# PUNYFORTH v0.5 GLOSSARY ------ for the ESP8266 Wifi module

The purpose of this glossary is to provide a sorted list of words, their stack effects, and a short description. It is recommended to go to the relevant source code to glean more information if necessary.

Stack notation shows input parameters with the rightmost being the top of the stack, input parameters on the left, output on the right, separated by --

Where a word has a compile time and a run time behaviour, the two stack effects are shown as <compile effect> ; <runtime effect>

e.g. ( a b -- c ) A word removes the top two stack items as input parameters, b is top of stack (tos). The word places c on the stack as the result

e.g. ( n -- ; -- adr ) During compilation this word removes n from the stack. During execution the word leaves an address on the stack

e.g ( <text> n -- ) This word expects another word it will consume from the input stream and n that it will pop off the stack as inputs

The Code Type column is coded as follows - C - Assembly language word or H - High level word

Located in File indicates in which source file the definition can be found

Data is referred to as bytes (8 bits), words (16 bits), longs (32 bits) and doubles (64 bits). The Punyforth stacks are 32 bit wide, so types other than longs are padded or split when placed on the stack

# CORE WORDS

This section lists all the core words in Punyforth. These are words that are always available in the dictionary to write programs with.

*NAME STACK CODE LOCATED DESCRIPTION*

*TYPE IN FILE*

## DATA STACK

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \_s0 | ( -- adr ) | C | words.s | Address of stack pointer |
| -rot | ( n1 n2 n3 -- n3 n1 n2 ) | C | ext.s | tos moved to 3rd entry |
| ?dup | ( n -- n n ) if n<>0 else ( n -- n ) | H | core.forth | if n <> 0, then duplicate n |
| 2drop | ( n1 n2 -- ) | C | ext.s | drop top 2 entries from stack |
| 2dup | ( n1 n2 -- n1 n2 n1 n2 ) | C | ext.s | duplicate top 2 entries from stack |
| 2over | ( n1 n2 n3 n4 -- n1 n2 n3 n4 n1 n2 ) | C | primitives.s | copy 3rd and 4th entry to tos |
| 2swap | ( n1 n2 n3 n4 -- n3 n4 n1 n2 ) | C | primitives.s | swap top two pairs |
| 3drop | ( n1 n2 n3 -- ) | H | core.forth | Remove the top 3 items on the stack |
| 3dup | ( n1 n2 n3 -- n1 n2 n3 n1 n2 n3 ) | H | core.forth | Duplicate the top 3 items on the stack |
| 4drop | ( n1 n2 n3 n4 -- ) | C | ext.s | drop top 4 entries from stack |
| depth | ( -- n ) | H | core.forth | returns the present depth of the statck |
| drop | (n -- ) | C | primitives.s | remove tos |
| dup | (n -- n n ) | C | primitives.s | copy tos |
| nip | ( n1 n2 -- n2 ) | C | words.s | Drop 2nd entry from stack |
| over | ( n1 n2 -- n1 n2 n1 ) | C | ext.s | copy 2nd entry to tos |
| rot | ( n1 n2 n3 -- n2 n3 n1 ) | C | primitives.s | rotate 3rd entry to tos |
| s0 | ( -- adr ) | H | core.forth | returns the address of the bottom of the stack |
| sp! | ( adr -- ) | C | primitives.s | set stack pointer to adr |
| sp@ | ( -- adr ) | C | primitives.s | read stack ptr adr |
| swap | (n1 n2 -- n2 n1 ) | C | primitives.s | swap top two entries |
| tuck | ( n1 n2 -- n2 n1 n2 ) | C | words.s | Copy tos under 2nd entry |

## RETURN STACK

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \_r0 | ( -- adr ) | C | words.s | returns address of return stack pointer |
| >r | ( n -- ) | C | primitives.s | pop tos and place on return stack |
| r@ | ( -- n ) | C | ext.s | copy top of rstack to tos |
| r> | ( -- n ) | C | primitives.s | pop n from return stack and place tos |
| r0 | ( -- adr ) | H | core.forth | returns the address of the bottom of the return stack |
| rdepth | ( -- n ) | H | core.forth | returns the present depth of the statck |
| rp! | ( adr -- ) | C | primitives.s | set return stack pointer to adr |
| rp@ | ( -- adr ) | C | primitives.s | read return stack ptr adr |

## LOGICAL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| and | ( n1 n2 -- n3 ) | C | primitives.s | n3 = n1 and n2 |
| invert | ( n1 -- n2 ) | C | primitives.s | n2 = not n1 - all bits inverted |
| nop | ( -- ) | H | core.forth | Do nothing |
| or | ( n1 n2 -- n3 ) | C | primitives.s | n3 = n1 or n2 |
| rshift | ( n shift -- ) | C | primitives.s | Shift n 'shift' places right |
| xor | ( n1 n2 -- n3 ) | C | primitives.s | n3 = n1 xor n2 |

## COMPARISON

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| < | ( n1 n2 -- flg ) | C | primitives.s | flg = true if n1 < n2 |
| <= | ( n1 n2 -- flg ) | C | primitives.s | flg = true if n2 less than or equal n1 |
| <> | ( n1 n2 -- flg ) | C | primitives.s | flg = true if n2 not equal n1 |
| = | ( n1 n2 -- flg ) | C | primitives.s | flg = true if n2 equal n1 |
| > | ( n1 n2 -- flg ) | C | primitives.s | flg = true if n1 > n2 |
| >= | ( n1 n2 -- flg ) | C | primitives.s | flg = true if n2 greater than or equal n1 |
| 0< | ( n1 -- flg ) | C | primitives.s | flg = true if n1 less than 0 |
| 0<> | ( n1 -- flg ) | C | primitives.s | flg = true if n1 not equal 0 |
| 0= | ( n1 -- flg ) | C | primitives.s | flg = true if n1 equal 0 |
| 0> | ( n1 -- flg ) | C | primitives.s | flg = true if n1 greater than 0 |
| 1= | ( n -- flg ) | C | words.s | flg = true of n=1 |

## MEMORY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -! | ( n addr -- ) | H | core.forth | decrement long at addr |
| ! | (n adr -- ) | C | primitives.s | store long b at adr |
| @ | ( adr -- n ) | C | primitives.s | read long n from address adr |
| +! | ( n addr -- ) | H | core.forth | increment long at addr |
| c! | ( b adr -- ) | C | primitives.s | store byte b at adr |
| c@ | ( adr -- b ) | C | primitives.s | read byte b from address adr |
| c+! | ( n addr -- ) | H | core.forth | increment byte or char at addr |
| cell | ( -- 4 ) | C | words.s | returns constant 4 |
| cells | ( n1 -- n2 ) | C | ext.s | n2 = n1 x 4 |
| cmove | ( src-addr dst-addr count -- ) | H | core.forth | Move count bytes from source to destination |
| freemem | ( -- n ) | H | core.forth | return free memory in bytes |
| heap-end | ( -- ) | C | words.s |  |
| heap-start | ( -- ) | C | words.s |  |
| heap? | ( a -- flg ) | H | core.forth |  |
| osfreemem | ( -- n ) | C | primitives.s | n = memory free for punyforth progam and data in bytes |
| usedmem | ( -- n ) | H | core.forth | return used memory in bytes |

