PE-BASIC REV 0.17

B.A.S.I.C. Interpreter for the Parallax Propeller Microcontroller

PE-Basic 0.17

PE-Basic 0.17

Overview:

PEBasic is an interpreted BASIC (Beginners All-purpose Symbolic Instruction Code) language for the Parallax Propeller microcontroller.

If you have every used any of the "home computers" of the 1980's you will be familiar with the language as it was built-in to most computers of the time. (Timex Sinclair, C64, Atari 400/800, Vic 20, TI 99/4A, etc).

The program is written using line numbers to indicate the order of execution. It is customary to number the lines in increments of 10 so that additional lines may be inserted later.

Commands can be entered as part of a program with a line number, or as a direct command without a line number. Direct commands are executed immediately.

Here is a short program that prints the numbers from 1 to 10.

```
10 FOR a=1 TO 10
20 PRINT a
30 NEXT a
```

By entering the direct command RUN the program will execute.

PE-Basic 0.17

Variables:

Variable names must start with a letter, may contain letters and numbers, may be up to 8 characters long.

FOR..NEXT variables must be a single letter.

Variables are 32-bit signed integers able to hold integer values from -2,147,483,648 to +2,147,483,647.

Upper and lower case are the same. The variable "value", "Value" and "VALUE" are all the same variable.

You cannot use a command or other reserved word as a variable name.

Single letter variable names execute faster.

Up to 100 multi-letter variable names may be created.

The following are valid variable names:

value
value5
value23

The following are NOT valid variable names:

5value - may not start with a number
BallXPosition - too long (more than 8 characters long)
value_5 - Contains an invalid character
free - "free" is a reserved word

PE-Basic 0.17

Registers:

`DIRA`	Pin direction 0=INPUT; 1=OUTPUT	write-only
`OUTA`	Pin outputs 0=LOW; 1=HIGH	write-only
`INA`	Pin inputs 0=LOW; 1=HIGH	read-only
`CNT`	System counter	read-only
`CTRA,CTRB`	Counter mode	write-only
`FRQA,FRQB`	Counter frequency	write-only
`PHSA,PHSB`	Counter phase	read/write
`VCFG,VSCL`	Sets video generator	write-only
`INKEY`	Returns value of key pressed	read-only
`VARS`	Address of variables	read-only
`FREE`	Returns number of free program bytes	read-only
`CHARS`	Address of character bitmaps	read-only - NTSC-only
`SCREEN`	Returns address of screen memory	read-only - NTSC-only

PE-Basic 0.17

Functions:

`ABS`(expr)	Returns the absolute value of expr
`RND`(expr)	Returns a random number from 0 to expr -1
`PEEK`(expr)	Returns byte(8-bit) value in memory at expr
`PEEKW`(expr)	Returns word(16-bit) value in memory at expr
`PEEKL`(expr)	Returns long(32-bit) value in memory at expr
`PIN`(expr)	Returns value of pin expr
`PIN`(expr_msb..expr_lsb)	Returns value of pin group

PE-Basic 0.17

Pin I/O:

`INPUT`   Make pin(s) inputs.

`OUTPUT`   Make pin(s) outputs.

`HIGH`   Make pin(s) output and high (3.3V)

`LOW`   Make pin(s) output and low (0V)

`PIN`   Sets a pin or pin group to a specific value

For all commands that operate on hardware pins you can specify a range of pins by using MSB..LSB.

For example to make pin 23 high use: `HIGH 23`

To make pins 24 thru 26 high use: `HIGH 24..26`

!!! NOTE if the MSB value is less than the LSB value, the bits will be reversed, this is the same as the spin language !!!

