

#JustPicoBasic manual

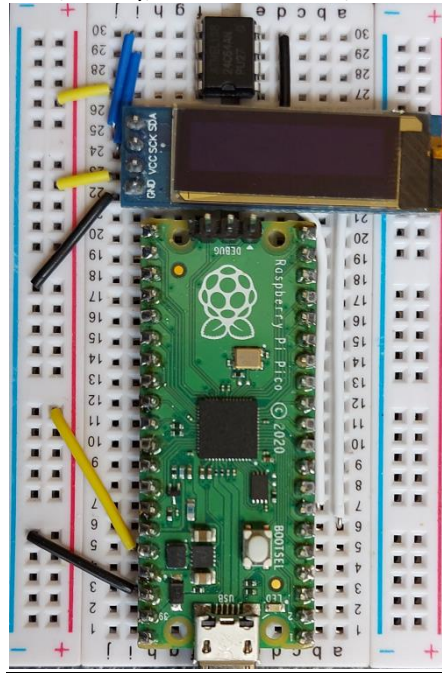
The project repository (i.e. binaries, manual, and examples) are available at: <https://github.com/bqolab/JustPicoBasic>

Wiring

NOTE: PICO running JustPicoBasic v1.0b50 and onward can boot without EEPROM. Older BASIC versions need to press ESC when EEPROM is NOT available (otherwise the PICO gets stuck).

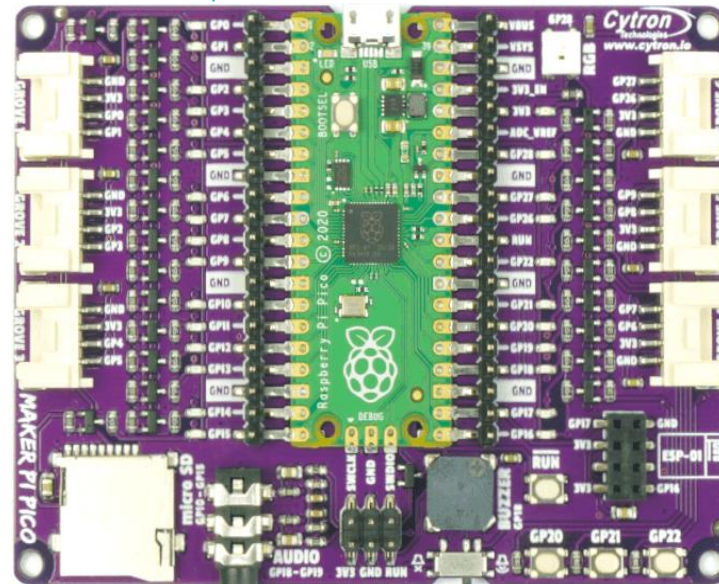
NOTE: I2C addresses reserved for hardware auto-detection: OLED 0.91" - 0x3C, EEPROM 24c64 - 0x50