## MATHS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| - | (n1 n2 -- n3 ) | C | primitives.s | n3 = n1 - n2 |
| \* | (n1 n2 -- n3 ) | C | primitives.s | n3 = n1 x n2 |
| / | ( n1 n2 -- quotient) | H | core.forth | n1 divided by n2, leaves quotient on stack |
| /mod | ( n1 n2 -- rem quot ) | C | primitives.s | quot = n1 / n2, remainder is rem |
| % | ( n1 n2 -- remainder ) | H | core.forth | n1 divided by n2, leaves remainder on stack |
| + | (n1 n2 -- n3 ) | C | primitives.s | n3 = n1 + n2 |
| 1- | ( n1 -- n2 ) | C | ext.s | n2 = n1-1 |
| 1+ | ( n1 -- n2 ) | C | ext.s | n2 = n1+1 |
| abs | ( n1 -- n2 ) | H | core.forth | n2 = |n1| |
| between? | ( min n2 max -- flg ) | H | core.forth | flg=true if inclusively between max, min |
| max | ( n1 n2 -- n3 ) | H | core.forth | n3 = unsigned largest of n1, n2 |
| min | ( n1 n2 -- n3 ) | H | core.forth | n3 = unsigned smallest of n1, n2 |
| random | ( -- n ) | C | ext.s | n = random number long |

## CONVERSION

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| >number | ( str len -- number flg ) | C | words.s | convert string to decimal number and flg=true if successful |
| hex>int | ( str -- n | throws:ECONVERT ) | H | core.forth | hex string str is converted to a long, else throws an ECONVERT exception |
| hex>int' | ( str len -- n | throws:ECONVERT ) | H | core.forth | part of hex>int |
| hexchar>int | ( char -- n | throws:ECONVERT ) | H | core.forth |  |

## LOOPING

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ?do | ( count start -- ) immediate | H | core.forth | start a counted loop, if count <> start, else don't do the loop contents |
| +loop | ( -- ) | H | core.forth |  |
| bounds | ( start len -- limit start ) | H | core.forth | converts start len parameters to limit start for the use by 'do' |
| do | ( count start -- ) immediate | H | core.forth | start a counted loop structure |
| end? | ( incr -- flg ) | H | core.forth |  |
| i | ( -- n ) | C | primitives.s | n = do loop count |
| j | ( -- n ) | C | primitives.s | n = next outer loop count |
| loop | ( -- ) immediate | H | core.forth | ends a counted loop structure |
| unloop | ( -- ) | H | core.forth | r> r> r> 2drop >r |

Examples:-

: test 10 0 do i . loop ; running test we display 0123456789

: test 10 0 do i . 2 +loop ; running test we display 02468

## CONDITIONAL BRANCH & LOOPING

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| again | immediate | H | forth.core |  |
| begin | immediate | H | forth.core |  |
| branch | ( -- ) | C | primitives.s |  |
| branch0 | ( n -- ) | C | primitives.s |  |
| else | ( -- ) immediate | H | forth.core |  |
| if | ( flg -- ) immediate | H | forth.core |  |
| repeat | ( -- ) immediate | H | forth.core |  |
| then | ( -- ) immediate | H | forth.core |  |
| until | ( flg -- ) immediate | H | forth.core |  |
| while | ( flg -- ) immediate | H | forth.core |  |

Examples:

|  |  |
| --- | --- |
| if <words to execute if condition true> then | begin <more words> while <more words> repeat |
| if <words if condition true> else <words if condition false> then | begin <more words> until |
|  | begin <more words> again - an endless loop |

## CASE STATEMENTS

CASE statements are constructed in a manner similar to C using SWITCH, CASE, and BREAK.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| case | ( -- branch-counter ) immediate | H | core.forth |  |
| endcase | ( #branches #branchesi\*a -- ) immediate | H | core.forth |  |
| endof | ( -- ) immediate | H | core.forth |  |
| of | ( n -- ) immediate | H | core.forth |  |

CASE example:-

|  |  |
| --- | --- |
| : day ( n -- )  case  1 of print: "Monday" endof  2 of print: "Tuesday" endof  3 of print: "Wednesday" endof  4 of print: "Thursday" endof  5 of print: "Friday" endof  6 of print: "Saturday" endof  7 of print: "Sunday" endof  print: "Unknown day: " .  endcase ; |  |

## VECTORED EXECUTION, COMBINATORS and QUOTATIONS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bi | ( a xt1 xt2 -- xt1.a xt2.a ) | H | core.forth | Applies quotation p to x, then applies quotation q to x. |
| bi@ | ( a b xt -- xt.a xt.b ) | H | core.forth | Applies the quotation to x, then to y. |
| bi\* | ( a b xt1 xt2 -- xt1.a xt2.b ) | H | core.forth | Applies the quotation to x, then to y. |
| dip | ( a xt -- a ) | H | core.forth | calls a quotation while temporarily hiding the tos |
| execute | ( adr -- ) | C | primitives.s | Execute word at adr |
| keep | (a xt -- xt.a a ) | H | core.forth | calls a quotation with an item on the stack, restoring that item after the quotation returns. |
| { | immediate | H | core.forth | start a quotation - a headless Punyforth word-within-a-word |
| } | immediate | H | core.forth | end a quotation |

## ERROR MANAGEMENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| catch | ( xt -- exception | 0 ) | H | core.forth |  |
| EASSERT |  | H | core.forth | exception type |
| ECONVERT |  | H | core.forth | exception type |
| EESCAPE |  | H | core.forth | exception type |
| ENOTFOUND |  | H | core.forth | exception type |
| EOVERFLOW |  | H | core.forth | exception type |
| eundef | ( -- ) | C | words.s | error word undefined ?? |
| eundefc | ( -- ) | C | words.s |  |
| eundefi | ( -- ) | C | words.s |  |
| EUNDERFLOW |  | H | core.forth | exception type |
| ex-type | ( exception -- ) | H | core.forth |  |
| exception: | ( <name> -- ) | H | core.forth |  |
| throw | ( i\*x exception -- i\*x exception | 0 ) | H | core.forth |  |
| traceback | ( code -- ) | H | core.forth |  |

## START-UP / SHUT DOWN / SYSTEM STATE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| abort | ( -- ) | C | primitives.s | call forth abort |
| deep-sleep | ( a2 -- ) | C | primitives.s | place cpu in deep-sleep |
| os-enter-critical | ( -- ) | C | primitives.s |  |
| os-exit-critical | ( -- ) | C | primitives.s |  |
| state | ( -- adr ) | C | words.s | address of compiler state flag, state @ returns true if compiling |
| task-yield | ( -- ) | C | primitives.s |  |
| xpause | ( -- ) | C | ext.s |  |

## I/O PORTS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| adcread | ( -- n ) | C | ext.s | read the analogue input |
| gpio-mode | ( direction num -- ) | C | ext.s |  |
| gpio-read | ( gpionum - n ) | C | ext.s |  |
| gpio-set-interrupt | ( inttype gpionum -- ) | C | ext.s |  |
| gpio-write | ( bool gpionum -- ) | C | ext.s |  |
| pulse-in | ( pin state timeout -- pulselen ) | C | ext.s | returns pulse input length in uS |
| pwm-duty | ( duty -- ) | C | ext.s | duty is 16 bits |
| pwm-freq | ( freq -- ) | C | ext.s | freq is 16 bits |
| pwm-init | ( pinsarray numberofpins -- ) | C | ext.s |  |
| pwm-start | ( -- ) | C | ext.s |  |
| pwm-stop | ( -- ) | C | ext.s |  |
| uart-set-bps | (uartnum bps -- ) | C | ext.s |  |

