NAME

tinybasic − Tiny BASIC interpreter and compiler

SYNOPSIS

tinybasic [**options**] *program-file*

DESCRIPTION

Tinybasic is an implementation of the Tiny BASIC language. It conforms to the specification by Dennis Allison, published in People's Computer Company Vol.4 No.2 and reprinted in Dr. Dobb's Journal, January 1976.

The package provides both an interpreter and a compiler in the same executable. Both of these tools are non-interactive, and load their input from a source file written in a text editor. No interactive interpreter is provided.

Tinybasic provides a few additional features. Comments with the **REM** statement were not part of the original specification, but are allowed here. There is support for optional line numbers, and a configurable upper limit for them. Because not all lines need a number, this manual will refer to them as ´line labels.´ Where the phrase ´line number´ appears, it will refer to the actual line count in the source file, as a text editor would show.

OPTIONS

−g *limit*, **−−gosub-limit=***limit*

Specifies the maximum depth of subroutine calls for the interpreter. Calling subroutinnes within subroutines to a level deeper than this will result in the "Too many GOSUBs" runtime error. This does not affect compiled code.

−n *value*, **−−line−numbers=***value*

Determines the handling of line labels. An argument of **m** or **mandatory** causes **tinybasic** to require a line label for every program line, in ascending order. An argument of **i** or **implied** causes **tinybasic** to supply labels internally for each line that lacks them; care must be taken when labelling lines so that there is room for a sequence of numbers between one line label and the next. An argument of **o** or **optional** makes line labels completely optional; those that are supplied need not be in ascending order.

−N *limit*, **−−line−number−limit=***limit*

Specifies the largest line label allowed in the BASIC program. The default is 32767, which is the highest value that **tinybasic** supports. The original Tiny BASIC had a limit of 255.

−o *comment-option*, **−−comments=***comment-option*

Enables or disables support for comments and blank lines in programs. *Comment-options* can be **e** or **enabled** to support comments and blank lines, which is the default setting. It can be **d** or **disabled** to disable support for comments, as per the original Tiny BASIC specification.

−O [*output-type*], **−−output[=***output-type*]

Specifies compilation or translation instead of interpretation, and what type of output is desired. If the option is supplied without an *output−type*, then the default is **lst**. If the option is absent altogether, then the program will be interpreted rather than compiled or translated. Current *output−type*s supported are **lst** for a formatted listing, **c** for a C program ready to compile, or **exe**. Where the output type is **lst** or **c** the output filename is the same as the input filename, with an added extension the same as *.output−type*. Where the output type is**exe**, the output file is dependent on the input filename and the **TBEXE** (see the section on Compilation).

PROGRAM FORMAT

Programs are text files loaded in on invoking **tinybasic**. Each line of the file consists of an optional line label, a command keyword, and the command's parameters, if it has any. Lines may be blank, or the command keyword may be REM, which denotes that the rest of the program line is a comment.

That describes a program written using **tinybasic**'s additional features. In a traditional Tiny BASIC program each line has a mandatory line label, a command keyword which may *not* be **REM**, and the command's parameters. The original Tiny BASIC specification did not provide for comments or blank lines,

and the line labels were required by the language's interactive line editor.

COMMANDS

LET *variable=expression*

Assigns a value, the result of *expression*, to a variable, *variable*. *Variable* must be a single letter, A..Z. *Expression* must evaluate to an integer in the range -32768 to 32767.

IF *condition* **THEN** *statement*

Conditional execution. If*condition* is true, then *statement* is e xecuted. *Statement* can be another **IF**, allowing conditions to be chained, effectively mimicking an AND operator.

GOTO *expression*

Transfer execution to another part of the program. *Expr ession* is evaluated, and program execution continues at the line marked with the corresponding label.

GOSUB *expression*

Calls a subroutine. *Expr ession* is evaluated, and program execution transfers to the line marked with the corresponding label. The position of the **GOSUB** is remembered so that a **RETURN** can bring program execution back to the statement following the **GOSUB**.

RETURN

Return from a subroutine. Program execution returns to the statement following the **GOSUB** which called the present subroutine.

END Terminates program execution.

PRINT *output-list*

Produces output to the console. *Output-list* is a list of items separated by commas. Each item can be either a string literal enclosed in double quotation marks, or a numeric expression. An end of line sequence is output after all the values, so that the next **PRINT** statement will put its output on a new line.

INPUT *variable-list*

Asks for input from the console. *Variable-list* is a list of variable names. For each variable given, a question mark is output and the value typed by the user is stored in that variable. **Tinybasic** allows multiple values to be typed by the user on one line, each separated by any non-numeric character.