PE-Basic 0.17

Operators:

Order of precedence:

Parenthesis ( )
UNARY +, UNARY -, !, ABS, RND, PEEKB, PEEKW, PEEKL, PIN, ..
SHL, SHR, ROL, ROR, SAR, REV
&
|, ^
*, /, //
+, -
=, <, >, <=, >=, <>
NOT
AND
OR

Description:

SHL	Shift left	2 SHL 3 gives 16
SHR	Shift right	16 SHR 3 gives 2
ROL	Rotate left	
ROR	Rotate right	
SAR	Shift Right Arithmetic	
REV	Reverse bits	4 REV 3 gives 1
!	Bitwise NOT	!0 gives -1
&	Bitwise AND	6 & 3 gives 2
|	Bitwise OR	6 | 1 gives 7
^	Bitwise XOR	6 ^ 4 gives 2
*	Multiply	
/	Divide	
//	Modulus	
+	Addition	
-	Subtraction	
=	Logical is equal to	1 = 2 gives 0; 2 = 2 gives -1
<	Logical is less than	
>	Logical is greater than	
<=	Logical is less than or equal to	
>=	Logical is greater than or equal to	
<>	Logical is not equal to	
NOT	Logical NOT	
AND	Logical AND	
OR	Logical OR	

Notes:
Logical operators take zero as false and non-zero as true.
Logical operators return zero as false and -1 as true.

&, |, ^ are bitwise (AND,OR,XOR); "AND" and "OR" are logical AND and OR.
4 | 1 = 5
4 OR 1 = -1

x..y returns (x + y*256 + 11141120) 11141120 = $AA0000 and is just a unique number which means (this is a .. result)

x..y - 0..0 = x + y*256

PE-Basic 0.17

Commands:

BCOLOR   BCOLOR {expression}
               BCOLOR 4
      Sets the background color (see COLOR, FCOLOR)
               0 = BLACK
               1 = MAGENTA
               2 = RED
               3 = YELLOW
               4 = GREEN
               5 = CYAN
               6 = BLUE
               7 = DARK GREY
               8 = LIGHT GREY
               9 = BRIGHT MAGENTA
               10 = BRIGHT RED
               11 = BRIGHT YELLOW
               12 = BRIGHT GREEN
               13 = BRIGHT CYAN
               14 = BRIGHT BLUE
               15 = WHITE

CLS   CLS
               CLS
      Clears the screen to the currently set color

COLOR   COLOR {expression}
               COLOR 4+15*16 ' White on Green
      Sets both background and foreground colors with one value (see BCOLOR, FCOLOR)
      Color = background + foreground * 16

CONT   CONT {optional expression}
               CONT
      Continue program after ESC is pressed

DATA   DATA expression, expression, expression
               DATA 0,1,2,4,8,16,32
      Define data to be read with READ (see READ, RESTORE)

DEBUG   DEBUG
               DEBUG
      Shows line #'s as program runs

DISPLAY   DISPLAY {expression}
               DISPLAY 42 ' prints a "*"
      Prints ascii character. May use multiple paramters.
      Value 10 moves to next line and moves back to starting position (for multi line displays)

DUMP   DUMP
               DUMP
      Shows program bytes, press a key to stop

END   END
               END
      Stops program and returns to command prompt

PE-Basic 0.17

FCOLOR FCOLOR {expression}
     FCOLOR 7
   Sets the foreground color (see COLOR, BCOLOR)
     0 = BLACK
     1 = MAGENTA
     2 = RED
     3 = YELLOW
     4 = GREEN
     5 = CYAN
     6 = BLUE
     7 = DARK GREY
     8 = LIGHT GREY
     9 = BRIGHT MAGENTA
     10 = BRIGHT RED
     11 = BRIGHT YELLOW
     12 = BRIGHT GREEN
     13 = BRIGHT CYAN
     14 = BRIGHT BLUE
     15 = WHITE

FOR   FOR {single letter var} = {start value} TO {limit value} [STEP {step value}]
     FOR A = 1 TO 10
   Creates a program loop

GOSUB GOSUB {expression}
     GOSUB 1000
   Go to subroutine (see RETURN)

GOTO   GOTO {expression}
     GOTO 100
   Jumps to specified line number

HIGH   HIGH {expression} or HIGH {expression..expression}
     HIGH 23
     HIGH 23..26
   Make pin(s) an output and high

IF   IF {condition expression} THEN commands [ELSE commands]
     IF A = B THEN 1000
     IF A <> B THEN c=1000:d=1000 ELSE e=1000
   If the condition is true, execute commands following THEN, otherwise skip to next line

INPUT   INPUT {expression} or INPUT {expression..expression}
     INPUT 23
     INPUT 23..26
   Make pin(s) an input

LET   LET {var} = {expression}
     LET A=A*10
     LET A=PIN 27..24
   Assigns a value to a variable. (LET is optional)