## TIMING and FREQUENCY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| cpufreq! | ( freq -- ) | C | ext.s | set cpu frequency (MHz) |
| cpufreq@ | ( -- freq ) | C | ext.s | read cpu frequency (MHz) |
| ms | ( n -- ) | C | ext.s | wait n milliseconds |
| ms@ | ( -- n ) | C | ext.s | get system time in milliseconds |
| us | ( n -- ) | C | ext.s | wait n uS |
| us@ | ( -- n ) | C | ext.s | get system time in uS |
| wait-event | ( delayms eventbuffer -- n ) | C | ext.s |  |

## DEFINITIONS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| : | ( <word> -- ) | C | words.s | start forth definition |
| ' | ( -- xt | throws:ENOTFOUND ) | H | core.forth | find the xt of the next word in the inputstream |
| ['] | ( -- ) | C | primitives.s | compile only |
| ['], |  | H | core.forth |  |
| ] | ( -- ) | C | words.s | resume compilation mode |
| allot | ( n -- ) | C | words.s | allot n bytes of memory as part of a definition |
| backref, | ( n -- ) | H | core.forth | n is stored at here-1, here is unchanged |
| create: |  | H | core.forth |  |
| createheader | ( <name> -- ) | C | words.s |  |
| defer: | ( <name> -- ) | H | core.forth |  |
| defer! | ( dst-xt src-xt -- ) | H | core.forth |  |
| does> |  | H | core.forth |  |
| exit | ( -- ) | C | primitives.s | pop the forth PC from the return stack |
| handler |  | H | core.forth | defer type |
| interpret? | ( -- flg ) | H | core.forth | flg = true if currently interpreting |
| is: | immediate | H | core.forth |  |
| lastword | ( -- adr ) | C | words.s |  |
| override | ( <word> -- ) immediate | H | core.forth | used to refer to the previous word of the same name. Punyforth otherwise defaults to recursion |
| postpone: | ( -- | throws:ENOTFOUND ) | H | core.forth | force compile semantics of an immediate word |
| prepare-forward-ref | ( -- a ) | H | core.forth | here 0 , |
| resolve-forward-ref | ( a -- ) | H | core.forth | here over - swap ! |
| unhandled |  | H | core.forth | defer type |
| var-lastword | ( -- adr ) | C | words.s |  |

## RECOGNISERS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \_ | ( addr len -- ? ) | H | core.forth |  |
| chr? |  | H | core.forth |  |
| hex? |  | H | core.forth |  |
| str, | ( len -- ) | H | core.forth |  |
| str? |  | H | core.forth |  |

## COMMENTS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ( | ( -- ) | H | core.forth | Start a comment, must finish with ) on same line |
| \ | ( -- ) | H | core.forth | The rest of the line is ignored as a comment |

## COMPILE LITERALS

Bytes, words, and longs may be compiled directly into code memory usually for building fixed tables.

These cannot be used inside a definition as any preceding literal would have already been compiled as a literal.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| , | ( n -- ) | C | words.s | compile a long as used in building tables e.g. 1 , 2 , 3 , |
| c, | ( b -- ) | C | words.s | compile a byte or char as used in building tables e.g. 1 c, 2 c, 3 c, |
| c,-until | (separator -- ) | H | core.forth |  |
| literal | ( n -- ) | C | words.s | compile tos into the word e.g. [ 4 5 + ] literal |

## DEBUG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| help | ( -- ) | H | core.forth | List all words in the dictionary |
| stack-hide | ( -- ) | H | core.forth | Set the user prompt to nothing, no stack display |
| stack-print | ( -- ) | H | core.forth | Print the data stack contents |
| stack-show | ( -- ) | H | core.forth | Set the user prompt ot show the stack contents |

## CONSTANTS and VARIABLES

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| array: | ( size <name> -- ; index -- addr ) | H | core.forth | define an array of longs e.g. 10 array: myname |
| buffer: | ( size <name> -- ; -- addr ) | H | core.forth | define a block of memory as a buffer |
| byte-array: | ( size <name> -- ; index -- addr ) | H | core.forth | define an array of bytes e.g. 10 array: myname |
| constant: | ( <name> n -- ; -- n ) ) | C | words.s | define a constant value or string e.g. 3 constant: mynumber e.g. "Hello" constant: mystring |
| FALSE | ( -- 0 ) | H | core.forth |  |
| field: | ( <name> -- ) | H | core.forth | add a variable to a data structure |
| init-variable: | ( <name> n -- ; -- adr ) | C | words.s | defines a variable with a preset value e.g. 23 init-variable: myinitvar |
| single-handler | ( -- adr ) | H | core.forth | single threaded global handler |
| struct | ( -- 0 ) | H | core.forth | begin definition of a data structure |
| TRUE | ( -- -1 ) | H | core.forth |  |
| var-handler | ( -- adr ) | H | core.forth | stores the address of the nearest exception handler |
| variable: | ( <name> -- ) | H | core.forth | Create a new long, initialised to 0 |

## Private Words

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CALIGN | ( -- ) | X | PRI | long align code memory |

## STRINGS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| =str | ( str1 str 2 -- flg ) | H | core.forth | flg=true if str1 = str2 |
| compare | ( str1 len1 str2 len 2 -- flg ) | C | words.s | flg = true id str1 = str2 |
| str-in? | ( str substr -- flg ) | H | core.forth | flg=true if substr is found within str |
| str-starts? | ( str substr -- flg ) | H | core.forth | flg=true if str starts with substr |
| strlen | ( str -- len ) | H | core.forth | len is the length of string str |
| word | ( <word> -- str len ) | C | words.s | converts next word in input stream to string |

Punyforth strings are null terminated and are defined without leading space e.g. "Hello world".

Thus a string constant is "This is a fine day" constant: mystring

## STREAMING I/O

From start-up Punyforth is set 115200 baud 8 bit, no parity, 1 stop bit. N.B. The line terminator is cr-lf and a local echo is necessary

Character based devices such as serial, VGA, LCD etc are treated as streaming I/O where the device code automatically detects and handles special characters. EMIT words will send a single character via the currently selected output device. Conversely KEY is the input from the device.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \_emit | ( b -- ) | C | primitives.s | part of emit |
| ? | ( adr -- ) | H | core.forth | Display long at adr with no formatting |
| . | ( n -- ) | H | core.forth | Display n with no formatting |
| #tib | ( -- adr ) | C | words.s | returns addr of the input stream char counter. so #tib @ returns the number of chars |
| chr>in | ( chr -- ) | C | words.s | inject chr into input stream |
| cr | ( -- ) | H | core.forth | Emit a carriage return character |
| crlf? | ( chr1 chr2 -- flg ) | H | core.forth | flg=true if chr2=10 and chr1=13 |
| emit | ( chr -- ) | C | words.s | display chr |
| eschr | ( char -- char ) | H | core.forth | read next char from stdin |
| key | ( -- ch | false ) | C | words.s | read chr from input stream, waits if none available |
| print: | ( <words within ""> -- ) | H | core.forth | Print an inline string e.g. print: "hello world" |
| println: | ( <words within ""> -- ) | H | core.forth | Like print: but adds a cr |
| prompt | ( -- adr ) |  |  | return address of user prompt string |
| readchar |  | C | ext.s |  |
| readchar-nowait | ( -- chr | false ) | C | ext.s | read next char in input stream. If none available returns false |
| readchar-wait | ( -- chr ) | C | ext.s | wait indefinitely for next char on input stream |
| separator | ( -- chr ) | H | core.forth | Consume input stream until non-whitespace character chr |
| show\_prompt | ( -- ) |  |  | enable the user prompt |
| space | ( -- ) | H | core.forth | Emit a space return character |
| tib | ( -- adr ) | C | words.s | returns addr of input stream char buffer |
| type | ( asciiz -- ) | C | words.s | display 0 terminated string |
| type-counted | ( addr len -- ) | C | words.s | display counted string |
| whitespace? | (chr -- flg ) | H | core.forth | Is chr any of the whitespace characters? |
| xemit | ( addr -- ) | C | words.s | pointer to current 'emit' word, so that emit can be vectored |
| xtype | ( addr -- ) | C | words.s | pointer to current 'type' word, so that type can be vectored |

