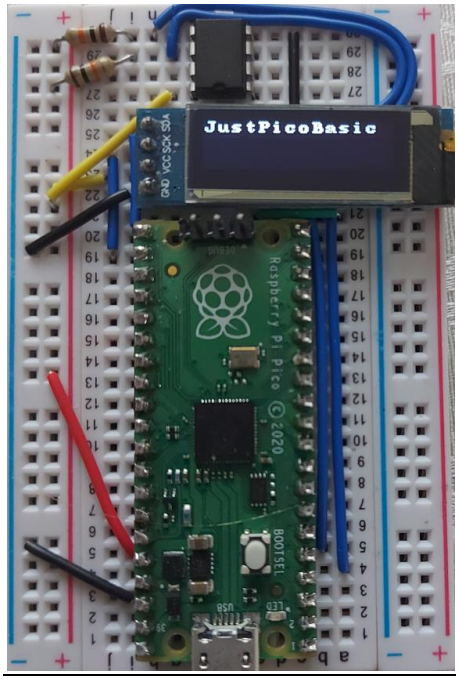
#JustPicoBasic manual

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

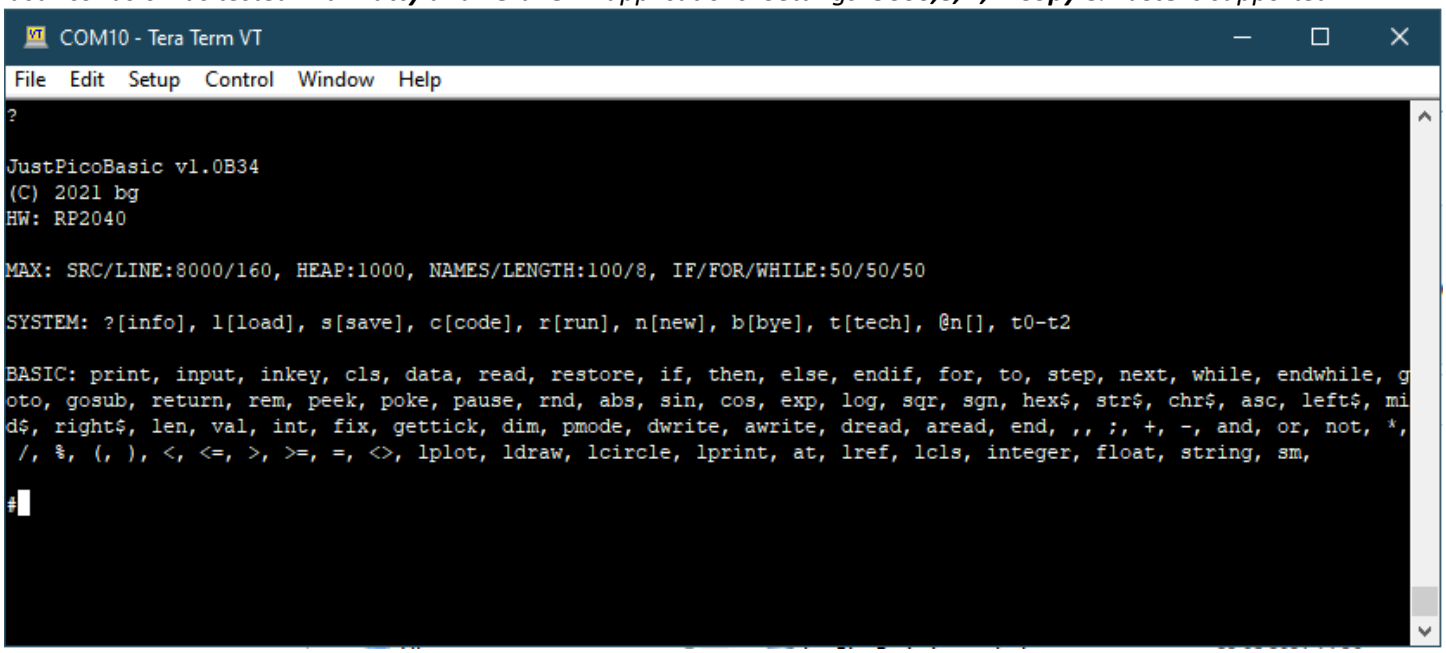
Wiring

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



Terminal emulator

JustPicoBasic was tested with ***Putty*** *and* ***TeraTerm*** *applications. Settings:* ***9600,8,N,1****.* ***Copy & Paste*** *is supported.*