1. Original breadboard wiring: PICO I2C0(GP04, GP05), I2C OLED 0.91" , I2C EEPROM 24c64



2. Cytron Maker Pi Pico board wiring: PICO I2C0(GP04, GP05), I2C OLED 0.91" , I2C EEPROM 24c64

NOTE: Use Grove 3 connector with I2C hub to split the I2C bus into two devices



Getting started

Press the **BOOTSEL** button and hold it while you connect the other end of the micro USB cable to your PC. Copy the latest BASIC UF2 file (e.g. JustBasic-xx.uf2) to the PICO disk. Start a terminal emulator e.g. **TeraTerm** (use settings: **9600,8,N,1**). The PICO will boot. Enter '?' and press ENTER to see available commands.

```

COM4 - Tera Term VT
File Edit Setup Control Window Help
?

JustPicoBasic v1.0B50
(C) 2021 bg
HW: RP2040

SYSTEM: ?, l[load], s[save], c[code], r[run], n[new], b[bye], @n[], t0-t2

BASIC: integer, float, string, print, input, inkey, cls, data, read, restore, if, then, else, endif, for, to, step, break, next, while, endwhile, goto, gosub, return, rem, abs, sin, cos, exp, log, sqr, sgn, hex$, str$, chr$, asc, left$, mid$, right$, len, val, int, fix, dim, end, ,, :, +, -, and, or, not, *, /, %, (, ), <, <=, >, >=, =, <>, ==, !=, rnd,

HW: peek, poke, pause, gettick, pmode, dwrite, awrite, dread, aread, lplot, ldraw, lcircle, lprint, at, lref, lcls, iinit, ideinit, ireadable, iwriteable, iread, iwrite, uinit, udeinit, ureadable, uwritable, uread, uwrite, sinit, sdeinit, sreadable, swritable, sread, swrite, ts, esc, oled, tech, sm,

CONST: IN, OUT, PULLUP, PULLDOWN, ADC, PWM, TSENSOR, I2C0, I2C1, UART0, UART1, SPI0, SPI1, HIGH, LOW, ENABLE, DISABLE,

#
  
```

Enter an example BASIC program (you can use Copy & Paste)

```

pmode 25,OUT
for k=1 to 5
    dwrite 25,HIGH pause 500
    dwrite 25,LOW pause 500
next k
end
  
```

Use 'c' or 'cc' to see the code, 'r' to run the code. The built-in LED will blink. The '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. Capital and small letters accepted for keywords (i.e. 'CLS' equals 'cls'). For the names small and capital letters are recognized (i.e. 'as' is different than 'AS')

NOTE: If you use TeraTerm and some characters are lost during Copy & Paste operation consider increasing the 'Paste delay per line' in Setup->Additional Settings->Copy and Paste

Program structure

Program Structure	Description/Comments	Example
<pre> gosub subr1 end subr1: [subr1 code] return </pre>	Command 'end' has to be the last line of the main code. Subroutines have to follow the 'end'.	<pre> gosub callme end callme: print "Hi!" return </pre>

System commands

Command	Description/Comments	Example
ESC key	Stop the running program or stop initial loading program while booting	
?	Shows info about the VM (version, available commands)	
c, cc	c-show the code w/ line numbers, cc-show the code w/o line numbers	

<i>r</i>	Run – run code from SRAM memory	
<i>r <code></i>	Run - run single line of code NOT stored in memory (ad-hoc)	<i>r for i=1 to 5 print i next i end</i>
<i>n</i>	New – clear VM memory and code	
<i>b</i>	Bye: PICO – reboot VM in disk mode, Windows - exist	
<i>l</i>	l - Load program from EEPROM (auto.bas)	
<i>s</i>	s - Save program to EEPROM (auto.bas)	
<i>ee</i>	<i>ee - EEPROM erase – now command disabled</i>	
<i>ed</i>	ed - EEPROM dump – show EEPROM content	
<i>is 0 1</i>	I2C scan devices (bus: 0 or 1); NOTE: 'is' stuck when a bus not terminated	<i>is 0 is 1</i>
<i>t0 t1 t2</i>	Program Flow Tracing: t0 – OFF; t1 - Step Mode; t2 – Cont Mode	<i>t1 t2 - enable tracing mode</i>
<i>@N</i>	Pico Editor: @N - delete N-th line	<i>@3 – removes 3rd line of code</i>
<i>@N <code></i>	Pico Editor: @N <code> - insert <code> before N-th line	<i>@4 cls – insert 'cls' before line 4</i>

The BASIC language

Command	Description/Comments	Example
<i>SM <entity> enable/disable</i>	System Mode. Entity: ESC (default=enabled); ESC check; 'disable' boosts perf) OLED (default: auto-detected, if OLED exists: enabled)	<i>sm esc disable</i>
<i>REM</i>	Comment	<i>rem MyFunc</i>
<i>CLS</i>	Clear Screen	<i>cls</i>