## DICTIONARY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| compile-time |  | C | words.s |  |
| find | ( str len -- link | false ) | C | words.s | Find str in dictionary and return link else false |
| here | ( -- adr ) | C | words.s | returns address of end of dictionary space |
| hidden? | ( link -- flg ) | C | words.s |  |
| hide | ( link -- ) | C | words.s |  |
| immediate | ( -- ) | C | words.s | Mark the last word defined as executing at compile time |
| immediate? | ( link -- flg ) | C | words.s |  |
| link>body | ( adr1 -- adr2 ) | C | words.s |  |
| link>flags | ( adr -- flags ) | C | words.s |  |
| link>flb | ( adr1 -- adr2 ) | C | words.s |  |
| link>len | ( adr -- n ) | C | words.s |  |
| link>name | ( adr1 -- adr2 ) | C | words.s |  |
| link>xt | ( adr1 -- adr2 ) | C | words.s |  |
| marker: | ( <name> -- ) | H | core.forth | Defines a module start marker. When this word is executed, all words defined after this marker are forgotten |
| reveal | ( link -- ) | C | words.s |  |

## I2C BUS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| i2c-init | ( bus sclpin sdapin freq -- result ) | C | ext.s |  |
| i2c-read | ( bus ack -- data ) | C | ext.s |  |
| i2c-read-slave | ( bus slaveaddr data buffer len -- result ) | C | ext.s |  |
| i2c-start | ( bus -- ) | C | ext.s |  |
| i2c-stop | ( bus -- flg ) | C | ext.s |  |
| i2c-write | ( bus byte -- flg ) | C | ext.s |  |
| i2c-write-slave | ( bus slaveaddr data buffer len -- result ) | C | ext.s |  |

## SPI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| spi-init | ( bus mode freqdiv msb endyness minimalpins -- n ) | C | ext.s |  |
| spi-send | ( bus outdata indata datasize wordsize -- n ) | C | ext.s |  |
| spi-send8 | ( bus data -- n ) | C | ext.s |  |

## EEPROM

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| /end | ( -- ) | C | ext.s | Used in EEPROM resident source to signal end of file - stop reading from the file |
| erase-flash | ( sector -- n ) | C | ext.s |  |
| load | ( blocknum -- ) | C | ext.s |  |
| loading? | ( -- flg ) | C | ext.s |  |
| read-flash | ( addr buffer size -- n ) | C | ext.s |  |
| write-flash | ( addr buffer size -- n ) | C | ext.s |  |

## WIFI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| dhcpd-start | (1stclientip maxleases -- ) | C | ext.s |  |
| dhcpd-stop | ( -- ) | C | ext.s |  |
| wifi-ip-str | ( interface buffer buffsize -- ) | C | ext.s | interface 0=station 1=softap |
| wifi-set-mode | ( mode -- flg ) | C | ext.s |  |
| wifi-set-softap-config | ( ssid password authmode hidden channels maxconnections -- flg ) | C | ext.s |  |
| wifi-set-station-config | (ssid password -- flg ) | C | ext.s |  |
| wifi-softap-start | ( -- flg ) | C | ext.s |  |
| wifi-station-connect | ( -- flg ) | C | ext.s |  |
| wifi-station-disconnect | ( -- flg ) | C | ext.s |  |
| wifi-station-start | ( -- flg ) | C | ext.s |  |
| wifi-stop | ( -- ) | C | ext.s |  |

## NETCON

HTTP and UDP communication:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| netbuf-data | ( netbuf -- buffer size ) | C | ext.s |  |
| netbuf-del | ( netbuf -- ) | C | ext.s |  |
| netbuf-next | ( netbuf -- n ) | C | ext.s |  |
| netcon-accept | ( netcon -- netcon err\_t ) | C | ext.s |  |
| netcon-bind | ( conn host port -- n ) | C | ext.s |  |
| netcon-close | ( conn -- ) | C | ext.s |  |
| netcon-connect | ( conn host port -- n ) | C | ext.s |  |
| netcon-delete | ( conn -- ) | C | ext.s |  |
| netcon-listen | ( netcon -- n ) | C | ext.s |  |
| netcon-new | ( contype -- n ) | C | ext.s |  |
| netcon-read-timeout | ( netcon timeoutsec -- ) | C | ext.s |  |
| netcon-read-timeout@ | ( netcon - timeoutsec ) | C | ext.s |  |
| netcon-recv | ( netcon -- netbuf err\_t ) | C | ext.s |  |
| netcon-recvinto | ( conn buffer size -- countread err\_t ) | C | ext.s |  |
| netcon-send | ( conn data len -- n ) | C | ext.s |  |
| netcon-set recvtimeout | ( conn recvtimeoutms -- ) | C | ext.s |  |
| netcon-write | ( conn data size -- n ) | C | ext.s |  |

**N.B port is type long, host is type string "192.168.1.8" or the name e.g. "Bob-PC"**

## APPLICATIONS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ws2612set | ( gpionum rgb -- ) | C | ext.s |  |
| ws2812rgb | ( gpionum rgb -- ) | C | ext.s |  |

## Unsorted words

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| \_type | ( string -- ) | C | ext.s |  |
| [str | ( -- forward-ref ) | H | core.forth |  |
| >in | ( -- adr ) | C | words.s |  |
| >s' | ( ? addr n -- addr2 ? ) | H | core.forth |  |
| >str | ( addr n -- ) | H | core.forth |  |
| align | ( -- ) | C | words.s |  |
| align! | ( -- ) | C | words.s |  |
| dp | ( -- ) | C | words.s |  |
| entercol |  | C | words.s |  |
| enterdoes |  | C | words.s |  |
| eundef | ( -- ) | C | words.s |  |
| link-type | ( link -- ) | H | core.forth |  |
| push-enter | ( -- ) | C | ext.s |  |
| str] | ( forward-ref -- ) | H | core.forth |  |
| var-dp | ( -- ) | C | words.s |  |

# SOURCE CODE LIBRARIES on EEPROM

The following words are located in source files stored on the EEPROM. This is to economise on space in a user application, in that libraries of words that aren't used don't take up precious space in the ram. A library can be loaded into the dictionary by:-

<library name> load e.g. WIFI load

Some libraries are dependant on others, in which case the dependancies are automatically loaded.

