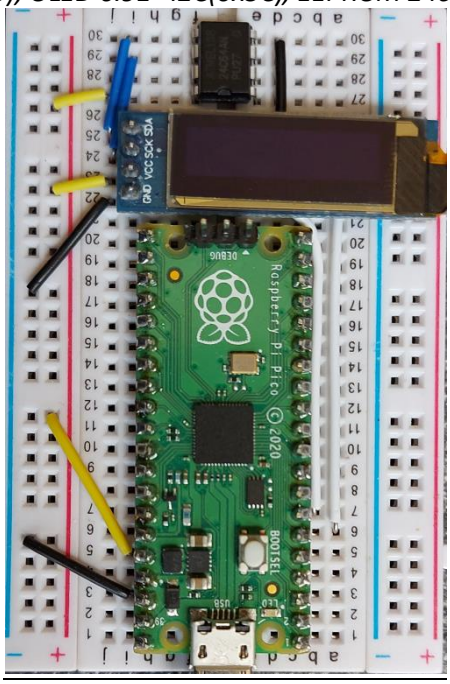
#JustPicoBasic manual

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

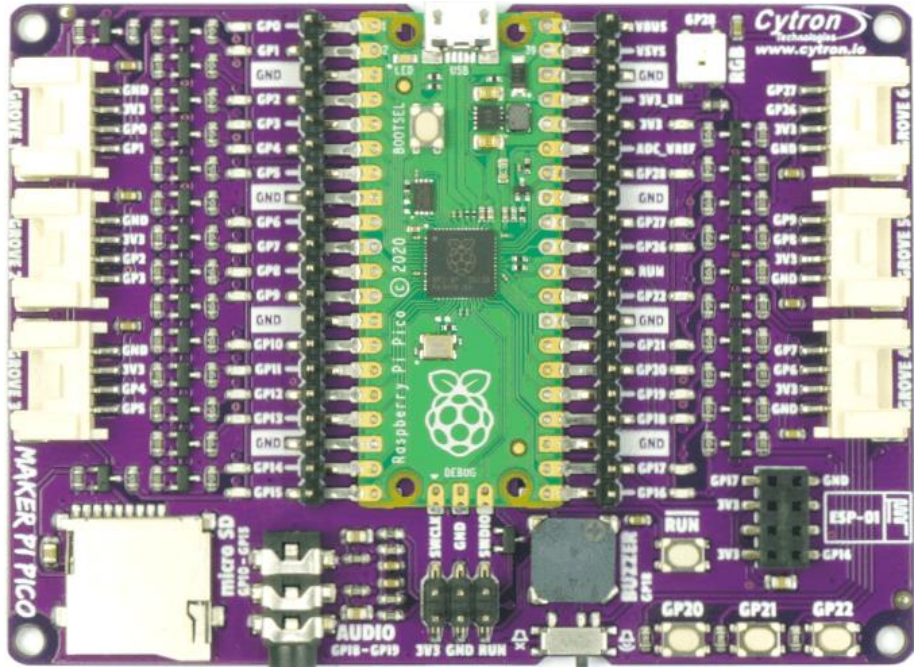
Wiring

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

Breadboard wiring: PICO I2C0(GP04, GP05), OLED 0.91" I2C(0x3C), EEPROM 24c64 I2C(0x50)

