ZX81 **Basic version 1.02 manual

**Basic is an extender for the ZX81-Basic, that can be used on a ZX81 without any hardware modifications.

With **Basic you will have modified and new ZX81-Basic commands.

All the new commands start with the ZX81 symbol "**" like the mathematical power function. The symbol "**" will not be shown in the listing. So the new commands will be aligned to all other commands.

example:

```
10 REM NEW SYNTAX
20 HELP
30 STOP
```

Basic has therefore a syntax-extension to make "" to be handled as a new keyword. To start entering a new command simply press SHIFT-H. This will show "**" in in the edit-line. Then enter the new command letter by letter. With this syntax a program which is generated with **Basic will be viewable with regular ZX81 Basic or any other PC-program (ZXtool, ZX81List, etc.). But you will see the "**" in the listing then.

New commands can be used in direct mode, entry mode and of course in program run too. The syntax testing of the new command will not be done in line entry. A syntactcal error will only be shown on executing the line.

In addition to the new commands there are some hotkeys in **Basic that control the behavior of **Basic while in program run.

Another new feature is that a program BREAK is now made with SPACE and NEWLINE pressed together. So you can have SPACE or £ be entered by INKEY$.

Different to Sinclair-Basic the new **Basic has a flashing K-, L-, F- and G-cursor.

And it has more readable error-messages now. For instance **Basic now reports an errorfree run with „OK“ instead of „0/10“. When an error occurs it now reports „ERROR 2 IN LINE 10“ instead of „2/10“.

This all helps to easily realize that **Basic is running.

system requirements

For **Basic you need a ZX81 with at least 16k RAM and the original 8k-Basic-ROM installed.

If you have more RAM in the adressrange 8k-16k or above 32k (with M1NOT-circuit) then it can be used for **Basic. Then the whole 16k RAM will be free for your program. Because **Basic uses some private variables it must reside in writable RAM. It will not run in ROM or write-protected RAM.

To run **Basic it is not necessary to have a HRG-capable RAM. But you can run HRG programs with **Basic if you have the adequate hardware. As long as there is no adress-conflict on HRG-data or HRG-code you can use **Basic with most HRG-Software tools as well.

**Basic can be used on Emulator „EightyOne“ (Version 1.0 10/03/08) without problems as long as all needed hardware-options for ROM and RAM are set correctly.

variants

There are four variants of **Basic which are made for different memory-adresses.

PB8K.P
PB12K.P

These variants load between 8k and 16k. Then the whole 16k RAM is free for Basic.

PB16K.P

This variant can be used on any ZX81. It only needs a 16k RAM. It lowers RAMTOP automaticly and copies itself above. Therefore RAMTOP has to be at the original value for a 16k RAM before. Otherwise the installation will stop with an error.

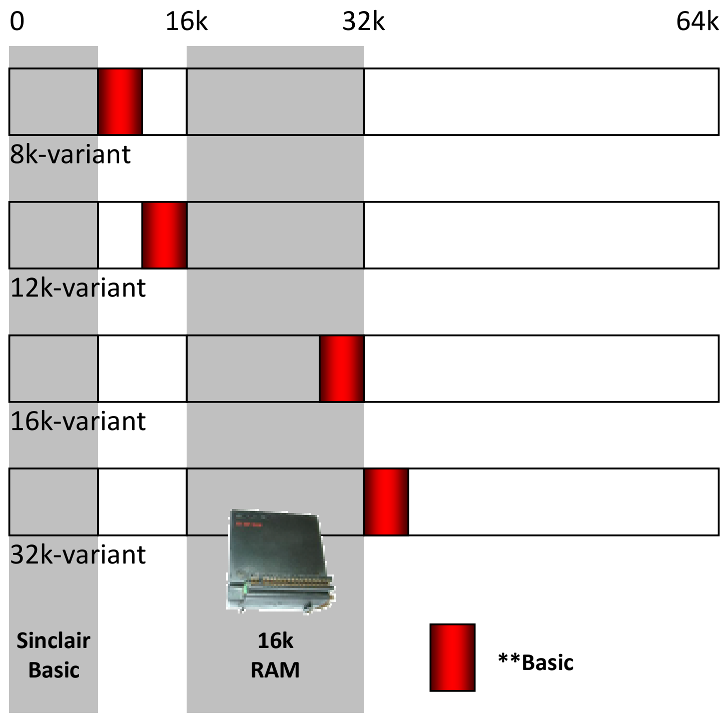
About 12kByte RAM will be left over for Basic.

PB32K.P

This variant loads above 32k. To run a machine program there you have to have a M1NOT circuit installed in your ZX81! The whole 16k RAM will be free for Basic.

memory map

The following diagram shows the memory-usage of the different variants.



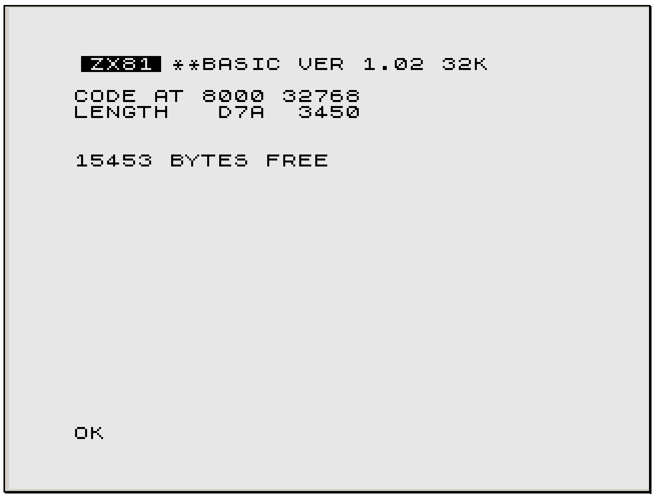
*start of **Basic*

Installation

After loading the program you have to start it with RUN.