*NAME STACK CODE LOCATED DESCRIPTION*

*TYPE IN FILE*

## DHT22

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| bit-at | ( i -- bit ) | H | dht22.forth |  |
| bits | ( -- adr ) | H | dht22.forth | 40 byte array |
| bytes | ( -- adr ) | H | dht22.forth | 5 bytes array |
| bytes-clear | ( -- ) | H | dht22.forth |  |
| checksum | ( -- ) | H | dht22.forth |  |
| convert | ( lsbyte msbyte -- value ) | H | dht22.forth |  |
| dht-measure | ( -- humidity temperature ) | H | dht22.forth | measures temperature and humidity using DHT22 sensor  temperature and humidity values are multiplied with 10 |
| dht-pin | ( -- gpiopin ) | H | dht22.forth | read pin used by dht22 |
| dht-pin! | ( gpiopin -- ) | H | dht22.forth | set pin used by dht22 |
| ECHECKSUM |  | H | dht22.forth | exception |
| fetch | ( -- ) | H | dht22.forth | high pulse for 26-28 us is bit0, high pulse for 70 us is bit1 |
| humidity | ( -- humidity%-x-10 ) | H | dht22.forth |  |
| init | ( -- ) | H | dht22.forth |  |
| measure | ( -- ) | H | dht22.forth |  |
| process | ( -- ) | H | dht22.forth |  |
| temperature | ( -- celsius-x-10 ) | H | dht22.forth |  |
| validate | ( -- | throws:ECHECKSUM ) | H | dht22.forth |  |
| var-dht-pin | ( -- adr ) | H | dht22.forth | 1 long, initial value 2 , var-dht-pin \ default D4, wemos d1 mini dht22 shield, use dht-pin! to override |

## EVENT

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Event | ( -- n ) | H | event.forth | Record containing .type .ms .us .payload |
| event-timeout | ( -- adr ) | H | event.forth | 1 long variable, initial value 70 |
| EVT\_GPIO | ( -- 100 ) | H | event.forth | 1 long constant |
| next-event | ( eventstruct -- event ) | H | event.forth |  |

## FLASH

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| b | ( y -- ) | H | flash.forth | block editor command - blank row |
| block | ( block# -- addr ) | H | flash.forth |  |
| buf | ( -- adr ) | H | flash.forth | buffer, SIZE long |
| c | ( -- ) | H | flash.forth | block editor command - clear screen |
| ch | ( y x -- adr ) | H | flash.forth |  |
| check | ( code -- | flag ) | H | flash.forth | flag = 0=OK,1=ERR,2=TIMEOUT,3=UNKNOWN |
| COLS | ( -- 128 ) | H | flash.forth | 1 long constant |
| copy-row | ( dsty srcy -- ) | H | flash.forth |  |
| d | ( y -- ) | H | flash.forth | block editor command - delete row |
| dirty | ( -- adr ) | H | flash.forth | 1 long variable, initial value FALSE |
| EBLOCK |  | H | flash.forth | exception |
| list | ( block# -- ) | H | flash.forth |  |
| offs | ( -- adr ) | H | flash.forth | 1 long variable |
| p | ( y -- ) | H | flash.forth | block editor command - prepend empty row before row y |
| r | ( y <line> -- ) | H | flash.forth | block editor command - overwrite row |
| row | ( y -- adr ) | H | flash.forth |  |
| ROWS | ( -- 32 ) | H | flash.forth | 1 long constant |
| SIZE | ( -- 4096 ) | H | flash.forth | 1 long constant |
| type# | ( y -- ) | H | flash.forth |  |

## FONT57

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| font5x7 | ( -- ) | H | font5x7.forth | large data table containing the font patterns |

## GPIO

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| GPIO\_INTTYPE-NONE | ( -- 0 ) | H | gpio.forth | 1 long constant - gpio interrupt type |
| GPIO\_INTTYPE\_EDGE\_POS | ( -- 1 ) | H | gpio.forth | 1 long constant - gpio interrupt type |
| GPIO\_INTTYPE\_EDGE\_NEG | ( -- 2 ) | H | gpio.forth | 1 long constant - gpio interrupt type |
| GPIO\_INTTYPE\_EDGE\_ANY | ( -- 3 ) | H | gpio.forth | 1 long constant - gpio interrupt type |
| GPIO\_INTTYPE\_LEVEL\_LOW | ( -- 4 ) | H | gpio.forth | 1 long constant - gpio interrupt type |
| GPIO\_INTTYPE\_LEVEL\_HIGH | ( -- 5 ) | H | gpio.forth | 1 long constant - gpio interrupt type |
| GPIO\_IN | ( -- 1 ) | H | gpio.forth | 1 long constant - gpio modes |
| GPIO\_OUT | ( -- 2 ) | H | gpio.forth | 1 long constant - gpio modes |
| GPIO\_OUT\_OPEN\_DRAIN | ( -- 3 ) | H | gpio.forth | 1 long constant - gpio modes |
| GPIO\_HIGH | ( -- 1) | H | gpio.forth | 1 long constant - gpio values |
| GPIO-LOW | ( -- 0 ) | H | gpio.forth | 1 long constant - gpio values |
| blink | ( pin -- ) | H | gpio.forth | switch pin on and off once, 0.5s period |
| times-blink | ( pin ntimes -- ) | H | gpio.forth | blink pin n times |
| ENOPULSE |  | H | gpio.forth | exception |

## NETCON

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| UDP | ( -- 1 ) | H | netcon.forth | 1 long constant |
| TCP | ( -- 2 ) | H | netcon.forth | 1 long constant |
| RECV\_TIMEOUT\_MSEC | ( -- 70 ) | H | netcon.forth | 1 long constant |
| ENETCON |  | H | netcon.forth | exception |
| ERTIMEOUT |  | H | netcon.forth | exception |
| NC\_ERR\_TIMEOUT | ( -- -3 ) | H | netcon.forth | 1 long constant - netcon error |
| NC\_ERR\_CLSD | ( -- -15 ) | H | netcon.forth | 1 long constant - netcon error |
| netcon-new | ( type -- netcon | throws:ENETCON ) | H | netcon.forth |  |
| check | ( errcode -- )( errcode -- | throws:ENETCON ) | H | netcon.forth |  |
| netcon-connect | ( port host type -- netcon | throws:ENETCON ) | H | netcon.forth | Connect to a remote port/ip. Must be used in both TCP and UDP case |
| netcon-bind | ( port host netcon -- | throws:ENETCON ) | H | netcon.forth |  |
| netcon-listen | ( netcon -- | throws:ENETCON ) | H | netcon.forth |  |
| netcon-tcp-server | ( port host -- netcon | throws:ENETCON ) | H | netcon.forth | Create a TCP server by binding a connection to the given port host. Leaves a netcon connection associated to the server socket on the stack. |
| netcon-udp-server | ( port host -- netcon | throws:ENETCON ) | H | netcon.forth | Create a UDP server by binding a connection to the given port host. Leaves a netcon connection associated to the server socket on the stack. |
| netcon-accept | ( netcon -- new-netcon | throws:ENETCON ) | H | netcon.forth | Accept an incoming connection on a listening TCP connection. Leaves a new netcon connection that is associated to the client socket on the stack. |
| netcon-send-buf | ( netcon buffer len -- | throws:ENETCON ) | H | netcon.forth | Write the content of the given buffer to a UDP socket |
| netcon-write-buf | ( netcon buffer len -- | throws:ENETCON ) | H | netcon.forth | Write the content of the given buffer to a TCP socket |
| netcon-write | ( netcon str -- | throws:ENETCON ) | H | netcon.forth | Write a null terminated string to a TCP socket |
| netcon-writeln | ( netcon str -- | throws:ENETCON ) | H | netcon.forth | Write a null terminated string then a CRLF to a TCP socket |
| read-ungreedy | ( size buffer netcon -- count code | throws:ERTIMEOUT ) | H | netcon.forth |  |
| netcon-read | ( netcon size buffer -- count | -1 | throws:ENETCON/ERTIMEOUT ) | H | netcon.forth | Read maximum `size` amount of bytes into the buffer. Leaves the amount of bytes read on the top of the stack, or -1 if the connection was closed. |
| netcon-readln | ( netcon size buffer -- count | -1 | throws:ENETCON/EOVERFLOW/ERTIMEOUT ) | H | netcon.forth | Read one line into the given buffer. The line terminator is CRLF. Leaves the length of the line on the top of the stack, or -1 if the connection was closed. If the given buffer is not large enough to hold EOVERFLOW is thrown. |
| netcon-dispose | ( netcon -- ) | H | netcon.forth | Close then dispose the given socket. |