VARIABLES & EXPRESSIONS

Suffix-based (suffix #, \$, OR no suffix 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-dimensions 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
- array size can be an expression

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

BUILT-IN CONSTANTS

Command	Description	Example
<i>ENABLE (1), DISABLE (0), HIGH (1), LOW (0)</i>	Generic const	<i>dwrite 1,low</i>
<i>IN, OUT, PULLUP, PULLDOWN, ADC, PWM, TSENSOR, I2C0, I2C1, UART0, UART1, SPI0, SPI1</i>	HW const	<i>pmode 2,out</i>

Suffixless varname and array syntax

- 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 have always 2-dimensions e.g. string a(2,5) – two 5-chars strings; string arrays are referred via single index a(0) then
- array size can be an expression

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

PROGRAM FLOW CONTROL

-lexpr: combination of expressions, conditionals('>', '<', '>=', '<=', '=', '==', '<>', '!=') and logical operators (AND, OR, NOT); e.g. *a>5 AND gettick()<1000*;

NOTE: the '<>' is equivalent to '!=', the '=' is equivalent to '=='

Command	Description/Comments	Example
<i>FOR v=expr TO expr [STEP expr] [code] NEXT var</i>	If STEP[default=1] is negative var decreases; INT/FLOAT supported; nesting supported	<i>FOR k = 5 TO 1 STEP -1 NEXT k END</i>
<i>WHILE lexpr [code] ENDWHILE</i>	INT/FLOAT supported; nesting supported; cond: AND/OR/NOT supported;	<i>REM Simple while-loop a=0 while a<5 print a a=a+1 endwhile REM Wait for 500msec (non-blocking) waitTime=500 t_start=gettick() while gettick()-t_start>waitTime endwhile</i>
<i>BREAK</i>	Can be used in FOR and WHILE loops.	<i>while 1>0 k=inkey() if(k!=0) then print k endif if(k==113) then break endif endwhile</i>
<i>IF lexpr THEN [code] [ELSE] [code] ENDIF</i>	INT/FLOAT supported; nesting supported AND/OR/NOT supported;	<i>if a>1 and b#>3.4 then print "ok" else print "bad" endif</i>
<i>label: GOTO label</i>	Label name starts with a letter, terminated by colon; up to 8 letter & digits(plus '_');	<i>k=1 again: print k k=k+1 if k<5 then goto again: endif</i>
<i>GOSUB label</i>	Label must be located after END	<i>gosub task0 end</i>
<i>label: RETURN</i>	label: [code] RETURN	<i>task0: print "done" return</i>
<i>END</i>	Last instruction. GOSUB labels follows END	

INPUT, OUTPUT, DATA

Command	Description/Comments	Example
<i>PRINT expr[, sexpr], [;]</i>	Prints expr, sexpr separated by ',' ';' to skip NEW LINE	<i>PRINT "6/3=", 6/3 (with NEW LINE) PRINT 1; (w/o NEW LINE because of ';')</i>
<i>INPUT var,...</i>	Assign int/float/str values to (array) var	<i>INPUT a(2), d#, n\$ print a(2), d#, n\$</i>
<i>DATA expr, sexpr;</i>	INT/FLOAT/STR supported	<i>DATA 1.5, 2*a</i>
<i>READ a, b#, d\$</i>	Assign DATA specified input to vars	<i>READ v, v#, v(), v#();</i>
<i>RESTORE</i>	Reset data pointer	