Get started

Power the PICO through the USB. Start the terminal emulator.
Drag & Drop the current UF2 file (e.g. JustBasic-1.0b30.uf2) into the PICO emulated disk. The JustPicoBasic will boot.
*Enter '**?**' (the question mark) and press ENTER. You will see available commands (CLI & BASIC commands).*
Enter the following program:

```
pmode 25,1
```

```
for k=1 to 5
        dwrite 25,1
        pause 500
        dwrite 25,0
        pause 500
next k
end
```

*Use enter '**c**' to show the code. Use '**r**' to run the code. The built-in LED will blink.'**n**' clears the program memory.*
*You can use **@N** (e.g. @0) to delete Nth line or **@N <code>** (@0 print 1) to insert the line of code.*
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 subr1 end subr1: [subr1 code] return	Command 'end' has to follow the last line of the main 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	Show code ('SM LN 0' prevents from displaying line numbers)	
r r <code>	Run – run code from SRAM memory Run - run single line of code 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	**Program Flow 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>	**Pico Editor.** Normally new code is appended at the end; @N delete Nth line; @N <code> insert <code> before Nth line	@3 – removes 3rd line of code @4 CLS – inserts 'CLS' before line 4

The BASIC language

Command	**Description/Comments**	**Example**
SM <entity> 1|0	System Mode command for system entities configuration. Entities: ESC (default=enabled) – enable / disable ESC key check (disable to boost performance) LN (default=enabled) – enable / disable line numbering for 'c' command OLED (default=disabled) – enable OLED hw (cannot be disabled now) NOTE: OLED auto-detection is supported now so 'SM OLED 1' is not longer required	SM ESC 0 SM OLED 1 SM LN 0
REM	Comment	REM MyFunc
CLS	Clear Screen	CLS

VARIABLES & EXPRESSIONS
Suffix-based (suffix #, $, OR no suffic to declare variable type) varname and array syntax

-variable name: up to 8chars letter&digits starting w/ a letter (digits, '#", '$', '_', ':' accepted)
*-expr(arithmetic expression): combination of INT/FLOAT and ops/brackets(+, -, *, /, %,(,)) and INT/FLOAT vars;*
-sexpr(string expression): combination of string, string functions (with suffix $) and string vars (with suffix $) and '+' op
-variable type differentiation through the suffix (no suffix – integer, '#' suffix – float, '$' suffix – string
-array index counts from 0 (for: DIM a(3) available array elements are referred by a(0), a(1), a(2)
-string arrays must have 2-dimentions e.g. DIM a$(2,5) – 2 strings, maximum length of a string is 5 characters; string arrays are always referred through single index a$(0), a$(1) for DIM a$(2,5) array

Command	**Description/Comments**	**Example**
var=expr	INT var, name=expr, 1st-reference creates var(value=0);	sy=2*abs(-15) + a*20
var#=expr	FLOAT var (# suffix), name#=expr, 1st-reference creates var w/ value=0	w#=2*a#+abs(-1.0)
var$=sexpr	STR var ($ suffix), name$=sexpr	v$=a$+left$(str$(13),1)
DIM var(s1[,s2]), var#(s1[,s2]), var$(s1,s2)	INT/FLOAT/STRING array, 1/2-dimensions; multi-array declaration (array names separated by comma)	DIM a(3), b#(4,4), c$(4,5)
var(expr[,expr])=expr	INT: name(item)=expr	a(0)=3
var#(expr[,exp])=expr	FLOAT: name#(item)=expr	b#(0)=2.5
var$(expr)=sexpr	STRING: name$(item)=sexpr	c$(2)="abc"

Suffixless varname and array syntax (experimental)
-variable needs to be declared (if not declared integer type is assumed), multi-declaration in single command supported
-initialization during the declaration phase is not supported yet
-array index counts from 0 (for: DIM a(3) available array elements are referred by a(0), a(1), a(2)
-string arrays must have 2-dimentions e.g. string a(2,5) – 2 strings, maximum length of a string is 5 characters; string arrays are always referred through single index a(0), a(1) for string a(2,5) array

Command	**Description/Comments**	**Example**
integer vname, vname2(s1[,s2])),..	Declare integer var or 1/2-dimensional array; value=0 set	integer a, b(8,2) b(0,0)=1
float vname, vname2(s1[,s2]),..	Declare float var or 1/2-dimensional array; value=0 set	float c, b(0,0)=1.0
string vname, vname2(s1,s2),..	Declare string var or 1/2-dimensional array; value=null set	string b(2,4) a(1)="no"

PROGRAM FLOW CONTROL
-cond: logical expression e.g. a>5 and b<10
-lexpr(logical expression): combination of conditional and logical operators