**N.B port is type long, host is type string "192.168.1.8" or the name e.g. "Bob-PC"**

## NTP - Network Time Protocol

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ENTP |  | H | ntp.forth | exception |
| con | ( -- adr ) | H | ntp.forth | 1 long variable |
| SIZE | ( -- 48 ) | H | ntp.forth | 1 long constant |
| packet | ( -- adr ) | H | ntp.forth | byte array, SIZE in size |
| request | ( -- buffer ) | H | ntp.forth |  |
| connect | ( port host -- ) | H | ntp.forth |  |
| send | ( -- ) | H | ntp.forth |  |
| receive | ( -- #bytes ) | H | ntp.forth |  |
| dispose | ( -- ) | H | ntp.forth |  |
| ask | ( port host -- #bytes ) | H | ntp.forth |  |
| parse | ( -- ) | H | ntp.forth |  |
| network-time | ( port host -- seconds-since-1970 | throws:ENTP ) | H | ntp.forth |  |

Example - this prints the number of seconds since start of 1970 until the ESC key is pressed

: test begin 123 "192.168.1.1" network-time . cr 1000 ms readchar-nowait 27 = until ;

## PING - ultrasound distance measurement

Measures the pulse generated by ultrasonic ranging module (tested with: HC-SR04 sensors)

Works the following way:

(1) Using IO trigger for at least 10us high level signal,

(2) The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.

(3) IF the signal comes back, time of high output IO duration is the time from sending ultrasonic to returning.

Distance = (high level time×velocity of sound (340M/S) / 2,

Usage example: PIN\_ECHO 100 cm>timeout PIN\_TRIGGER ping pulse>cm

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| emit-pulse | ( trigger-pin -- ) | H | ping.forth |  |
| listen-echo | ( echo-pin timeout-us -- ms ) | H | ping.forth |  |
| ping | ( echo-pin timeout-us trigger-pin -- pulse-duration-us ) | H | ping.forth |  |
| cm>timeout | ( cm -- us ) | H | ping.forth |  |
| inch>timeout | ( inch -- us ) | H | ping.forth |  |
| pulse>cm | ( us -- cm ) | H | ping.forth |  |
| pulse>inch | ( us -- inch ) | H | ping.forth |  |

## SONOFF Smart Power Socket

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| RELAY | ( -- 12 ) | H | sonoff.forth | 1 long constant |
| relay-state | ( -- adr ) | H | sonoff.forth | 1 long variable, initial value FALSE |
| on | ( -- ) | H | sonoff.forth |  |
| off | ( -- ) | H | sonoff.forth |  |
| toggle | ( -- ) | H | sonoff.forth |  |
| LED | ( -- 13 ) | H | sonoff.forth | 1 long constant |
| led-on | ( -- ) | H | sonoff.forth |  |
| led-off | ( -- ) | H | sonoff.forth |  |
| flash | ( n -- ) | H | sonoff.forth |  |
| alert | ( -- ) | H | sonoff.forth | flash LED 10 times |

## SSD1306I2C

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| WIDTH | ( -- 64 ) | H | ssd1306-i2c.forth | 1 long constant |
| HEIGHT | ( -- 48 ) | H | ssd1306-i2c.forth | 1 long constant |
| SCL | ( -- 5 ) | H | ssd1306-i2c.forth | 1 long constant |
| SDA | ( -- 4 ) | H | ssd1306-i2c.forth | 1 long constant |
| RST | ( -- 0 ) | H | ssd1306-i2c.forth | 1 long constant |
| SLAVE | ( -- 16r3C ) | H | ssd1306-i2c.forth | 1 long constant |
| BUS | ( -- 0 ) | H | ssd1306-i2c.forth | 1 long constant |
| FREQ | ( -- 2 ) | H | ssd1306-i2c.forth | 1 long constant, 400kHz i2c speed |
| SIZE | ( -- WIDTH HEIGHT \* 8 / ) | H | ssd1306-i2c.forth | 1 long constant |
| screen1 | ( -- adr ) | H | ssd1306-i2c.forth | buffer |
| screen | ( -- buffer ) | H | ssd1306-i2c.forth |  |
| EI2C |  | H | ssd1306-i2c.forth | exception |
| wire | ( -- ) | H | ssd1306-i2c.forth |  |
| check | ( code -- | throws:EI2C ) | H | ssd1306-i2c.forth |  |
| buf | ( -- adr ) | H | ssd1306-i2c.forth | two byte table |
| cmd | ( byte -- | throws:EI2C ) | H | ssd1306-i2c.forth |  |
| reset | ( -- ) | H | ssd1306-i2c.forth |  |
| init | ( -- ) | H | ssd1306-i2c.forth |  |
| width\* | immediate | H | ssd1306-i2c.forth |  |
| clampx | immediate | H | ssd1306-i2c.forth |  |
| clampy | immediate | H | ssd1306-i2c.forth |  |
| clamp | ( x y -- x' y' ) | H | ssd1306-i2c.forth |  |
| y>bitmask | ( y -- bit-index ) | H | ssd1306-i2c.forth |  |
| xy>i | ( x y -- bit-mask buffer-index ) | H | ssd1306-i2c.forth |  |
| or! | ( value addr -- ) | H | ssd1306-i2c.forth |  |
| and! | ( value addr -- ) | H | ssd1306-i2c.forth |  |
| set-pixel | ( x y -- ) | H | ssd1306-i2c.forth |  |
| unset-pixel | ( x y -- ) | H | ssd1306-i2c.forth |  |
| pixel-set? | ( x y -- flg) | H | ssd1306-i2c.forth |  |
| hline | ( x y width -- ) | H | ssd1306-i2c.forth |  |
| rect-fill | ( x y width height -- ) | H | ssd1306-i2c.forth |  |
| fill-buffer | ( value -- ) | H | ssd1306-i2c.forth |  |
| c1 | ( -- n ) | H | ssd1306-i2c.forth |  |
| c2 | ( -- n ) | H | ssd1306-i2c.forth |  |
| display | ( -- ) | H | ssd1306-i2c.forth |  |
| display-clear | ( -- ) | H | ssd1306-i2c.forth |  |
| bus-init | ( -- ) | H | ssd1306-i2c.forth |  |
| display-init | ( -- | throws:ESSD1306 ) | H | ssd1306-i2c.forth |  |
| font | ( -- adr ) | H | ssd1306-i2c.forth | 1 long variable, initial value is 0 |
| text-left | ( -- adr ) | H | ssd1306-i2c.forth | 1 long variable, initial value is 0 |
| text-top | ( -- adr ) | H | ssd1306-i2c.forth | 1 long variable, initial value is 0 |
| font-size | ( -- adr ) | H | ssd1306-i2c.forth | 1 long variable, initial value is 1 |
| font-small | ( -- ) | H | ssd1306-i2c.forth |  |
| font-medium | ( -- ) | H | ssd1306-i2c.forth |  |
| font-big | ( -- ) | H | ssd1306-i2c.forth |  |
| font-xbig | ( -- ) | H | ssd1306-i2c.forth |  |
| draw-lf | ( -- ) | H | ssd1306-i2c.forth |  |
| draw-cr | ( -- ) | H | ssd1306-i2c.forth |  |
| dot | ( x y -- ) | H | ssd1306-i2c.forth |  |
| stripe | ( bits -- ) | H | ssd1306-i2c.forth |  |
| draw-char | ( char -- ) | H | ssd1306-i2c.forth |  |
| draw-str | ( str -- ) | H | ssd1306-i2c.forth |  |
| str-width | ( str -- ) | H | ssd1306-i2c.forth |  |

