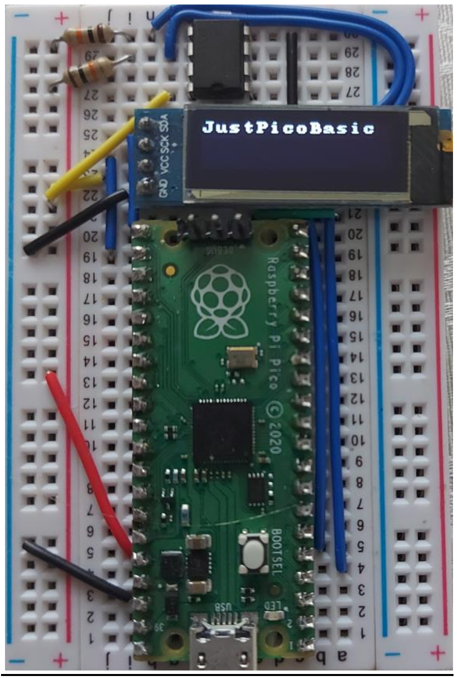
#JustPicoBasic manual

The best way to see how it works is to run a few examples: https://github.com/bgolab/JustBasic/tree/main/examples

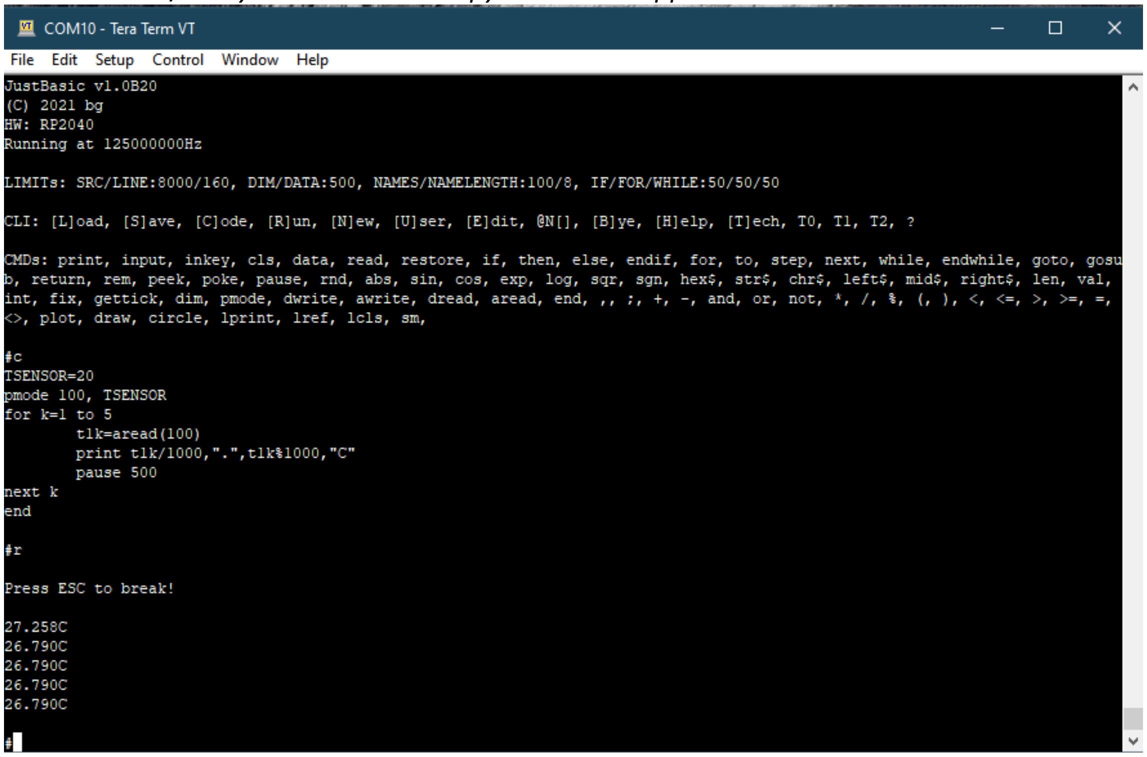
Wiring

Hardware components: RPI PICO, OLED 0.91" I2C, EEPROM 24c64 I2C, 2x resistors 10kohm