REM *comment-text*

Provides space for free-format comment text in the program. Comments have no effect on the execution of a program, and exist only to provide human-readable information to the programmer. Use of this command will raise an error if support for comments is disabled (see the **-o**/**--comment** option above).

EXPRESSIONS

Expressions in Tiny BASIC are purely arithmetic expressions, involving integers only. The four basic arithmetic operators are supported: multiplication (*****), division (/), addition (+) and subtraction (-). Unary operators for positive (+) and negative (-) are supported, as are parentheses for affecting the order of operations.

Standard operator precedence evaluates parentheses first, then unary signs, then multiplication and division, with addition and subtraction last.

CONDITIONS

The relational operators are =, >, <, <> or ><, >=, and <=. They are not supported within arithmetic expressions, but can only be used as conditions in **IF** statements in the form: *expression relational-operator expression*

COMPILATION

Tinybasic is capable of compiling programs into executables with the help of a C compiler. To use this facility, the **TBEXE** environment variable must be set before invoking **tinybasic**. The variable should contain the command that compiles a C program into an executable, and may contain the following tokens:

$(SOURCE): the C source filename is substituted here.

$(TARGET): a target filename is substituted here.

The C source filename will be the same as the BASIC filename but with the extension **.c** added. The target filename is the BASIC source filename with the **.bas** extension removed; if the BASIC source filename has no extension, then **.out** is added to prevent the source being overwritten by the executable. If your operating system requires an extension like **.exe** for its executables, then you need to add it explicitly (i.e. **$(TARGET).exe**) - unless the compiler adds that itself. As an example, the file **test.bas** could be compiled on a Unix system with the following commands:

```
$ TBEXE='gcc -o $(TARGET) $(SOURCE)'
$ tinybasic -Oexe test.bas
```

This would produce the executable file **test**, and as a side effect, the C source file **test.bas.c**.

ERROR MESSAGES

Program error messages can be in one of two forms:

Parse error: *description*, line *line-number*, label *line-label*
Run-time error: *description*, label *line-label*

Parse errors are those that are detected before the program starts. Run-time errors are those that cannot be detected until the program is running. If a parse error is detected on a line without a label, then the label section is omitted from the error message. The error messages and their meanings are as follows.

Invalid line number
One of the following has occurred: (i) a line label is missing when line numbers are mandatory; (ii) a line label is lower than the previous one when line numbers are mandatory or implied.

Unrecognised command
The command keyword is not recognised. Note that **REM** will not be recognised when comments are disabled, and will produce this error.

Invalid variable
In a **LET** or **INPUT** statement, something other than a letter from **A** to **Z** was supplied when a variable name was expected.

Invalid assignment
The **=** sign was missing from a **LET** statement.

Invalid expression
An expression in this line is invalid. It is possibly lacking an operator, variable or value where one is expected.

Missing)
An expression contains a left parenthesis and no corresponding right parenthesis.

Invalid PRINT output
Something is wrong with the output list in a**PRINT** statement. It could be: (i) completely missing, (ii) missing a separator between two items, or (iii) missing an item between two separators or at the start or end of the list.

Invalid operator
An unrecognised operator was encountered in an expression or a condition.

THEN expected
The mandatory **THEN** keyword is missing from its expected place in an **IF** statement.

Unexpected parameter
A parameter was given to a command that should not have one, such as **END** or **RETURN**.

RETURN without GOSUB

A **RETURN** was encountered without having executed a **GOSUB**. This commonly occurs when a programmer forgets to put an **END** or a **GOTO** before a subroutine, and allows execution to blunder into it.

Divide by zero

The divisor in an expression was **0**. If dividing by a variable or an expression, it is advisable to check beforehand that it cannot be zero. An intentional division by zero is not the most graceful way to stop a program.

Overflow

When given as a parse error, there is a value in the program that is outside the range of **-32768** to **32767**. When given as a runtime error, an expression in the program or an input from the user has produced a result outside this range.

Too Many GOSUBs

Subroutines were called to a level deeper than the **GOSUB** limit allows. Often encountered because of runaway recursion, or because an incorrect label was given in a **GOSUB** statement causing a subroutine to unintentionally call itself.

VERSION INFORMATION

This manual page documents **tinybasic**, version 1.0.4.

AUTHORS

Tiny BASIC was originally designed by Dennis Allison. This implementation was written by Damian Gareth Walker.

EXAMPLE

This program prints out all of the numbers in the Fibonnaci series between 0 and 1000.

```
      LET A=0
      LET B=1
      PRINT A
100   PRINT B
      LET B=A+B
      LET A=B-A
      IF B<=1000 THEN GOTO 100
      END
```