

NAME

antBASIC - A modern version of Tiny BASIC with GPIO functions

SYNTAX

antbasic [*file* [arguments]]

ARGUMENTS

number-to-A [number-to-B [string-to-@]]

example:

antbasic

activate interactive session

antbasic test.bas

execute *test.bas* and return to the shell

antbasic test.bas 10

invoke program and pass number 10 to variable **A**

antbasic test.bas 10 20

invoke program and pass numbers 10 and 20 to variables **A** and **B**

antbasic test.bas 10 20 'Hello, world!'

invoke and pass numbers to variables and string to **string array @**

RETURNS

antBASIC returns an 8bit status code to the shell (default value is zero). You can use the **END** statement to pass a non-zero value to the shell.

example: **END 123** returns 123 as a status code to the shell

VERSION

This man page documents antBASIC version **1.0.1**.

DESCRIPTION

antBASIC is a product of the **BMH (Bare Metal Hacking)** project. It is a modernized version of **Tiny BASIC**, with the addition of I/O manipulation instructions prepared for the **Raspberry Pi**. Although the language specification is minimal, beginners can learn the basics of programming, and a wide range of I/O controls through antBASIC.

PROGRAM

A program consists of several lines, and each line always starts with a line number (1-9999). The maximum program size is **2,000 lines** and **30,000 bytes** (you can check the program size with the **FREE** command). Users can enter sentences interactively through the **GNU Readline** input editor. Usually, there will be ten intervals between line numbers so that you can make additions easily later. When there is no more space between lines, you can create a new gap with the **RENUM** command. Multiple statements can be written in a single line, separated by a *colon*.

example:

```
10 FOR I=1 TO 10
20 PRINT "Hello, ";:PRINT "world! ";
30 NEXT
40 END
```

NUMBERS

Signed 16bit integer (range from -32768 to 32767). Decimal and hexadecimal (*0x* prefix is needed) numbers are distinguished internally.

example: 1234, -1234, 0xABCD, 0xEF

STRINGS

A string is defined as **Unicode** characters (encoded by **UTF-8**) surrounded by double quotations. Escaped special characters are as follows.

| | |
|--------------|----|
| Alarm (BELL) | \a |
| BS | \b |
| TAB | \t |

| | |
|------------|---|
| LF | \n |
| CR | \r |
| ESCAPE | \e |
| backslash | \\ |
| ASCII code | \x## (## is a two-digit hexadecimal number) |

Special array @ holds a string. It must be terminated with *NULL (0)*.

example:

```
@="hello!":@[0]=@[0]-0x20:print @ -> Hello!
@[0]=33:@[1]=7:@[2]=0:print @ -> ! with alarm
```

VARIABLES

Vaiables A to Z hold integer.

example: A=123:B=A+0x1234

ARRAYS

Arrays A[] to Z[] hold integers (*index starts from ZERO*). Two-dimensional array form is X[column,row].

example:

```
DIM A[1],B[2,3]:A[0]=1:B[0,0]=0,1,2,3,4,5
A[0] -> 1,B[0,2] -> 2,B[1,0] -> 3,B[1,2] -> 5
```

OPERATORS

Operator precedence: Unary > Mul/Div/Mod > Add/Sub > Condition > Bitwise

| | |
|-------------|----------------------|
| Unary | -xxx, +xxx |
| Mul/Div/Mod | *, /, % |
| Add/Sub | +, - |
| Condition | ==, !=, <, <=, >, >= |
| Bitwise | &, |

STATEMENTS

| | |
|--------------|---|
| CLS | Clear screen |
| DIM | Define array <i>size</i> (not the maximum index number): DIM[column,row]. NOTE: There is an array size limitation (<i>column*row <= 512</i>). |
| END | Terminate program. If a number is given, antBASIC returns the value to the shell. |
| FOR/NEXT | Iterate statements between FOR and NEXT. example: S=0:FOR A=1 TO 10:S=S+A:NEXT NOTE: <i>increment step is fixed to ONE</i> |
| GOTO | Jump to specified line number. example: GOTO 100, GOTO X |
| GOSUB/RETURN | Call subroutine / return to caller. example: GOSUB 200, GOSUB Y |
| IN | Read bit status. Argument is <i>BMH-style GPIO number (1-14)</i> . returns: 0 or 1 example: IN (B) -> 0 1 |
| INPUT | Input data from user and stores it in a variable or string array @. example: in the case of number) INPUT A, string) INPUT @ |