## SSD1306SPI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SCL | ( -- 14 ) | H | ssd1306-spi.forth | 1 long constant |
| SDA | ( -- 13 ) | H | ssd1306-spi.forth | 1 long constant |
| DC | ( -- 2 ) | H | ssd1306-spi.forth | 1 long constant |
| RST | ( -- 0 ) | H | ssd1306-spi.forth | 1 long constant |
| BUS | ( -- 1 ) | H | ssd1306-spi.forth | 1 long constant |
| SPI\_WORD\_SIZE\_8BIT | ( -- 1 ) | H | ssd1306-spi.forth | 1 long constant |
| freq | ( divider count -- freq ) | H | ssd1306-spi.forth |  |
| DISPLAY\_WIDTH | ( -- 128 ) | H | ssd1306-spi.forth | 1 long constant |
| DISPLAY\_HEIGHT | ( -- 64 ) | H | ssd1306-spi.forth | 1 long constant |
| ESSD1306 |  | H | ssd1306-spi.forth | exception |
| ESSD1306\_WRITE |  | H | ssd1306-spi.forth | exception |
| BUFFER\_SIZE | ( -- DISPLAY\_WIDTH DISPLAY\_HEIGHT \* 8 / ) | H | ssd1306-spi.forth | 1 long constant |
| screen1 |  | H | ssd1306-spi.forth | buffer, size BUFFER\_SIZE |
| actual | ( -- adr ) | H | ssd1306-spi.forth | 1 long variable, initial value screen1 |
| screen | ( -- buffer) | H | ssd1306-spi.forth |  |
| wire | ( -- ) | H | ssd1306-spi.forth |  |
| check-write-result | ( code -- | ESSD1306\_WRITE ) | H | ssd1306-spi.forth |  |
| write-command | ( cmd -- | ESSD1306\_WRITE ) | H | ssd1306-spi.forth |  |
| display-invert | ( -- ) | H | ssd1306-spi.forth |  |
| display-normal | ( -- ) | H | ssd1306-spi.forth |  |
| RIGHT | ( -- 38 ) | H | ssd1306-spi.forth | 1 long constant |
| LEFT | ( -- 39 ) | H | ssd1306-spi.forth | 1 long constant |
| scroll-start | ( stop-row start-row direction -- ) | H | ssd1306-spi.forth |  |
| scroll-stop | ( -- ) | H | ssd1306-spi.forth |  |
| write-data | ( data -- | ESSD1306\_WRITE ) | H | ssd1306-spi.forth |  |
| display-on | ( -- ) | H | ssd1306-spi.forth |  |
| init | ( -- ) | H | ssd1306-spi.forth |  |
| display-reset | ( -- ) | H | ssd1306-spi.forth |  |
| y>bitmask | ( y -- bit-index ) | H | ssd1306-spi.forth |  |
| xy-trunc | ( x y -- x' y' ) | H | ssd1306-spi.forth |  |
| xy>i | ( x y -- bit-mask buffer-index ) | H | ssd1306-spi.forth |  |
| or! | ( value addr -- ) | H | ssd1306-spi.forth |  |
| and! | ( value addr -- ) | H | ssd1306-spi.forth |  |
| set-pixel | ( x y -- ) | H | ssd1306-spi.forth |  |
| unset-pixel | ( x y -- ) | H | ssd1306-spi.forth |  |
| pixel-set? | ( x y -- flg ) | H | ssd1306-spi.forth |  |
| hline | ( x y width -- ) | H | ssd1306-spi.forth |  |
| rect-fill | ( x y width height -- ) | H | ssd1306-spi.forth |  |
| fill-buffer | ( value -- ) | H | ssd1306-spi.forth |  |
| display | ( -- ) | H | ssd1306-spi.forth |  |
| display-clear | ( -- ) | H | ssd1306-spi.forth |  |
| display-init | ( -- | ESSD1306 ) | H | ssd1306-spi.forth |  |
| font | ( -- adr ) | H | ssd1306-spi.forth | 1 long variable, initial value is 0 |
| text-left | ( -- adr ) | H | ssd1306-spi.forth | 1 long variable, initial value is 0 |
| text-top | ( -- adr ) | H | ssd1306-spi.forth | 1 long variable, initial value is 0 |
| font-size | ( -- adr ) | H | ssd1306-spi.forth | 1 long variable, initial value is 1 |
| font-small | ( -- ) | H | ssd1306-spi.forth |  |
| font-medium | ( -- ) | H | ssd1306-spi.forth |  |
| font-big | ( -- ) | H | ssd1306-spi.forth |  |
| font-xbig | ( -- ) | H | ssd1306-spi.forth |  |
| draw-lf | ( -- ) | H | ssd1306-spi.forth |  |
| draw-cr | ( -- ) | H | ssd1306-spi.forth |  |
| dot | ( x y -- ) | H | ssd1306-spi.forth |  |
| stripe | ( bits -- ) | H | ssd1306-spi.forth |  |
| draw-char | ( char -- ) | H | ssd1306-spi.forth |  |
| draw-str | ( str -- ) | H | ssd1306-spi.forth |  |
| str-width | ( str -- ) | H | ssd1306-spi.forth |  |

