TDL12KB.TXT

TDL 12 KB Z-80 BASIC Small Doc

(Retyped by Emmanuel ROCHE.)

Foreword

I am retyping and releasing the following because there are no other documentation available about "TDL 12 KB Z-80 BASIC".

As its name says, it is a 12 KB BASIC written by TDL (Technical Design Labs) in Z-80. It was written in a time when "microcomputer hobbyists" where each making their own software and no common software (such as an Operating System like CP/M) was recognized as the "de facto" standard. Hence, it was expected that each user was able to patch his own assembly language subroutines to interface TDL12KB to its particular hardware, hence the references to "PUNCH", "READER", and "TTY" (the "standard" console before screens).

I found this program as a hex dump in an awful German book whose title and author only deserve to be forgotten. Using SID, I entered the 13 Kilo-Bytes in the TPA of my CP/M microcomputer. Using the crude checksum program found in the German book (which was checking blocks of 512 bytes, if I remember well, instead of each 16 bytes as is standard using HEX files which are the standard CP/M way of transmitting binary files), I finally got a "memory image" of TDL12KB. I just hope that the crude checksum subroutine did not pass too many retyping errors.

If you have any original doc about this particular BASIC, please upload it to (at least) the www.retroarchive.org Web site and let me know via the comp.os.cpm Newsgroup (even if I am not reading it, many people have my address and will warn me). (If you have a really interesting doc, I could go as far as retype it, but I would prefer that you do it. I already have enough work to do.)

TDL 12 KB Z-80 BASIC

This BASIC occupies RAM from 100H to 2FFH. For it to run on your system, you must provide the following assembly language subroutines, which must fit from 100H to 300:

CIN	Fetches a character from the console into the Accumulator.
RIV	Fetches a character from the reading device into the Accumulator.
CON	Output a character stored in C register to console.
WRTV	Output a character stored in C register to "PUNCH" (later called "Auxiliary device" (like the RS-232-C port) by CP/M).
LISTX	Output a character stored in C register to the LIST device (in other words: the printer).
CSTSX	Console Status. Outputs the value 0FFH into A register if there is a character (without reading it). Otherwise, outputs 00H.

IOCHX   The current configuration is output to A register.
        May be short circuited by XRA A, RET.

IOSTX   The new I/O Byte is output to C register.
        May also be short circuited by XRA A, RET.

MEMCK   The higher value address of the highest memory location available to the BASIC program is in A register, and the lower value address in B register.
Adter starting up, the size of the memory is requested, the response is obtained by means of CR ("ENTER" on IBM PC keyboards), and the value here specified is used.

TRAPX   Start up of the "proprietary monitor program" (Remember: it was done before the advent of Disk Operating Systems...) (Under CP/M, you can use it to
leave BASIC and reach the debugger, for example to check your assembly language subroutines...)

1 General Service Commands

AUTO

The AUTO command causes line numbering to be produced automatically. At the same time, a start line and the step size may be specified, e.g. AUTO 100 and
AUTO 100,5.

CLEAR

All the variables are cleared. If a number is specified, the area for strings
is set to this value. CLEAR 200 clears all variables and sets the area available for strings to 200.

CONTINUE

If the program is stopped running by means of CTRL-C, its execution may be resumed by this command. If no changes have been made to the program.

DELETE

A range of lines may be deleted. DELETE 100-135 deletes all lines from 100 to
135 inclusive.

KILL

By means of this command, unused storage space may be recovered from arrays.
KILL A,B clears the memory which the arrays A and B used up.

LOAD

Loads a program from the "READER" (the paper tape reader...). In doing so, NEW
is performed first. LOAD P loads a program which was stored by means of SAVE

P. For this purpose, only a single letter is allowed as a name. LOAD? P performs a test read. If the data is loaded, the ASCII character "BEL" ("beep") is output.

LOADGO

Like LOAD, but starts the program. LOADGO P,100 loads program P and starts it
at line 100.

NEW

Deletes the whole of the main memory, i.e., the BASIC program and all the variables.

PRECISION

A computational accuracy of 11 digits is presupposed. PRECISION 4, for example, causes numbers to be rounded to 4 places before output. Internally, however, calculations continue to be made with 11 digits.