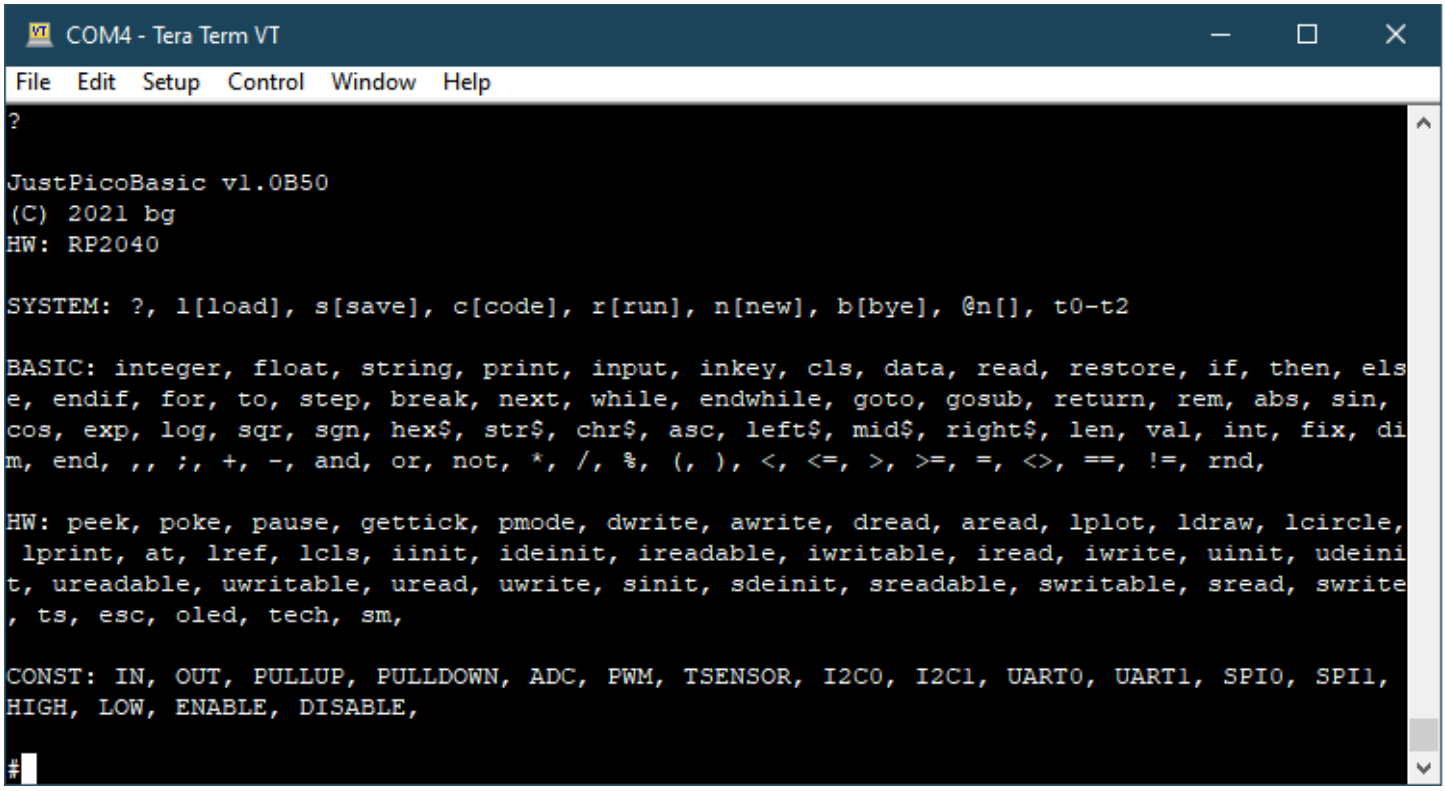


Cytron Maker Pi Pico board wiring: use Grove 3 connecttor for OLED 0.91" I2C(0x3C), EEPROM 24c64 I2C(0x50)



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.*



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')

Program structure

Program Structure	Description/Comments	Example

gosub subr1
end
subr1:
  [subr1 code]
return
	Command 'end' has to be the last line of the main code.
	Subroutines have to follow the 'end'.
		gosub callme
		end
		callme:
		  print "Hi!"
		return

System commands

Command	Description/Comments	Example
ESC key	Stop the running program or prevent program from loading 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	
r	Run – run code from SRAM memory	
r <code>	Run - run single line of code NOT stored in memory (ad-hoc)	r for i=1 to 5 print i next i end
n	New – clear VM memory and code	
b	Bye: PICO – reboot VM in disk mode, Windows - exist	
l	l - Load program from EEPROM (auto.bas)	
s	s - Save program to EEPROM (auto.bas)	

ee	ee - EEPROM erase – now command disabled	
ed	ed - EEPROM dump – show EEPROM content	
is 0|1	I2C scan devices (bus: 0 or 1); NOTE: 'is' stuck when a bus not terminated	is 0 is 1
t0|t1|t2	**Program Flow Tracing**: t0 – OFF; t1 - Step Mode; t2 – Cont Mode	t1|t2 - enable tracing mode
@N	**Pico Editor:** @N - delete N-th line	@3 – removes 3rd line of code
@N <code>	**Pico Editor:** @N <code> - insert <code> before N-th line	@4 cls – insert 'cls' before line 4