## TASKS

Punyforth supports cooperative multitasking which enables users to run more than one task simultaneously.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PAUSED | ( -- 0 ) | H | tasks.forth | 1 long constant |
| SKIPPED | ( -- 1 ) | H | tasks.forth | 1 long constant |
| Task |  | H | tasks.forth | structure |
| .next |  | H | tasks.forth | field within structure |
| .status |  | H | tasks.forth | field within structure |
| .sp |  | H | tasks.forth | field within structure |
| .rp |  | H | tasks.forth | field within structure |
| .ip |  | H | tasks.forth | field within structure |
| .s0 |  | H | tasks.forth | field within structure |
| .r0 |  | H | tasks.forth | field within structure |
| .handler |  | H | tasks.forth | field within structure |
| REPL | ( -- here Task allot ) | H | tasks.forth | 1 long constant |
| task-stack-size | ( -- adr ) | H | tasks.forth | 1 long variable, initial value 112 |
| task-rstack-size | ( -- adr ) | H | tasks.forth | 1 long variable, initial value 112 |
| last | ( -- adr ) | H | tasks.forth | 1 long variable, initial value REPL |
| current | ( -- adr ) | H | tasks.forth | 1 long variable, initial value REPL |
| alloc-stack | ( -- a ) | H | tasks.forth |  |
| alloc-rstack | ( -- a ) | H | tasks.forth |  |
| task: | ( user-space-size <name> ; -- task ) | H | tasks.forth |  |
| choose | ( -- ) | H | tasks.forth |  |
| save | ( sp ip rp -- ) | H | tasks.forth |  |
| restore | ( -- ) | H | tasks.forth |  |
| switch | ( task -- ) | H | tasks.forth |  |
| user-space | ( -- a ) | H | tasks.forth |  |
| pause |  | H | tasks.forth | deferred word |
| pause-multi | ( -- ) | H | tasks.forth |  |
| s0-multi | ( -- top-stack-adr ) | H | tasks.forth |  |
| r0-multi | ( -- top-rstack-adr ) | H | tasks.forth |  |
| activate | ( task -- ) | H | tasks.forth |  |
| stop | ( task -- ) | H | tasks.forth |  |
| deactivate | ( -- ) | H | tasks.forth |  |
| task-find | ( task -- link ) | H | tasks.forth |  |
| tasks-print | ( -- ) | H | tasks.forth |  |
| semaphore | ( -- ) | H | tasks.forth |  |
| mutex | ( -- ) | H | tasks.forth |  |
| wait | ( semaphore -- ) | H | tasks.forth |  |
| signal | ( semaphore -- ) | H | tasks.forth |  |
| multi-handler | ( -- a ) | H | tasks.forth |  |
| multi | ( -- ) | H | tasks.forth |  |
| single | ( -- ) | H | tasks.forth |  |

## MAILBOX

Often tasks need to communicate with each other. A mailbox is a fixed size blocking queue where messages can be left for a task. Receiving from an empty mailbox or sending to a full mailbox blocks the current task.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| mailbox: | ( size -- ) | H | mailbox.forth |  |
| mailbox-send | ( message mailbox -- ) | H | mailbox.forth |  |
| mailbox-receive | ( mailbox -- message ) | H | mailbox.forth |  |

## TCP-REPL

To set Punyforth to work with a telnet terminal:-

1. Set up the wifi connection with wifi-connect "yourpassword" "yourrouterssid" this will report an IP address and port given

2. Load the REPL over TCP module with TCPREL load

3. Start the remote terminal session repl-start

4. On the remote PC, open a telnet session to the ip address the router gave Punyforth e.g 192.168.1.8 on port 1983

5. Remember that local echo and CR / LF are needed to terminate a line

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| HOST | ( -- wifi-ip ) | H | tcp-repl.forth | 1 long constant |
| PORT | ( -- 1983 ) | H | tcp-repl.forth | 1 long constant |
| client | ( -- adr ) | H | tcp-repl.forth | 1 long variable, initial value 0 |
| line | ( -- adr ) | H | tcp-repl.forth | buffer, size 128 |
| connections |  | H | tcp-repl.forth | mailbox |
| repl-server-task |  | H | tcp-repl.forth | task |
| repl-worker-task |  | H | tcp-repl.forth | task |
| type-composite | ( str -- ) | H | tcp-repl.forth |  |
| emit-composite | ( chr -- ) | H | tcp-repl.forth |  |
| eval | ( str -- i\*x ) | H | tcp-repl.forth |  |
| server | ( task -- ) | H | tcp-repl.forth |  |
| command-loop | ( -- ) | H | tcp-repl.forth |  |
| worker | ( task -- ) | H | tcp-repl.forth |  |
| repl-start | ( -- ) | H | tcp-repl.forth |  |

## TURNKEY

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SIZE | ( -- 4096 ) | H | turnkey.forth | 1 long constant |
| BOOT\_ADDR | ( -- 16r5100 ) | H | turnkey.forth | 1 long constant |
| ETURNKEY |  | H | turnkey.forth | exception |
| boot |  | H | turnkey.forth | deferred word |
| dst | ( -- n ) | H | turnkey.forth |  |
| heap-size | ( -- n ) | H | turnkey.forth |  |
| check | ( code ( code -- | ETURNKEY ) | H | turnkey.forth |  |
| n, | ( addr n -- addr+strlen ) | H | turnkey.forth |  |
| s, | ( str-dst str-src -- str-dst+strlen ) | H | turnkey.forth |  |
| save-loader | ( -- ) | H | turnkey.forth |  |
| turnkey | ( -- ) | H | turnkey.forth |  |

## WIFI

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| NULL\_MODE | ( -- 0 ) | H | wifi.forth | 1 constant long |
| STATION\_MODE | ( -- 1 ) | H | wifi.forth | 1 constant long |
| SOFTAP\_MODE | ( -- 2 ) | H | wifi.forth | 1 constant long |
| STATIONAP\_MODE | ( -- 3 ) | H | wifi.forth | 1 constant long |
| MAX\_MODE | ( -- 4 ) | H | wifi.forth | 1 constant long |
| AUTH\_OPEN | ( -- 0 ) | H | wifi.forth |  |
| AUTH\_WEP | ( -- 1 ) | H | wifi.forth |  |
| AUTH\_WPA\_PSK | ( -- 2 ) | H | wifi.forth |  |
| AUTH\_WPA2\_PSK | ( -- 3 ) | H | wifi.forth |  |
| AUTH\_WPA\_WPA2\_PSK | ( -- 4 ) | H | wifi.forth |  |
| AUTH\_WPA\_WPA2\_PSK | ( -- 5 ) | H | wifi.forth | ?? |
| EWIFI |  | H | wifi.forth | exception |
| >ipv4 | ( octet1 octet2 octet3 octet4 -- n ) | H | wifi.forth |  |
| check-status | ( status -- | throws:EWIFI ) | H | wifi.forth |  |
| wifi-connect | ( password ssid -- | throws:EWIFI ) | H | wifi.forth | Connect to an existing Wi-Fi access point with the given ssid and password. For example: "ap-pass" "ap-ssid" wifi-connect |
| wifi-softap | ( max-connections channels hidden authmode password ssid -- | throws:EWIFI ) | H | wifi.forth | Creates an access point mode with the given properties. For example:  172 16 0 1 >ipv4 wifi-set-ip  4 3 0 AUTH\_WPA2\_PSK "1234567890" "my-ssid" wifi-softap  8 172 16 0 2 >ipv4 dhcpd-start  max-connections should be <= max-leases |
| ip | ( interface -- str ) | H | wifi.forth |  |
| wifi-ip | ( -- str ) | H | wifi.forth | station ip |
| softap-ip | ( -- str ) | H | wifi.forth | station ip |

# Document version

Version 1.0 Initial version based on Punyforth v0.5, compiled by Bob Edwards, SW U.K. Ham radio callsign G4BBY