RENUMBER

All references, jumps, etc, are renumbered. RENUMBER numbers all lines starting at 10, in increments of 10. RENUMBER 110 begins at 110. RENUMBER 120,
5 numbers in increments of 5, beginning at 120. RENUMBER 500,5,300 first begins numbering at line 300 and then proceeds in steps of 5 from 500 onwards,
thus allowing gaps to be incorporated (???).

RUN

Clears all variables and starts executing the program. Specifying a line number causes the program to start from that line onwards.

SAVE

Store a BASIC program via the "PUNCH" (the paper tape punch device). SAVE P stores an existing BASIC program under the name P. Only one letter is allowed
as a name. (Who said that CP/M 8+3 filespecs were too small, compared to Unix
14 chars?)

2 The EDIT Command

EDIT

A line of a program may be corrected by means of the EDIT command. EDIT 10 corrects line 10, it it exists. In addition, there are further commands, consisting of letters and numbers, which are not output via the console. Numbers extend from 1 to 255 and are here denoted by a lower case "n".

A	Reloads the EDIT buffer from the program memory.
nD	n characters are deleted.
E	End the edit and replace the line.
nFx	Find the nth occurrence of character x in the buffer, and place the pointer directly before it.
H	Delete everything to the right of the pointer and go into the insert mode.
I	Insert all following characters until CR or ESC are input.
nKx	Delete characters from the pointer up to the nth occurrence of character x, but do not delete it.
L	Output the line (List).
Q	Leave the edit without changing anything (Quit).
nR	Replace the following n characters with n existing characters.
X	Pointer to the end of the line and insert.
Space	Pointer to the right.
Rubout	Pointer to the left.
CR	End of the edit with substitution.
ESC	End of the insert mode.

3 Commands for the console

LIST

Output of a program. LIST 10-100 outputs all lines from 10 to 100. LIST 20-
outputs all lines from 20 onwards.

LVAR

Output all current variables and their contents (values?).

NULL

For slow Teletypes: NULL 3.255 ensures that, after each CR/LF, three 0FFH
bytes are sent, while the "print wheel" moves back.

POS

The current position of the "print wheel" is transferred, e.g. A=POS(B). B
is
a "dummy" value.

PRINT

Physically "print" on the roll of paper of the Teletype, not on the paper

tape
(hence is name, since there was no screen on which to "display"). Example:
PRINT 123,A,"TEST",B,CS. When a comma (",") is specified, all 14 columns are
positioned (???), with semicolons (";") two blanks characters (space?) are
left (where?). If there is a comma or a semicolon at the end, then no CR/LF
is
output.

PRINT USING

There are 2 forms of PRINT USING:

1) PRINT USING line; output list
2) PRONT USING string; output list

In the second form, formatting is determined by the string variable. When a
line is specified, the format must be stored in it, beginning with a (').

```
#       Numeric field.

.       Position of decimal point.

+       May stand at the beginning or at the end of a numerical
        specification.

-       Corresponds to the plus sign; only positive numbers are output,
        preceded by a space.

**      Empty positions are filled with *.

$$      $ is placed immediately before the first digit.

**$     Combination of both characters.

^^^^    Four of these signs stipulate that the number will be output in
        exponential notation. The beginning of string fields are marked by .
        They may be followed by one or several of the following characters:

L       Left alignment
R       Right alignment
C       Centering
E       Left alignment with expansion, if the string is too long.
```

SPC

With this command, a number of spaces may be output. Example: PRINT
A:SPC(5);B
(5 additional spaces are inserted between A and B).

SWITCH

With this command, the console allocation may be changed. For this purpose,
SWITCH receives an argument between 0 and 3, where:

0 = TTY
1 = CRT
2 = BATCH USE
3 = USER DEFINED

TAB

With this command, a particular print position may be reached. Example: PRINT
A:TAB(30);B (B will be output starting at position 30).

TRACE

TRACE 1 switches on the trace mode, TRACE 0 switches it off. An arithmetical expression may also stand in place of the number. If this is not equal to 0, the trace mode will be switched on. All line numbers executed will then be output between "<>" brackets.