BUILT-IN FUNCTIONS

Command	Description/Comments	Example
<i>LEFT\$(sexpr, expr)</i>	Left part of the string	<i>k\$=LEFT\$("abc", 2) + "123" i=12</i>
<i>RIGHT\$(sexpr, expr)</i>	Right part of the string	
<i>MID\$(sexpr,expr,expr)</i>	Middle of the string	<i>i\$=MID\$(STR\$(i),2,3) PRINT i\$</i>
<i>HEX\$(expr)</i>	Hex\$(expr to hex string)	<i>PRINT HEX\$(NOT(0x0F))</i>
<i>STR\$(expr)</i>	Str\$(expr to string)	
<i>CHR\$(expr)</i>	Chr\$(expr%256 to ascii e.g. 65 to 'A')	<i>a=65 d\$=chr\$(a)</i>
<i>LEN(sexpr)</i>	LEN(string length)	<i>PRINT LEN("1234")->4</i>
<i>VAL(sexpr)</i>	Val(string to value)	<i>PRINT VAL("-1234")+1->-1233</i>
<i>ASC(sexpr)</i>	ASC(ascii code of the 1 st char of the string)	<i>PRINT ASC("AB")->65</i>
<i>SIN(expr)</i>	Sine	<i>PRINT "SIN:", SIN(3.14/6)</i>
<i>COS(expr)</i>	Cosine	<i>PRINT "COS:", COS(3.14/6)</i>

<i>SQR(expr)</i>	Square root	<i>PRINT "SQRT:", SQR(5)</i>
<i>EXP(expr)</i>	Exponential	<i>PRINT "EXP:", EXP(1)</i>
<i>LOG(expr)</i>	Logarithm	<i>PRINT "LOG:", LOG(2.71)</i>
<i>SGN(expr)</i>	Sign	<i>PRINT "SGN:", SGN(-5)</i>
<i>ABS(expr)</i>	Absolute	<i>PRINT "ABS:", ABS(-5)</i>
<i>RND(max)</i>	Hw-based rnd generator + von Neuman whitenizer	<i>PRINT "RND: ", RND(1000)</i>
<i>GETTICK()</i>	Tick number	<i>a=gettick()</i>
<i>PAUSE msec</i>	Delay (blocking) in msec	<i>PAUSE 2*500</i>
<i>INKEY()</i>	Pressed key, OR 0; non-blocking (no-waiting)	
<i>INT(expr)</i>	QBASIC like	<i>a=INT(1.1) b=INT(-1.1) PRINT a, ", ", b (1,-2)</i>
<i>FIX(expr)</i>		<i>c=FIX(1.9) d=FIX(-1.9) PRINT c, ", ", d (1,-1)</i>
<i>AND(expr,expr)</i>		<i>PRINT AND(0x3,0xF)</i>
<i>OR(expr,expr)</i>		<i>PRINT OR(0x3,0xF)</i>
<i>NOT(expr)</i>		<i>PRINT HEX\$(NOT(0x0F))</i>

GPIO SUPPORT

Command	Description/Comments	Example
<i>POKE addr,value</i>	Memory write; hex supported	<i>REM SYSTICK (ST)</i>
<i>PEEK(addr)</i>	Memory read; hex supported	<i>STCSR=0xe000e010 STRVR=0xe000e014 STCVR=0xe000e018 poke STCSR,0 poke STRVR,0x1e847 poke STCSR,5 for k=0 to 9 print and(peek(STCVR),0xFFFFF) pause 100 next k</i>
<i>PMODE pin,mode</i>	m: const: IN,OUT,PULLUP,PULLDOWN, ADC, PWM, TSENSOR	<i>pmode 100, tsensor temp= aread(100)</i>
<i>AREAD(pin)</i>	Read analog pin; pins=26-29; pin=100 – temperature virtual pin	<i>pmode 26, adc voltage=aread(26) pmode 100,tsensor temp= aread(100)</i>
<i>AWRITE pin,cycle</i>	PWM duty=cycle/65535,cycle<65535	<i>pmode 22, pwm awrite 22, 16000</i>
<i>DREAD(pin)</i>	Read digital pin	<i>y=15 pmode y,in pmode y,pullup for k=1 to 2 step 0 pause 50 print dread(y) next k</i>
<i>DWRITE pin,value</i>	Write digital pin	<i>pmode 25,out dwrite 25,high pause 3000 dwrite 25,low</i>

