'MyStruct' the real '1.23456'
```

7 Procedures

Syntax

Procedure[.<type>] nom(<variable1>[,<variable2>,...])
...
[**ProcedureReturn** valeur]
EndProcedure

Description

A **Procedure** is a part of code independent from the main code which can have any parameters and it's own variables. At each call of the procedure the variables inside will start with a value of 0 (null).

A procedure can return a result (value) if necessary. You have to set the type after **Procedure** and use the **ProcedureReturn** keyword at any moment inside the procedure. A call of **ProcedureReturn** exists immediately the procedure, even when its called inside a loop.

Remarks

Basic program for POBEYE must contain one procedure « **ProcedureMain** ». This procedure is call when the POBEYE is launch. After fist call of "ProcedureMain", you can call extern procedure.

```
; The main procedure for a POB-EYE program, this procedure is call at first time:
Procedure ProcedureMain()
    …
    AnotherProcedure()              ; call to 'AnotherProcedure' in the main function.
    …
EndProcedure
```

Syntax

Declare nom

Description

Sometimes a procedure needs to call another procedure which isn't declared before its definition or in another file. This is annoying because the compiler will complain 'Procedure <name> not found'. **Declare** can help in this particular case by declaring only the header of the procedure.

```
; File Main.bas :

; Declare 'AnotherProcedure' as extern
Declare  AnotherProcedure

; The main procedure for a POB-EYE basic program
Procedure ProcedureMain()
    …
    AnotherProcedure()              ; call to a procedure of name 'AnotherProcedure'
    …
EndProcedure



; File  Function.bas

; The 'AnotherProcedure' is defined in the file 'Function.bas'.
; If you define procedure in multiple file, don't forget in POB-basic software to add the basic file.
```

Procedure AnotherProcedure()

        …

EndProcedure

POB-Technology contacts

POB-TECHNOLOGY
4, rue Nicéphore Niepce
69 680 CHASSIEU
FRANCE

Web: www.pob-technology.com

Mail address: contact@pob-technology.com

Phone: +33 (0)4 72 43 02 36
Fax: +33 (0)4 78 58 04 92