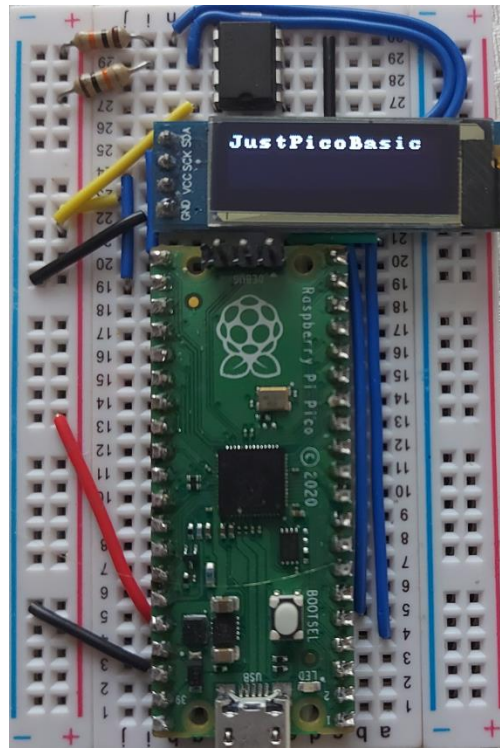


## #JustPicoBasic manual

This manual is common for both Windows- and PICO-based BASIC interpreter versions.

I believe that the best way to see how it works is to follow the provided program examples:

<https://github.com/bgolab/JustBasic/tree/main/examples>



### Terminal

JustPicoBasic was tested w/ Putty and TeraTerm. Copy & Paste is supported.

```
COM10 - Tera Term VT
File Edit Setup Control Window Help

JustBasic v1.0B20
(C) 2021 bg
HW: RP2040
Running at 125000000Hz

LIMITs: SRC/LINE:8000/160, DIM/DATA:500, NAMES/NAMELENGTH:100/8, IF/FOR/WHILE:50/50/50
CLI: [L]oad, [S]ave, [C]ode, [R]un, [N]ew, [U]ser, [E]dit, @[ ], [B]ye, [H]elp, [T]ech, T0, T1, T2, ?
CMDs: print, input, inkey, cls, data, read, restore, if, then, else, endif, for, to, step, next, while, endwhile, goto, gosub,
return, rem, peek, poke, pause, rnd, abs, sin, cos, exp, log, sqr, sgn, hex$, str$, chr$, left$, mid$, right$, len, val,
int, fix, gettick, dim, pmode, dwrite, awrite, dread, aread, end, ,, ;, +, -, and, or, not, *, /, %, (, ), <, <=, >, >=, =,
<>, plot, draw, circle, lprint, lref, lcls, sm,

#c
TSENSOR=20
pmode 100, TSENSOR
for k=1 to 5
    tlk=aread(100)
    print tlk/1000,".",tlk%1000,"C"
    pause 500
next k
end

#r

Press ESC to break!

27.258C
26.790C
26.790C
26.790C
26.790C
```

### **Code formatting**

You can freely format the code – can put many commands in single line, etc. Capital and small letters accepted for keyword. Names recognizes small and capital letters.

#### Example

```
print "Hi" for k=1 to 10 print k next k
end
```

### **Program structure**

Command 'end' should follow the last line of the code. Subroutines shall follow the 'end' command otherwise they will be wrongly interpreted.

```
code
end
subroutines
```

#### Example

```
print "Hi!"
gosub doitagain
end
doitagain:
    print "Hi!"
return
end
```

### **System modes**

There are two systems modes using different prompts: user ('>' prompt) and enhanced ('#' prompt)

**U-MODE** (aka single-line mode) can be entered by issuing: 'U' command; useful to run single line programs, for example:  
m=3 data 2, 3, 4 sum=0 for i=1 to m read a sum=sum+a next i print "s=", sum, ", av= ", sum/m end  
E – enter E-mode

**E-MODE** (aka enhanced mode); can be entered by issuing: 'E' command; this mode supports many system commands:

? - shows quick info about the VM (version, list of available commands)

H – help – shows some program examples

U – enter U-mode

C – code – list current program code (use: 'SM LN 0' to see program using 'c' without line numbers)

R – run program

N – new – clear VM memory and code

B – bye – reboots the PICO in USB disk mode, in Windows version exists the VM

L – load / save program from / to persistent memory (auto.bas) – EEPROM 24C64 supported

### **Built-in editor**

Every command is appended to the existing code (at the end).