WIDTH

With this command, the length of output may be specified, at which a CR/LF is
automatically inserted. WIDTH 80 sets the length to 80 characters per line: the minimum is 15, the maximum is 255.

4 Commands for the LIST device (printer)

Most of the previous commands may also be switched to the printer, if the following commands are used:

LLIST
LVAR
LNULL
LPRINT
LPRINT USING
LTRACE
LWIDTH
LPOS
SPC
TAB

5 Commands and functions which move data

LET

Assignment of a value to a variable (always to the left of "="). Example: 10 LET A=20 LET may be omitted, however, ans only serves the purpose of clarity.

DIM

Reservation of storage space for arrays. Arrays may have 1 to 255 dimensions.
Examples:
```
200 DIM A(10),B(40)
210 DIM C(50,10)
220 DIM D(J)
230 DIM A$(221)
```

DATA

Constant file (???), which may be read into by means of READ, e.g. 10 DATA 5,8,7,9,1.4

READ

Reading in of constants, which are declared in DATA. Example: 20 READ A The
first time through, A receives the value 5, then 8, etc, until 1.4; after
this, a further call results in an error.

RESTORE

By means of this command, the read pointer for the READ command may be set
to
the beginning. With this BASIC interpreter, the pointer may be positioned at
a
particular line by specifying a line number, e.g. 200 RESTORE 20.

LINE INPUT

By means of this command, a whole line may be read into a string variable.
General form: LINE INPUT "prompt string"; input list. "Prompt string" is
optional.

INPUT

Input of data via the console. Example: 10 INPUT "INPUT A,B,C"; A,B,C 20
INPUT B$

INP

Reading of a Z-80 port. Example: A=INP(0) A receives the value of channel 0.

OUT

OUT 1,7 gives the value 7 to channel 1.

WAIT

Automatic wait loop for ports. Example: WAIT A,B,C The value in port A is
XORed with C as well as being ANDed with B. The BASIC program is continued
only when the result is no longer zero.

PEEK

Direct memory access: B=PEEK(A) fetches the decimal content of the memory
location with the decimal addres A, and places the result in B.

POKE

With POKE A,B B is written into the line (???) with the address A.

COPY

By means of this command, parts of the BASIC program may be relocated or duplicated. COPY new line,increment=range of lines. The lines of the BASIC program specified by the range of lines are copied to a new area, and renumbered in the process.

EXCHANGE

Quick exchange of values of variables (probably "SWAP").
EXCHANGE A$,B$ EXCHANGE C,D(I,J) In the case of strings, only pointers are exchanged internally.

6 Control of program flow

GOTO

Jump statement (argument = line number).

GOSUB

Call of a subroutine (argument = line number).

RETURN

Return from subroutine.

ON x GOTO
ON x GOSUB

A jump is made to line number x. Example: 10 ON A GOTO 100,125,145 If A=1, then a jump is made to line 100. If A is equal to 0 or greater than the number
of specified line numbers, then the next line is executed.

CALL

Machine sub program call.
CALL address, argument 1, ..., argument n
Calls up a program beginning at "address". Each of the arguments is converted
into a 16-bit number and is passed according to the following schema:

```
SP --> argument n
       ........
       argument 1
HL --> return address
BC --> number of arguments on the stack.
```

FOR,TO,STEP,NEXT

Do statement which may be nested arbitrarily:

```
10 FOR A=B TO C STEP D
20 ...
```

30 NEXT A

IF,THEN,ELSE

Conditional instruction, in which ELSE may here be used to supplement standard
BASIC. Example: 10 IF B=4 THEN 50 ELSE 30
Also possible are
```
20 IF Z$="YES" GOTO 50
30 IF G=5 THEN G=4 ELSE G=7
```
The following comparators are allowed:
```
=  Equals
<> Unequal
<  Less than
>  Greater than
<= Less than or equal to
>= Greater than or equal to
```
Logical operators:
```
NOT Negation
AND AND operation
OR  OR operation
```
Example 20 IF (A=0) OR NOT (B=4) THEN C=5

7 Trigonometric functions

ATN

Calculation of arctangent: A=ATN(.45) The result is given in radian measure.

