Terminal-BASIC

User reference manual

Andrey Skvortsov <starling13@mail.ru>

http://starling13.clan.su

Version 2.3-a2

03/09/21 16:48:35

Contents

1 Introduction

Terminal-BASIC (TB) is a free implementation of BASIC programming language. It is s partially compliant to the standards ISO/IEC 6373:1984 and USSR/Russian ГОСТ 27787-88 and also supports some features, allowing to use many BASIC type-in utilities and games from old BASIC books and magazines.

TB was inspired by the TinyBASIC port for Arduino-compatible uC-based systems and Dartmouth BASIC - the World's first BASIC system. But the main reason, author decided to make such an archaic piece of software, was happy times, he spent dealing with the education computers Electronica MS-0511 and interactive Vilnius BASIC programming system at high-school classes.

TB is designed to run on the simplest embedded systems, based on micro controllers and microprocessors with the program memory of at least 16kb and 1kb of RAM (such as AVR 8-bit Arduino boards), but it is cross-platform, has Linux and Windows versions, suitable for evaluation and software debugging.

The main features of TB interpreter:

- supports number of data types (integer (2 bytes signed), long integer (4 bytes signed), real (4 bytes binary floating point), long real (8 bytes binary floating point), boolean and string) using variables and function suffixes;
- multidimensional arrays of arbitrary size and dimensions;
- Dartmouth-BASIC-like matrix operations;
- optional time-sharing system mode with round-robin scheduling using multiple I/O devices for each user (i.e. USART);
- configuration headers provide the number of options, which enable inclusion of the language parts and features thus allowing to adjust the code size.

2 Getting and installing Terminal-BASIC

If you don’t plan to improve/develop Terminal-BASC and work with the source repositories, you can download TB packages from TB Sourceforge site: https://sourceforge.net/projects/terminal-basic/files. There are both binary precompiled and source tarball packages in the download section of SourceForge TB page. Precompiled packages exist for Windows platform. Also there are HEX-files for AVR 8-bit controllers for stable versions. Source tarballs provided include POSIX tarballs (which can be built the standard way “./configure && make && make install”), the Arduino sketches, which can be configured by the editing some header files and easily built and burned in the Arduino IDE.

2.1 Example 1: POSIX-compatible systems

Pure posix version (Linux, *BSD, MinGW etc.) has all features except graphical capabilities (see SDL2 version instead).

The source package for POSIX systems can be downloaded from files section and has the name “terminalbasic-<version>-tar.gz”.

Default configuration files enables all the necessary options, but you may decide to look at them and change something, before building from source package.

Figure 1 shows the sequence of configuration files for POSIX version.

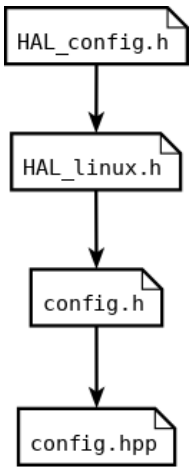

Figure1 1: Hierarchy of configuration files
on POSIX target platform

HAL_config.h file enables the parts of hardware abstraction level API, used by the TB application code.

Option	Values	Description
HAL_TERMINAL_NUM	1-4	Number of available terminals. Currently POSIX version

		supports only 1 terminal
HAL_NVRAM	1 / 0	Enables NVRAM HAL feature. On POSIX systems NVRAM is emulated as binary file. If you wish to use SAVE/LOAD commands don’t disable it
HAL_EXTMEM	1 / 0	Enables external memory feature. On POSIX systems external memory is modelled as directory, where BASIC programs and created files are stored.
HAL_EXTMEM_NUMFILES	1-255	Number of the simultaneously opened files. On POSIX systems there is no need to reduce this option very much.
HAL_GFX	1 / 0	Enables HAL graphics API, not supported by the POSIX version
HAL_GPIO	1 / 0	Enables HAL feature of controlling GPIO capabilities of the target platfrm. Not supported on POSIX systems
HAL_BUZZER	1 / 0	Enables HAL feature of the simple one-channel sound device. On POSIX platforms it only suitable for supporting **BEEP** BASIC command.

HAL_pc.h file contains the options of POSIX HAL implementation.

Option	Values	Description
HAL_PC_FILES_PATH	cstring	Path to the directory of the all TB runtime files for NVRAM and external memory operations

3 Commands and functions reference

3.1 Data types

Constants, expressions, variables and arrays can represent different data types:

- integer values (2 bytes signed [-32768; 32768));
  - Variable and array names should end with % suffix: A%, VR1%, BUF%(10).
- long integer (4 bytes signed [-2147483648; 2147483648));
  - Variable and array names should end with %! suffix: A%!, VR%!, BF%!(10).
- floating point real (4 bytes single precision IEEE 754);
  - Variable and array names do not use specific suffix: A, VAR1, BUF(10).
- floating point long real (8 bytes double precision IEEE 754);
  - Variable and array names should end with ! suffix: A!, VR1!%, BUF!(10)
- logical (1 bit, arrays are bit-packed and very compact);
  - Variable and array names should end with @ suffix: A@, VR1@, BUF@(10)
- string (up to 72 characters);
  - Variable and array names should end with $ or ¤ suffix: A$, VR1¤, BUF$(10).