PE-Basic 0.17

LIST   LIST {optional expression}
                    LIST
                    LIST 100
         Show program listing (Press a key to stop)

LOAD   LOAD {optional expression}
                    LOAD
                    LOAD 1
         Retrieves program from EEPROM, if 64K eeprom can use LOAD [1-4]

LOCATE   LOCATE {expression},{expression}
                    LOCATE 5, 10
         Sets print location to x,y

LOW   LOW {expression} or LOW {expression..expression}
                    LOW 23
                    LOW 23..26
         Make pin(s) an output and low

NEW   NEW
                    NEW
         Clears program and displays version info

NEXT   NEXT {single letter variable}
         Adjusts value and loops back to FOR line

NODEBUG   NODEBUG
         Does NOT show line #'s as it runs (see DEBUG)

OUTPUT   OUTPUT {expression} or OUTPUT {expression..expression}
                    OUTPUT 23
                    OUTPUT 23..26
         Makes pin(s) an output

PAUSE   PAUSE {expression}
                    PAUSE 1000
         Pauses for milliseconds

PIN   PIN {expression},{expression} or PIN {expression}..{expression},{expression}
                    PIN 23,1
                    PIN 27..24,15
         Sets pin output state. NOTE: DOES NOT SET PIN TO OUTPUT MODE

POKE   POKE {expression},{expression}
                    POKE a,100
         Changes a byte of program memory

POKEW   POKEW {expression},{expression}
                    POKEW a,1000
         Changes a word of program memory

POKEL   POKEL {expression},{expression}
                    POKEL a,100000
         Changes a long of program memory (RAM, not EEPROM)

PE-Basic 0.17

PRINT   PRINT {expression} or PRINT "TEXT"
                PRINT a
                PRINT "The value is ";a
        Prints to the screen.

QUIT   QUIT
                QUIT
        Ends the TELNET session. (Telnet version only)

READ   READ {variable} [,{variable},etc]
                READ a,b,c
        Reads data from the DATA lines

REM   REM {any characters} may use apostrophe in place of REM
                REM This is a comment
                dirx = 1 ' set direction to 1
        Comment

RESTORE   RESTORE {optional expression}
                RESTORE 1000
        Set program line number that READ will start reading data from

RETURN   RETURN
                RETURN
        Return from subroutine

RUN   RUN {optional expression}
                RUN
                RUN 1000
        Runs program

SAVE   SAVE {optional expression}
                SAVE
                SAVE 1
        Saves program to EEPROM, if 64K eeprom can use SAVE [1-4]

PE-Basic 0.17

NOTES:

Single letter variable names are faster than multi-letter variable names

FOR...NEXT is faster than GOTO
  GOTO needs to scan from the beginning to find the line # requested

FOR does NOT have to be the first command on a line.
  10 CLS: FOR a=1 TO 10:PRINT a:NEXT a

PE-Basic 0.17

EXAMPLE PROGRAMS:

```
1 REM ----------------
2 REM Guess my number
3 REM ----------------
10 CLS
20 a=RND(100)+1
30 PRINT "Guess my number (1 to 100):";
40 b=0
50 c=INKEY:IF c=0 THEN 50
60 IF c=13 THEN 120
70 IF c=8 THEN DISPLAY 8,32,8:b=b/10:GOTO 50
80 c=c-48:IF c<0 OR c>9 THEN 50
90 PRINT c;
100 b=b*10+c
110 GOTO 50
120 PRINT
130 IF b>a THEN PRINT b;" is too high..."
140 IF b<a THEN PRINT b;" is too low..."
150 IF b<>a THEN 30
160 PRINT b;" is the correct answer!!!"
```

```
1 REM --------------------
2 REM Hardware Counter Demo
3 REM --------------------
10 OUTPUT 16..17 ' Make pins outputs
20 FRQA=200 ' 80MHz * 200 / (2^32) = 3.72Hz
30 CTRA=5 SHL 26 + 16 SHL 9 + 17 ' Mode=5,bpin=16,apin=17
```

```
1 REM ---------
2 REM I/O Demo
3 REM ---------
10 LOW 23..16
20 FOR a=16 TO 23
30 PIN a,PIN(a) ^ 1
40 PAUSE 50
50 NEXT a
60 GOTO 20
```