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

NOTE: use 'lref' to refresh the screen as ALL the lxxx command update only the memory buffer

Command	Description/Comments	Example
<i>LPLOT X, Y</i>	Draw point at X, Y	<i>for x=0 to 127 lplot x,fix(15+15*sin(6.28*x/128)) next x lref</i>
<i>LDRAW X0, Y0, X1, Y1</i>	Draw line from X0, Y0 to X1, Y1	<i>ldraw 10, 10, 20,20 lref</i>
<i>LCIRCLE x, y, r</i>	Draw circle at x, y, r	<i>lcircle 15, 15, 10</i>
<i>LPRINT expr, sexpr [AT x,y]</i>	Print expr, sexpr separated by ',' [AT x, y] (default 0,0)	<i>lprint "2+2=", 2+2 AT 10,10 lref</i>
<i>LCLS</i>	Clear Screen	<i>lcls lref</i>
<i>LREF</i>	Refresh LCD (copy mem to LCD)	<i>lref</i>

TONE SUPPORT (tested on piezo)

Command	Description/Comments	Example
<i>TONE pin, freq</i>	Start PWM square signal at freq, duty 50% on a pin; freq>10Hz; non-blocking;	<i>TONE 18, 440</i>
<i>NOTONE pin</i>	Stop generating the signal on a pin	<i>NOTONE 18</i>

I2C/UART/SPI SUPPORT

-the API is based on the PICO SDK API; all xREAD/xWRITE are blocking (*can wait for the data forever*);

-non-blocking approach possible by checking if the fifo is empty using xREADABLE/xWRITEABLE functions

NOTE: Internal buffer size = 256 (max number of bytes to read/write);

NOTE: SPI csn is controlled through gpio (dwrite)

NOTE: UART data,stop,parity options are NOT supported now

NOTE: SPI format is NOT supported now

NOTE: string is preferred to WRITE commands, for READ command need to allocate fake string to space the read data

Command	Description/Comments	Example
IINIT in, sda, scl[,baud] UINIT in, tx, rx[,baud[,data,stop,parity]] SINIT in, tx, rx, sck[,baud[,format]]	initialize hw; sda/scl/tx/rc/sck: GPIO in: i2c0, i2c1, uart0, uart1, spi0, spi1 defaults: i2c:400k; uart:115.2k,8,N,1; spi:10M	REM UART loop-test l=8 integer bi(l), bo(l) for k=0 to l-1 bi(k)=k next k uinit uart0, 0, 1, 9600 uwrite uart0, bi, l pause 100 uread uart0, bo, l udeinit uart0 end
IREAD in, ad, arr , len, nstop UREAD in, arr , len SREAD in, arr , len	read / write operations ad: i2c device address; arr: array name for read;	
IWRITE in, ad, arr/str , len, nstop UWRITE in, arr/str , len SWRITE in, arr/str , len	arr/str: array OR string name for write; len: number of bytes to read/write; nstop: 1-no Stop issued, 0-Stop issued	
IDEINIT in; UDEINIT in; SDEINIT in	de-initialize hw	end
IREADABLE (in); IWRITABLE (in) UREADABLE (in); UWRITABLE (in) SREADABLE (in); SWRITABLE (in)	for non-blocking write / write returns: 0,1;	----- REM MAX7219 (SPI test) TESTREG=0x0F tx=11 rx=12 sck=10 csn=13 baud=10000000 ll=2 integer buf(ll) pmode csn,OUT sinit spi1, tx, rx, sck, baud buf(0)=TESTREG buf(1)=0x1 gosub MaxWrite pause 1000 buf(0)=TESTREG buf(1)=0x0 gosub MaxWrite sdeinit spi1 end MaxWrite: dwrite csn, LOW swrite spi1, buf, ll dwrite csn, HIGH return