'C' shows code and internal line numbers used for @N commands.

@N (e.g. @3) – removes Nth line of code (use C to see the line numbers)

@N cmd (e.g. @4 PRINT 5 – inserts 'PRINT 5' before line 4) inserts the new code line following the @N before the Nth line

### **Built-in program flow tracing**

T0 – disables tracing

T1 – enables stepping mode (can step program, and see next command, variables)

T2 – normal run with tracing (full run rate with tracing enabled to track program flow)

Example: enables stepping mode, runs program, can select [S] to step or [G] to go

T1 or T2 – to enable particular tracing mode  
R – to run program end enter particular tracing mode

## **The programming language**

### **MISC**

-ESC - break program when loading or running;  
-SM ENTITY 1|0 - system mode command to set system configuration  
  SM ESC 1|0 (default=enabled) – enable / disable ESC key check (disable to boost performance – temporary fix)  
  SM OLED 1|0 (default=disabled) – enable OLED hw (cannot be disabled now)  
  SM LN 1|0 (default=enabled) – enable / disable line numbering in 'c' command  
-REM – comment

### Example

REM Initial procedure

-CLS – clear screen

### **VARIABLES, EXPRESSIONS**

-var types (suffix matters): INT (no suffix, name=expr), FLOAT (suffix '#', name#=expr), STRING (suffix '\$', name\$=expr)  
-variable name: up to 8chars letter&digits starting w/ a letter(digits, '#', '\$', '\_', ': ' accepted),  
-var initialization: 1<sup>st</sup>-reference creates var(value=0); any variable can be assigned an expressions: var=expr;  
-expressions: INT/FLOAT, +, -, \*, /, %, (, ), vars;  
-array: 1-dimension; INT/FLOAT supported; **STRING arrays NOT supported**; DIM name(size); name(item)=expr  
-strings supports only '+' in expressions + string functions and variables

### Example: int, float, string variables

hi5=2  
w#=2.5  
name\$="John"

### Example: int, float arrays

DIM a(3)  
a(0)=3  
DIM(b#(3)  
b#(0)=2.5

### Example: complex expressions

sy=2\*abs(-15) + a\*20  
v#=fexpr; b#=a#-2\*(2+3)+abs(-1.0)+aa#(2);

### Example

v\$="a"+a\$+left\$(str\$(13),1);  
PRINT sexpr; VAL("-1")->-1

### **LOOPS & PROGRAM FLOW**

-label name: up to 8chars letter & digits starting w/ a letter( '\_' accepted),  
-loop/if nesting supported  
-FOR var=expr TO expr [STEP expr] [] NEXT var; if STEP[default=1] is negative var decreases; FLOAT supported

### Example

FOR i = 5 TO 1 STEP -1 NEXT I END

-WHILE expr1 op expr2 [code] ENDWHILE; *FLOAT supported;*

Example:

a=0 while a<5 print a a=a+1 endwhile end

-GOTO – label (aka name with colon); can be located everywhere (before and after the GOTO): Label: [code] GOTO label

Example

k=1

again:

print k k=k+1

if k<5 then goto again: endif

end

-GOSUB – label can be located everywhere (before and after the GOTO): Label: [code] RETURN GOSUB label

Example

gosub task0

end

task0: print "done" return

-IF conditions THEN [code] [ELSE] [code] ENDIF; *FLOAT supported; AND/OR/NOT supported*

Example

if a>1 and b#>3.4 then print "good" else print "bad" endif

-END last instruction (GOSUB labels can be located behind the END)

**INPUT, OUTPUT, DATA**

-PRINT expr, sexpr,...( separate items by '); ' at the end to skip NEW LINE;

Example

PRINT "How:", 6/3 ;

PRINT 1 (prints 1 w/ NEW LINE); PRINT 1; (prints 1 w/o NEW LINE)

-INPUT – assigns int/float/string values to var or array element: INPUT var, array\_element, a\$, ...;

Example

INPUT a(2), d#, name\$

print a(2), d#, name\$

-DATA expr, fexpr, str; READ a, b#, d\$; RESTORE clears data pointer; INT/FLOAT/STR supported

Example

DATA 1.5, 2\*a; READ v, v#, v(), v#();

**BUILT-IN FUNCTIONS**

-LEFT\$/RIGHT\$/MID\$, LEN/VAL(sexpr)