COS

Calculate cosine; the trigonometric information is given in radian measure.
B=COS(3.1416)

SIN

Calculation of sine: C=SIN(3.1416/2)

TAN

Calculates the tangent of an angle: A=TAN(.254)

8 Diverse functions

ABS

Absolute value: ABS(-4.5) results in 4.5.

DEF FN

This allows the user to define his own functions. In so doing, a function must
begin with FN, followed by the name of a variable, e.g. FNA, FNB6. The name

of
the function is followed by a parameter enclosed in brackets (???
parentheses?). Example: 200 DEF FNQ(X)=X*B+3 Here, X is a variable which is
used globally in the BASIC program. X is confined locally to the definition
and only represents the parameter.
Example:

```
10 DEF FNA(X)=X*X
   ...
100 A=FNA(3)
110 PRINT A
```

The value 9 is output. In this BASIC, the potentiality of the DEF statement
has been extended considerably. It is possible to extend the statement over
several lines; and recursive functions may also be defined.
The general format is:

```
DEF FN name (parameter, ..., parameter)
...
Body of function
...
FNEND function-value
```

In contrast to the standard definition, the equal sign is simply omitted
here.
The definition of the function is terminated by FNEND, where the value of
the
function is specified. The function may also be terminated prematurely, by
causing a return by means of "FNRETURN function-value". Examples:

```
100 DEF FNFAC(I)
200 IF I=0 THEN FNRETURN 1
300 FNEND FNFAC(I-1)*I
400 PRINT FNFAC(6)

100 DEF FNREP$(I$,I)  ' Construction of an iterating string
200 J$=" "
300 IF I<=0 THEN FNRETURN J$
400 FOR J=1 TO I
500 J$=J$+1
600 NEXT
700 FNEND J$
800 PRINT FNREP$("TEST",5)
```

EXP

Exponential function. EXP(1) results in 2.7182...

FRE

By means of this command, the remaining memory is calculated for variables
and
programs if a variable is specified as a dummy parameter, and for strings if
a
string value is used as a parameter. FRE(X) outputs the variables and
remaining program memory. FRE(X$) outputs the remaining memory for strings.

INT

Calculation of the integral part of numbers. Example: 20 C=INT(A) If A has
the value 4.56, C receives the value 4.

LOG

Calculation of the natural logarithm of a number (i.e. to the base
e=2.7...).

LOG(EXP(1)) therefore yields the value 1.

SGN

Gives the result 1 if the argument is greater than 0, zero if the argument
is
equal to 0, and -1 if it is less than 0. SGN(56.5) gives the result +1.

SQR

Calculation of the square root. The argument must not be less than 0. SQR(2)
yields 1.4142...

RND

Generation of a pseudo-random number between 0 and 1. For this purpose, RND
requires a dummy parameter. If the value is less than 0, the RND sequence is
initiated. If the argument is 0, then the previous value is passed. An
argument greater than 0 provides the next RND value in the sequence.

```
10 R=RND(I)  ' R contains a value between 0 and 1.
```

RANDOMIZE

This is not a function, but rather may be used to select a random starting
point of a pseudo-random number sequence.

9 String processing

ASC

The decimal value of the first character of a string is passed.

```
10 A=ASC(A$)  With A$="A", A receives the value 65.
```

CHR$

In this case, precisely the opposite occurs: the character is passed which
is
represented by the decimal value of the argument. The coding on this
occasion
is given in ASCII.

```
10 A$=CHR$(66)  A$ contains the character B.
```

LEFT$

Receives two parameters. The first parameter is a string. The second
parameter
specifies the number of characters which are to be passed, counting from the
left end of the string.

```
30 B$=LEFT$("STRING",2)  B$ receives the character string "ST".
```

LEN

By means of this function, the length of a string may be determined.
40 X=LEN(S$) X contains the number of bytes occupied by S$.

MID$

MID$ requires three parameters. The first specifies the string, the second determines the starting position, and the third specifies the number of characters which are to be used from that position onwards.
50 A$=MID$(B$,5,6) A$ receives 6 characters, beginning with the fifth character in the string B$. MID$ may only be used on the left hand side.