| | |
|--------|---|
| OUT | Set bit output as zero or one. First argument is a BMH-style GPIO number (1-14) and second argument is a bit Level (0 GND 1 Vdd). example: OUT (B, L) |
| OUTHZ | Set bit output as zero or high-impedance (HiZ). First argument is a BMH-style number (1-14), second argument is a bit Level (0 GND 1 Vdd), and third argument is a mode of internal Pull-up resistor (0 None 1 Pull-up). example: OUTHZ (B, L, P) |
| PRINT | Print data. integer: immediate value, variable, array hexadecimal format (2 or 4-digit): HEX2(<i>number</i>), HEX4(<i>number</i>) string: @ separator: semicolon = no spacing, comma = do tabulation example: PRINT "H"; "I"; "!" -> HI! |
| REM | Insert a remark. <i>Comment must be added as a STRING</i> . example: REM, REM "This is a comment string" |
| RND | Returns random number (range from 0 to 32767). example: RND () |
| MSLEEP | Suspend execution for <i>milli</i> -seconds. example: MSLEEP (1000) -> 1sec wait |
| USLEEP | Suspend execution for <i>micro</i> -seconds. example: USLEEP (1000) -> 1msec wait |

DIRECT COMMANDS

| | |
|--------|---|
| CLEAR | Clear containers (variables and arrays). |
| CLS | Clear screen. |
| DELETE | Delete program lines. example: single line) DELETE 100, multiple lines) DELETE 210, 290 |
| DUMP | Dump containers. example: DUMP (all), DUMP V (variables), DUMP A (arrays), DUMP S (string), DUMP L (program lines), DUMP B (bytecodes) |
| END | Quit antBASIC. |
| FILES | List files. example: current working directory) FILES, specified directory) FILES " <i>path-name</i> " |
| FREE | Display memory usage. |
| HELP | Display help information. |
| LIST | List all or part of program. example: all) LIST, single line) LIST 100, multiple lines) LIST 210, 330 |
| LOAD | Load a source file into memory. example: LOAD "example/hello.bas" |
| MERGE | Merge an additional file into memory. example: MERGE "mylib/addon.bas" |
| NEW | Clear program. |
| RENUM | Renumber program lines. |

example: default [start 100, step 10]) RENUM, define start) RENUM 1000, specify start and step) RENUM 5000,5

RUN Start-up program. *CONTROL-C* aborts the program.

SAVE Save program as a text file.

example: SAVE "myprogram.bas"

ENVIRONMENT VARIABLE

ANT_MICROWAIT

There are two types of wait functions, **MSLEEP()** and **USLEEP()**, in antBASIC. The former is a delay in *seconds*, while the latter is in *micro-seconds*. By default, both functions use the *usleep system call* internally, but a delay in the order of micro-seconds can lead to time variability.

If more precise control in micro-seconds is required, set the **ANT_MICROWAIT** environment variable. Then the **USLEEP()** function does not use the *usleep system call* but uses a simple loop for the number of times specified by **ANT_MICROWAIT**.

antcalib is a utility for estimating the number of loops required for a microsecond delay. The first argument specifies the number of loops, and the second argument specifies the number of loop calls.

```
$ ./antcalib 220 10000000
Loopcount = 220
Number of loops = 10000000

Elapsed time --> 10 sec 9327 usec
Mean time --> 1.000933 usec/loop
```

On a *Raspberry pi 400*, the loop count is around 220. Once the loop count is determined, add the export command to the **~/.bashrc**.

```
export ANT_MICROWAIT=220
```

READLINE LIBRARY

Default Makefile will build an antBASIC binary linked with the **GNU Readline library**. This binary allows the user to do editing lines before sending them to antBASIC.

HOME PAGE & SOURCE REPOSITORY

<https://baremetalhack.com>
<https://github.com/baremetalhack/antBASIC>

FEEDBACKS

I'm looking forward to your comments and improvement reports.
antbasic@baremetalhack.com

AUTHOR

Doctor BMH
 Wataru Nishida, *M.D., Ph.D.*

PUBLICATION DATE

April 30rd, 2022
 Published from *Japan*

COPYRIGHT

Public domain, CC0, *Zero is Infinite*