Command	**Description/Comments**	**Example**
FOR v=expr TO expr [STEP expr] [code] NEXT var	if STEP[default=1] is negative var decreases; INT/FLOAT supported; nesting supported	FOR i = 5 TO 1 STEP -1 NEXT I END
WHILE lexpr [code] ENDWHILE	INT/FLOAT supported; nesting supported; cond: AND/OR/NOT supported;	a=0 while a<5 print a a=a+1 endwhile end
IF lexpr THEN [code] [ELSE] [code] ENDIF	INT/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 starts with a letter, terminated by colon; up to 8 letter & digits(plus '_');	k=1 again: print k k=k+1 if k<5 then goto again: endif
GOSUB label	Label must be located after END	gosub task0 end
label: RETURN	label: [code] RETURN	task0: print "done" return
END	Last instruction. GOSUB labels follows END	

INPUT, OUTPUT, DATA

Command	**Description/Comments**	**Example**
PRINT expr[, sexpr], [;]	Prints expr, sexpr separated by ',' ';' to skip NEW LINE	PRINT "6/3=", 6/3 (with NEW LINE) PRINT 1; (w/o NEW LINE because of ';')
INPUT var,...	Assign int/float/str values to (array) var	INPUT a(2), d#, n$ print a(2), d#, n$
DATA expr, sexpr;	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	Reset data pointer	

BUILT-IN FUNCTIONS

Command	Description/Comments	Example
LEFT$$/RIGHT$/MID$(sexpr),	Left$/right$/mid$ - string functions,	k$=LEFT$("abc", 2) + "123" i=12 i$=MID$(STR$(i),2,3) PRINT i$
HEX$/STR$/CHR$(expr)	Hex$(expr to hex string), Str$(expr to string), Chr$(expr%256 to ascii e.g. 65 to 'A')	PRINT HEX$(NOT(0x0F)) a=65 d$=chr$(a)
LEN/VAL/ASC(sexpr)	Val(string to value); LEN(string length), ASC(ascii code of the 1st char of the string)	PRINT VAL("-1234")+1 PRINT ASC("AB")->65
SIN/COS/SQR/EXP/LOG(expr), SGN/ABS(expr)	Math functions	PRINT "S:", SQR(5), "E:", EXP(1), "L:", LOG(2.71), "S:",SIN(3.14/6)
RND(max)	Hw-based random generator with von Neuman extractor-whitenizer	PRINT "RND: ", RND(1000)
GETTICK()	Tick number	a=gettick()
PAUSE msec	Delay (blocking) in msec	PAUSE 2*500
INKEY()	Pressed key, OR 0; non-blocking (no-wating)	
INT/FIX(expr)	QBASIC like	a=INT(1.1) b=INT(-1.1) c=FIX(1.9) d=FIX(- 1.9) PRINT a, ", ", b, ", ", c, ", ", d (1,-2,1,-1)
AND/OR(expr,expr), NOT(expr)		PRINT AND(0x3,0xF), OR(0x1,0x2), HEX$(NOT(0x0F))

PICO HARDWARE SUPPORT

Command	Description/Comments	Example

PEEK(addr)
POKE addr, value
	Memory read / write; 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

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; pin=100 –temperature virtual pin	pmode 26, 10 voltage=aread(26) pmode 100, 20 temp= aread(100)
AWRITE pin, cycles	PWM duty=cycles/65535 cycle: 0-65535)	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
DWRITE pin, value	Write digital pin	REM explorer led - pin 25 pmode 25, 1 dwrite 25,1 pause 3000 dwrite 25,0

GRAPHIC LCD/OLED SUPPORT (currently: OLED0.91 support)

NOTE: As of 1.0b31 OLED auto-detection was added - 'SM OLED 1' is not longer required

Command	Description/Comments	Example
SM OLED 1	Enable OLED support	
LPLOT X, Y	Draw point at X, Y	for x=0 to 127 lplot x,fix(15+15*sin(2*3.14*x/128))

		next x *lref*
LDRAW X0, Y0, X1, Y1	*Draw line between X0, Y0 and X1, Y1*	*ldraw 10, 10, 20,20 lref*
LCIRCLE x, y, r	*Draw circle at x, y, r*	*lcircle 15, 15, 10*
LPRINT expr, sexpr [AT x,y]	*Prints expr, sexpr separated by ',';* *AT x, y (default 0,0)*	*lprint "2+2=", 2+2 AT 10,10* *lref*
LCLS	*Clear Screen*	*lcls*
LREF	*Refresh LCD (copy mem content to LCD)*	*lref*