RIGHT$

Functions like LEFT$, only in this case the characters are counted from the right end.

STR$

The string is passed whose numerical value is contained in brackets (??? parentheses?). 10 C$=STR(7.8) C$ receives the string "7.8".

VAL

The converse of STR$: the numeric value of a string is passed.
20 A=VAL("3.4") A receives the value 3.4. In BASIC programs, hexadecimal constants may also be used. They are specified by placing the character & in front. B=&400 ' B receives the value 1024.

INSTR

Serves to search for strings. For this purpose, the first parameter is the string in which the second parameter is to be found, the second parameter the
string to be searched for. In addition, a start position and a length may also
be specified. Examples:

```
INSTR("123456789","456")      gives 4 as a result
INSTR("123456789","654")      gives 0 as a result
INSTR("1234512345","34")      gives 3 as a result
INSTR("1234512345","34",6)    gives 8 as a result
INSTR("1234512345","34",6,2)  gives 0 as a result
```

10 Further Commands

END

Terminates the execution of a program, and may be situated anywhere inside a BASIC program.

REM

Indicates that the line is a comment. The (') character may also be used.

```
10 REM Comment line
20 A=B ' Comment
```

STOP

Like END, except that "BREAK @ LINE ..." is output.

USR

USR allows a machine sub program to call procedures. The program is called up
via the branch address given with "USR:". In order to receive the parameter,
the sub program must be called up at address 300H + 27H. The address of the
parameter will then be in the pair of registers D and E. In order to pass the
information back, the sub program is called up at address 300H + 2AH. In
addition, the lower-order byte of the result is transferred to register B and
the higher-order byte of the result into register A. To return to the BASIC
program, a RET instruction must be executed.

```
10 A=USR(B)
20 PRINT A
```

11 Commands for ASCII Input/Output of programs

ASAVE

On the execution of this command, the BASIC program in memory is output in
readable format via the PUNCH device of the Teletype.

ALOAD,ALOADC
AMERGE,AMERGEC

For loading programs written in ASCII, which are therefore not in the internal
machine format. Each line must begin with a line number and end with a CR
(/LF?). Loading is terminated either by means of CTRL-Z in the input text, or
by means of EOF in the READ sub program. ALOAD deletes a previously existing
program, MERGE merges the incoming lines with the existing program. The
distinction between the "A..." and "A...C" commands lies in the manner of
handling by the paper tape READER. "A...C" commandspresupposes a controllable
READER, and after the input of each and every line, it is interpreted into the
internal format of the machine. "A..." commands read to the end and only
convert after the whole program has been read in. Because of this, more memory
is temporarily required for programs.

12 Control characters

COMMA	"," Go to the next TAB position; or a line delimiter.
SEMICOLON	";" Do not continue.
COLON	":" For several statements in a line.
RUBOUT	(on the ASR-33 Teletype: renamed "DEL"ete in the ASCII standard: the "<--" key on IBM Clowns) 7FH Deletion of an input character between a pair of "file://". (Before the advent of screens, this was

```
        the standard way of showing which character had been deleted.)

CTRL-S  Halt the output of data.

CTRL-Q  Continuation of the output

        (Historical note: at the origin, CTRL-S and CTRL-Q were controling
        the running of the paper tape reader of the ASR-33 Teletype...)

CTRL-C  Termination of the execution of a BASIC program.

CTRL-U  Deletion of the line just input.

CTRL-X  Return to the monitor (remember: no DOS then...).

CTRL-O  Suppression of output from the console.

CTRL-R  Output of the current line without the deleted character.

CTRL-T  This allows the line number just processed to be output
        during the run of the program (???).
```

12 Operators

In order of processing (hierarchically ordered):

()	Brackets (!!! Parentheses!)
^	Exponential operator
-	Negation
*	Multiplication
/	Division
+	Addition
-	Subtraction
=	Equality
<>	Inequality
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
NOT	Logical NOT
AND	Logical AND
OR	Logical OR

Conclusion

The above text is all the documentation that is known about the
TDL 12KB Z-80 BASIC in July 2001.

If you have the original doc, please share it with us!