3.2 Language core

3.2.1 Commands

CHAIN

Description: Load a program text from non-volatile memory, preserving the existing variables and arrays.

CONT

Description: Continue execution of a program, stopped by a STOP command.

DATA

Description: Start a section of the data

Example:

```
10 DATA 1,2,3,4,5
20 READ A%,B%
30 READ C%
```

```
40 RESTORE : READ D%,E%,F%,G%,H%
50 PRINT A%;B%;C%;D%;E%;F%;G%;H%
RUN
 1 2 3 1 2 3 4 5
```

DEF

Description: Define a one-line user function

Example:

```
10 DEF FN HYP(X,Y) = SQR(X*X + Y*Y)
20 LET A = 12 : LET B = 14
30 PRINT FN HYP(A,B)
RUN
 18.4391
```

DELAY

Description: Delay execution of a program for interval in milliseconds.

Syntax:

DELAY <expression>

DIM

Description: Define an array

Syntax:

DIM <name>(expression[, expression]*), [<name>(expression[, expression]*)]*

Example:

```
DIM A(12)
A(0) = 3.141592
LET I%=3,J%=2
DIM C%(I%,J%,I%*J%), D%(I%+J%)
```

DUMP

Description: Print contents of BASIC memory, variables and arrays

END

Description: Stop program execution and return to interactive mode.

FN

Description: Look at DEF.

FOR

Description: Make an iterative loop.

Example:

```
10 FOR I%=1 TO 6 STEP 2
20   FOR J%=1 TO 3
30     PRINT I%;J%
40   NEXT J%
50 NEXT I%
  1 1
  1 2
  1 3
  3 1
  3 2
  3 3
  5 1
  5 2
  5 3
```

GOTO

Description: Explicit jump to specified program line

Syntax:

GOTO <integer expression>

Example:

```
10 REM ENDLESS LOOP
20 PRINT I% : I% = I% + 1
30 GOTO 20
40 END
```

GOSUB

Description: Explicit jump to subroutine with the ability to return to the call point by RETURN.

Syntax:

GOSUB <integer expression>

Example:

```
10 INPUT A
20 GOSUB 1000
30 PRINT A
40 END
1000 REM SUBROUTINE
1010 A = A ^ 2
1020 RETURN
```

INPUT

LET

Description: Set variable or array element value of the expression.

Syntax:

LET <variable | arrayElement> = <expression>

Example:

```
LET A% = 3
PRINT A%
  3
LET B(1,2) = PI()
PRINT B(1,2)
  3.141592
```

LIST

Dexcription: List current program text.

Syntax:

LIST [startLine[-endLine]]

LOAD

Description: Load program from non-volatile memory, saved previously by the SAVE command.

MAT

Description: Command, performing matrix operations

NEW

Description: Free interpreter memory. No program text, variables, arrays or user functions remain after execution.

NEXT

Description: Next loop iteration, look at FOR.

POKE

PRINT

Description: Output the expressions values to the standart output.

Syntax:

PRINT [[expression | ; | ,]*

Examples:

```
PRINT «Hello, World!»
Hello World
```

```
PRINT «First line»
PRINT
```

```
PRINT «Second one»
First line

Second one
```

```
PRINT 2*2
4
```

```
PRINT «47.31+24.5^2=», 47.31+24.5^2
47.31+24.5^2=       647.56
```

READ

Description: Read a value from data section. Look at DATA;

REM

Description: Make a comment line. Al symbols after REM command to end of line will not be interpreted.

RESTORE

Description: Reset current data section pointer. Next READ statement will start from the begining of the data. Look at DATA.

RETURN

Description: Return from a subroutine, entered by the GOSUB.

RUN

Description: Run a program, stored in BASIC memory

SAVE

Description: Save program to non-volatile storage for loading with the LOAD command.

STEP

Description: Look at FOR.

STOP

Description: Stop a program execution with the ability to continue later by the CONT command.

TO

Description: Look at FOR.

3.2.2 Functions

ABS

Description: Returns the absolute value of its argument.

CHR$

Description: Returns one-character string with the ASCII code, defined by the parameter.

HEX$[1]

Description: Convert numeric expression value to string, containong hexadecimal representation of the value.

INKEY$

Description: Read input character. Unlike INPUT command, there is no waiting for actual input.

INT

RES

RND

SGN

STR$

Description: Return string with the decimal representation of the argument numeric expression.