-HEX\$/STR\$/CHR\$(expr);

Example: string functions

k\$=LEFT\$("abcdefgh", 3) + "1234"

PRINT VAL("-1234")+1

```
i=1234567
i$=MID$(STR$(i), 2, 3)
PRINT i$
PRINT MID$(STR$(i), 2, 3)
```

#### Example

```
PRINT HEX$(NOT(0x0F))
a=65
d$=chr$(a)
```

-SIN/COS/SQR/EXP/LOG;

#### Example

```
PRINT "FLOAT FUNC"
PRINT "SQR: ", SQR(5)
PRINT "EXP: ", EXP(1)
PRINT "LOG: ", LOG(2.718)
PRINT "SIN: ", SIN(30*3.14/180)
PRINT "COS: ", COS(60*3.14/180)
```

-RND(hw-based random generator with von Neuman extractor-whitenizer)/SGN/ABS

#### Example

```
PRINT "RND: ", RND(1000)
```

-GETTICK() – returns tick number

-PAUSE msec

#### Example

```
PAUSE 2*500
a=gettick()
```

-INKEY() – return current key (if pressed), otherwise 0; non-blocking (no-wating)

-INT/FIX (as in QBASIC)

#### Examples

```
a=INT(1.1)
b=INT(-1.1)
c=FIX(1.9)
d=FIX(-1.9)
PRINT a, " ", b, " ", c, " ", d (1, -2, 1, -1)
```

-AND, OR, NOT

#### Example

```
PRINT "BIT OPS"
PRINT AND(0x03,0x0F)
PRINT OR(0x01,0x02)
PRINT HEX$(NOT(0x0F))
```

### **PICO HARDWARE SUPPORT**

-PEEK(addr) – hex supported

*-POKE addr, value – hex supported*

Example

```
REM SYSTICK
SYSTCSR=0xe000e010
SYSTRVR=0xe000e014
SYSTCVR=0xe000e018
poke SYSTCSR, 0
poke SYSTRVR, 0x1e847
poke SYSTCSR, 5
for k=1 to 50
  print and(peek(SYSTCVR), 0x00FFFFFF)
  pause 1000
next k
end
```

*- PMODE gpio\_pin, mode*

*mode: 0-IN, 1-OUT, 2-PULLUP, 3-PULLDOWN, 10-ADC, 15-PWM, 20-TSENSOR*

*-AREAD(gpio\_pin), pins=26-29 – read analog pin*

Example: GPIO26 as analog input

```
pmode 26, 10
a=aread(26)
```

Example: read temperature sensor through virtual pin=100

```
pmode 100, 20
t= aread(100)
```

*-AWRITE – PWM (cycle: 65535) – initial implementation*

Example: PWM 25%

```
pmode 22, 15
awrite 22, 16000
```

*- DREAD(gpio\_pin) – read digital pin*

Example

```
REM explorer buttons: a, b, x, y
a=12
b=13
x=14
pmode a, 0
pmode b, 0
pmode y, 0
y=15
pmode y, 0
pmode y, 2
REM modes: 0-IN, 1-OUT, 2-PULLUP, 3-PULLDOWN
for k=1 to 10 step 0
  pause 50
  print dread(y)
next k
```

*-DWRITE gpio\_pin, value – write to digital pin*

Example

*REM explorer led - pin 25*

*pmode 25, 1*

*for k=1 to 10 step 0*

*pause 500*

*dwrite 25,1*

*pause 500*

*dwrite 25,0*

*next k*

*end*

*-OLEDO.91 support (commands may change in the future)*

***NOTE: enable the OLED hardware first with: SM OLED 1***

*DRAW X, Y – draws line from the last point (last PLOT/DRAW commands x, y)*

*PLOT X, Y – draws point*

*LPRINT x, y, "txt"*

*LCLS – clear lcd*

*LREF – refresh LCD (copy mem content to the LCD)*

Example" enable OLD hw (disabled by default), clear screen

*sm oled 1*

*lcls*

Example: clear lcd screen, draw line

*lcls*

*plot 10,10*

*draw 20,20*

*lprint 1, 1, "hi!"*

*lref*

*end*

examples

*REM OLED SINE*

*lcls*

*sineno=1*

*for x=0 to 127*

*plot x,fix(15+15\*sin(2\*3.14159\*x/128))*

*next x*

*lref*

*end*