The BASIC language

Command	Description/Comments	Example
SM <entity> enable|disable	System Mode. Entity: **ESC**(default=enabled); ESC check; 'disable' boosts perf) **OLED**(default: auto-detected, if OLED exists: enabled)	sm esc disable
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)	1/2-dimensions integer/float array; 2-dimensions string array; 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"

CONSTANTS (build-in)

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

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-dimentions e.g. string a(2,5) – two 5-chars strings; string arrays are referred via single index a(0) then

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 2-dims string array; value=null set	string b(2,4) a(1)="no"

PROGRAM FLOW CONTROL

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

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 k = 5 TO 1 STEP -1 NEXT k END
WHILE lexpr [code] ENDWHILE	INT/FLOAT supported; nesting supported; cond: AND/OR/NOT supported;	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

BREAK	Can be used in FOR and WHILE loops.	while 1>0
		k=inkey()
		if(k!=0) then print k endif
		if(k==113) then break endif
		endwhile

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$(sexpr, expr)	Left part of the string	k$=LEFT$("abc", 2) + "123" i=12
RIGHT$(sexpr, expr)	Right part of the string	
MID$(sexpr,expr,expr)	Middle of the string	i$=MID$(STR$(i),2,3) PRINT i$
HEX$(expr)	Hex$(expr to hex string)	PRINT HEX$(NOT(0x0F))
STR$(expr)	Str$(expr to string)	
CHR$(expr)	Chr$(expr%256 to ascii e.g. 65 to ‘A’)	a=65 d$=chr$(a)
LEN(sexpr)	LEN(string length)	PRINT LEN("1234")->4
VAL(sexpr)	Val(string to value)	PRINT VAL("-1234")+1->-1233
ASC(sexpr)	ASC(ascii code of the 1st char of the string)	PRINT ASC(“AB”)->65
SIN(expr)	Sine	PRINT "SIN:",SIN(3.14/6)
COS(expr)	Cosine	PRINT "COS:",COS(3.14/6)
SQR(expr)	Square root	PRINT "SQRT:", SQR(5)
EXP(expr)	Expotential	PRINT "EXP:", EXP(1)
LOG(expr)	Logarithm	PRINT "LOG:", LOG(2.71)
SGN(expr)	Sign	PRINT "SGN:", SGN(-5)
ABS(expr)	Absolute	PRINT "ABS:", ABS(-5)
RND(max)	Hw-based rnd generator + von Neuman 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(expr)	QBASIC like	a=INT(1.1) b=INT(-1.1) PRINT a, ", ", b (1,-2)
FIX(expr)		c=FIX(1.9) d=FIX(-1.9) PRINT c, ", ", d (1,-1)
AND(expr,expr)		PRINT AND(0x3,0xF)
OR(expr,expr)		PRINT OR(0x3,0xF)
NOT(expr)		PRINT HEX$(NOT(0x0F))

PICO HARDWARE SUPPORT

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

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

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

TONE SUPPORT (experimental, tested on piezo)

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

I2C/UART/SPI SUPPORT (experimental)

-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: 1. Internal buffer size = 256 (max number of bytes to read/write); 2. SPI csn is controlled through gpio (dwrite)

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)
IREAD in, ad, arrr, len, nstop UREAD in, arr, len SREAD in, arr, len	read / write operations ad: i2c device address; arr: array name for read;	for k=0 to l-1 bi(k)=k next k uinit uart0, 0, 1, 9600 uwrite uart0, bi, l
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	pause 100 uread uart0, bo, l udeinit uart0
IDEINIT in; UDEINIT in; SDEINIT in	de-initialize hw	end
IREADABLE(in); IWRITABLE(in)	for non-blocking write / write	------

UREADABLE(in); UWRITABLE(in) *SREADABLE(in); SWRITABLE(in)*	*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*