TIME

3.2.3 Operations

+

Addition

-

Subtraction or unary minus

*****

Multiplication

1 Controlled by option USE_HEX

/

Division

****

Integer division

^

Power

AND

Logical multiplication

DET

Matrix determinant

INV

Matrix inverted

MOD

Integer division remainder

OR

Logical addition

TRN

Matrix transposed

XOR

Logical exclusive or

3.2.4 Constants

CON

Matrix initializer to ones

FALSE

Logical false

IDN

Matrix initializer to identity

TRUE

Logical true

ZER

Matrix initializer to zeros

3.3 Mathematics module

3.3.1 Functions

ACS

ASN

ATN

CBR

COSH

COS²

COT

EXP

LOG

PI

SIN³

SQR

TAN

3.4 String manipulation module

3.4.1 Functions

ASC

2

3

CHR$

HEX$

MID$

LEFT$

RIGHT$

3.5 General purpose IO module

3.5.1 Commands

AWRITE

DNOTONE

DTONE

DWRITE

3.5.2 Functions

AREAD

AREAD%

DREAD

3.6 Filesystem operations module

3.6.1 Commands

DCHAIN

Description: Command equivalent to the sequence of DLOAD и RUN, except that the state of the running program (variables and arrays) is preserved.

Syntax:

DCHAIN <file name string expression>

DIRECTORY

Syntax:

DIRECTORY [firstFileIndex [, lastFileIndex]]

Description: Print external memory file list

DLOAD

Description: Load program text from file. The file should have BAS extension, but the command parameter has no extension.

Syntax:

DLOAD <BAS file name without BAS extension>

DSAVE

Description: Save current program to text file. Файл будет иметь расширение .BAS, но его имя в команде вводится без расширения. Если файл с указанным именем существовал, он будет перезаписан.

Syntax:

DLOAD <file name without extension>

FCLOSE

Description: Close previously opened with the FOPEN command file.

Syntax:

FCLOSE <file number>.

Example: look at FOPEN.

FSEEK

Description: move file rwead/write cursor to specified position.

FWRITE

Description: Записать в файл 1 байт

Syntax:

FWRITE <целочисленное выражение>, <дескриптор файла>

HEADER

SCRATCH

3.6.2 Functions

FOPEN

Description: Open file in external memory.

Parameters: file name

Return: Integer non-negative file number or -1 if error occures.

Example:

```
F% = FOPEN(«TEST.TXT»)
PRINT F%
0
FCLOSE F%
F% = FOPEN(«TET.TXT»)
PRINT F%
-1
```

FREAD

Description: Read next byte from file and move cursor one position forward.

Parameters: file number of the file, previously opened with the FOPEN command.

Return: Byte value [0;255] or -1 if can't read (i.e. end of file).

Example:

```
  5 REM Print text file content
 10 F% = FOPEN(«TEST.TXT»)
 20 IF F%=-1 THEN GOTO 110
 30 B% = FREAD(F%)
 40 IF B% = -1 THEN PRINT «End of file» : GOTO 100
 50 PRINT B%; : IF B%=10 THEN PRINT CHR$(13);
 60 GOTO 30
100 FCLOSE F%
110 END
```

FSIZE

Description: Get file size.

Parameters: Opened file number.

Return: File size in bytes.

3.6.3 Autorun program

Для автоматического выполнения программного кода после загрузки интерпретатора ТБ, в корневом каталоге файловой системы необходимо создать файл программы с именем AUTORUN.BAS.

3.7 Graphics output module

3.7.1 Commands

BOX

Syntax:

BOX <x>,<y>,<width>, <height>

CIRCLE

Syntax:

CIRCLE <x>,<y>,<radius>

COLOR

ELLIPSE

Syntax:

ELLIPSE <x>,<y>,<width>, <height>

CURSOR

LINE

Syntax:

LINE <x1>,<y1>,<x2>, <y2>

LINETO

POINT

Syntax:

POINT <x>,<y>

SCREEN

4 National lexical sets

English	Русский	Français
AND	И	AND
DATA	ДАННЫЕ	DATA
DEF	ОПР	DEF
DIM	РАЗМЕР	DIM
END	КОНЕЦ	FIN
FN	ФУНК	FN
FOR	ДЛЯ	POUR
GOSUB	ВХОД	GOSUB
GOTO	НА	GOTO
IF	ЕСЛИ	IF
INPUT	ВВОД	INPUT
LET	ПУСТЬ	LET
LIST	ЛИСТАТЬ	LISTER
MAT	МАТ	MAT
NEXT	ЦИКЛ	NEXT
NOT	НЕ	NOT
ON	ПРИ	ON
OR	ИЛИ	OR
PRINT	ВЫВОД	PRINT
READ	ВЗЯТЬ	READ
REM	КОМ	REM
RESTORE	СНОВА	RESTORE
RETURN	ВОЗВРАТ	RETOUR
RUN	ПУСК	RUN
STEP	ШАГ	STEP
STOP	СТОП	STOP
THEN	ТО	THEN
TO	ДО	JUSQUA