Terminal

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



```
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, @N[], [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, gosu
b, 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 (i.e. 'CLS' and 'cls' are the same). Names recognizes small and capital letters (i.e. 'as' is different than 'AS')

Program structure

Program Structure	***Description/Comments***	***Example***

code
gosub subroutines1
end
subroutines1:
[subroutine1 code]
return
	Command 'end' has to follow the last line of the code. Subroutines have to follow the 'end'.	*print "Hi!"*
		gosub callme
		end
		callme:
		    print "Hi!"
		return

System commands

Command	***Description/Comments***	***Example***
ESC key	*Break the program while Running or prevent from Loading.*	
?	*Shows info about the VM (ver, available commands)*	
c	*Code – show code ('SM LN 0' to turn off line numbers)*	
r *r <code>*	*Run Program from memory* *Run single line of program NOT stored in memory (ad-hoc)*	*r n=2 data 2, 3 s=0 for i=1 to n read a s=s+a next i print "s=", s end*
n	*New – clear VM memory and code*	
b	*Bye: PICO – reboot VM in disk mode, Windows - exist*	
l *s*	*l - Load program from EEPROM (auto.bas)* *s - Save program to EEPROM (auto.bas)*	
ee *ed*	*ee - EEPROM erase – to decide if required, now disabled* *ed - EEPROM dump – show EEPROM content*	
is	*I2C scan – show I2C devices on both I2C buses*	
t0|t1|t2	***Build in program tracing.*** *t0 - Disable tracing; t1 - Stepping mode; t2 - Run with tracing*	*T1|T2 - enable particular tracing mode* *R – run program in T1 or T2 mode*
@N *@N <code>*	***Build-in editor.*** *New code is appended at the end of code;* *'c' shows code and internal line numbers for @N commands.*	*@3 – removes 3rd line of code* *@4 CLS – inserts 'CLS' before line 4*

The language

MISC

Command	***Description/Comments***	***Example***
SM <entity> 1|0	*System Mode command for system configuration. Entities:* *ESC (default=enabled) – enable / disable ESC key check (disable to boost perf)* *OLED (default=disabled) – enable OLED hw (cannot be disabled now)* *LN (default=enabled) – enable / disable line numbering for 'c' command*	*SM ESC 0* *SM OLED 1* *SM LN 0*
REM	*Comment*	*REM MyProc*
CLS	*Clear Screen*	

VARIABLES

-variable name: up to 8chars letter&digits starting w/ a letter(digits, '#", '$', '_', ':' accepted)

Command	***Description/Comments***	***Example***
var	*INT var (no suffix), name=expr, 1st-reference creates var(value=0);*	*hi5=2*
var#	*FLOAT var (# suffix), name#=expr, 1st-reference creates var(value=0);*	*w#=2.5*
var$	*STR var ($ suffix), name$=sexpr*	*name$="John"*
DIM var(n)	*INT array, 1-dimension, DIM name(size); name(item)=expr*	*DIM a(3) a(0)=3*
DIM var#(n)	*FLOAT array, 1-dimension, DIM name#(size); name#(item)=expr*	*DIM(b#(3) b#(0)=2.5*
DIM var$(n)	*STRING arrays NOT supported yet*	

EXPRESSIONS

*-expr: INT/FLOAT, +, -, *, /, %,(,), vars; strings supports only '+' in expressions + string functions and variables*

Example: complex expressions
*sy=2*abs(-15) + a*20*
v#=fexpr; b#=a#-2(2+3)+abs(-1.0)+aa#(2);*
v$="a"+a$+left$(str$(13),1);
PRINT sexpr; VAL("-1")->-1

PROGRAM FLOW CONTROL

Command	***Description/Comments***	***Example***
FOR var=expr TO expr [STEP expr] [] NEXT var	*if STEP[default=1] is negative var decreases; FLOAT supported; nesting supported*	*FOR i = 5 TO 1 STEP -1 NEXT I END*
WHILE condition [code] ENDWHILE	*FLOAT supported; nesting supported; AND/OR/NOT supported;*	*a=0 while a<5 print a a=a+1 endwhile end*
IF condition THEN [code] [ELSE] [code] ENDIF	*FLOAT supported; nesting supported AND/OR/NOT) supported;*	*if a>1 and b#>3.4 then print "ok" else print "bad" endif*
label: *GOTO label*	*label name with colonat the end : up to 8chars letter & digits starting w/ a letter('_' accepted); can be located(before and after the GOTO; Label: [code] GOTO label*	*k=1* *again:* *print k k=k+1* *if k<5 then goto again: endif*
GOSUB label *label: [code] RETURN*	*label can be located after END*	*gosub task0 end* *task0: print "done" return*
RETURN		
END	*last instruction (GOSUB labels MUST be located behind the END)*	

INPUT, OUTPUT, DATA

Command	***Description/Comments***	***Example***
PRINT expr[, sexpr], [;]	*',' to separate items, ';' to skip NEW LINE*	*PRINT "How:", 6/3 ;* *PRINT 1 (w/ NEW LINE); PRINT 1; (w/o NEW LINE)*
INPUT var, …	*Assign int/float/string values to var or array element*	*INPUT a(2), d#, name$* *print a(2), d#, name$*
DATA expr, fexpr, str;	*INT/FLOAT/STR supported*	*DATA 1.5, 2*a*
READ a, b#, d$	*Assign DATA specified input to vars*	*READ v, v#, v(), v#();*
RESTORE clears data pointer		

BUILT-IN FUNCTIONS

Command	***Description/Comments***	***Example***
LEFT$/RIGHT$/MID$(sexpr), LEN/VAL(sexpr)		*k$=LEFT$("abcdefgh", 3) + "1234"* *PRINT VAL("-1234")+1* *i=1234567* *i$=MID$(STR$(i), 2, 3) PRINT i$* *PRINT MID$(STR$(i), 2, 3)*
HEX$/STR$/CHR$(expr)		*PRINT HEX$(NOT(0x0F))* *a=65* *d$=chr$(a)*
SIN/COS/SQR/EXP/LOG(expr)		*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(max)	*hw-based random generator with von Neuman extractor-whitenizer*	*PRINT "RND: ", RND(1000)*

SGN/ABS(expr)		
GETTICK()	Tick number	a=gettick()
PAUSE msec		PAUSE 2*500
INKEY()	Current key (if pressed), otherwise 0; non-blocking (no-wating)	
INT/FIX(expr)	QBASIC like	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(expr,expr), OR(expr,expr), NOT(expr)		PRINT AND(0x03,0x0F) PRINT OR(0x01,0x02) PRINT HEX$(NOT(0x0F))

PICO HARDWARE SUPPORT

Command	***Description/Comments***	***Example***

PEEK(addr)	hex supported	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

POKE addr, value	hex supported	
PMODE pin, mode	mode: 0-IN, 1-OUT, 2-PULLUP, 3-PULLDOWN, 10-ADC, 15-PWM, 20-TSENSOR	
AREAD(pin)	Read analog pin; pins=26-29 – analog pin; 100 –temperature virtual pin	pmode 26, 10 voltage=aread(26) pmode 100, 20 temp= aread(100)
AWRITE pin, cycles	PWM duty=cycles/65535 (max cycle: 65535) – initial implementation;	pmode 22, 15 awrite 22, 16000

DREAD(pin)	Read digital pin	REM explorer buttons: a, b, x, y
		a=12 b=13 x=14 y=15
		pmode y, 0 pmode y, 2
		for k=1 to 2 step 0 pause 50 print dread(y) next k
		end

DWRITE pin, value	Write digital pin	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

GRAPHIC LCD/OLED SUPPORT *(currently: OLED0.91 support,(commands may change in the future)*

Command	***Description/Comments***	***Example***
SM OLED 1	Enable OLED support	

LPLOT X, Y	Draw point	for x=0 to 127
		        lplot x,fix(15+15*sin(2*3.14159*x/128))
		next x
		lref

		end
LDRAW X, Y	*Draw line from the last PLOT / DRAW x, y*	*plot 10,10 ldraw 20,20 lref end*
LCIRCLE x, y, r	*Draw circle*	
LPRINT x, y, "txt"	*syntax to change*	*lprint 1, 1, "hi!" lref end*
LCLS	*Clear Screen*	*lcls end*
LREF	*Refresh LCD (copy mem content to LCD)*	