**Basic then installs itself into memory and is ready to use. To clear the memory simply execute NEW.

After executing NEW **Basic will display its status-screen, which shows the verison of **Basic and the left memory.



```
ZX81 **BASIC VER 1.02 32K
CODE AT 8000 32768
LENGTH    D7A  3450

15453 BYTES FREE




OK
```

You can also get this status-screen with **STATUS.

Battery backup memory

When using **Basic with battery-backup memory you can reactivate **Basic after a RESET or Power-Up by a call to its startadress. According to the used memory this is one of the following commands:

```
PRINT USR 8192
PRINT USR 12288
PRINT USR 32768
```

**Basic will then be ready to use again. When using the 16k variant this is not possible because the code might be overwritten by the memory-check routine of the Sinclair-ROM.

hotkeys

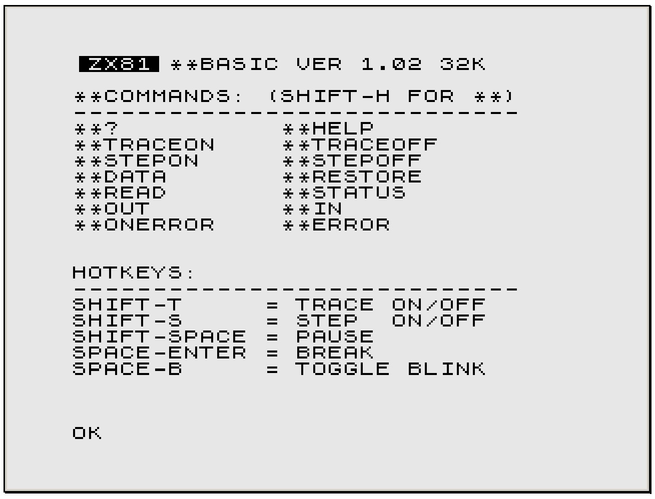
For the debug-options: BREAK, Pause, Trace and Singlestep there are the following hotkeys:

SHIFT + T	= Trace on/off
SHIFT + S	= Singlestep on
SHIFT + SPACE	= Pause
SPACE + NEWLINE	= BREAK
SPACE + B	= change blinkstyle of cursor

As long as SHIFT + SPACE is pressed the program will make a pause. Releasing the keys will make the program go on.

help

When entering **? (SHIFT-H and then SHIFT-C) you will get a helpscreen.



```
ZX81 **BASIC VER 1.02 32K
**COMMANDS: (SHIFT-H FOR **)
-----------------------------
**?             **HELP
**TRACEON       **TRACEOFF
**STEPON        **STEPOFF
**DATA          **RESTORE
**READ          **STATUS
**OUT           **IN
**ONERROR       **ERROR

HOTKEYS:
-----------------------------
SHIFT-T      = TRACE ON/OFF
SHIFT-S      = STEP  ON/OFF
SHIFT-SPACE  = PAUSE
SPACE-ENTER  = BREAK
SPACE-B      = TOGGLE BLINK


OK
```

It will give you a short reminder which commands and hotkeys are available in this version.

new commands

**?
**HELP

Shows all new command of **Basic.

**STATUS

Shows the version of **Basic and the left memory.

**TRACEON
**TRACEOFF

The command **TRACEON starts the trace-mode. With **TRACEOFF the trace-mode will be switched off again. When activated it shows the actually interpreted line on the right side of the screen. The linenumbers will scroll upwards to build a tracelist showing permanetly the last 20 lines.

**STEPON
**STEPOFF

With **STEPON you enter a singlestep mode. With **STEPOFF this mode will be deactivated. When activated the interpreter will pause after each line and waits for key S to be pressed. With SPACE you can deactivate the singlestep-mode at any time and the program will go on without pausing.

**DATA D1[, D2, D3, .. Dn]
**RESTORE [line]
**READ var

This command are for storing data within the Basic-lines. In each **DATA-line you can store one ore more strings or numerical values separated with commas. In a program there might be more than one **DATA lines which need not to be in a consecutive order.

**RESTORE lets the next **READ start with the selected line. If **RESTORE is used without a line then the first **DATA line in the program is selected. With **RESTORE you can read in different data lines in your program. Before any **READ is used make sure that a **RESTORE command is executed before.

**READ gets a value out of a **DATA line and copies it to the named variable. A numeric value can only be used with a numeric variable. A string value can only be used with a string variable.

When the last value of a **DATA was read the next **READ will get its value out of the following **DATA line. If there are no more data or the type of the data is wrong you will get an error.

**IN var, port
**OUT port, value

With this commands you can execute a read or write to Z80 I/O-ports. The port-adress might be in a 16-bit range.

Here the **IN is made as a command and therefore needs to have a numeric variable!

**ONERROR line

With **ONERROR you can handle runtime-errors with your Basic-program. All errors except D (BREAK) and 9 (STOP) will be bypassed and will not cause the program to terminate. In case of an error the program will do a GOTO to the given line at the **ONERROR command. The number and line in which the error occured will automatically be stored in the variables ERRNUM and ERRLINE. This can be used to handle the error.

After handling the error you can go on with your program by using CONT.

With **ONERROR 0 you can disable the errorhandling and the next error will terminate your program.

Be careful to make your errorhandler free of errors. Or you might get an endles loop then. In this case the trace-mode can be a helpful tool.

**ERROR errornumber

With **ERROR you can generate an error with precise errornumber in your program. Other than in Sinclair-Basic you can generate not only error numbers 0 to 15 (F) but also numbers 16 to 35 making error G to Z possible.

This is useful for debugging purpose or to test and trigger the errorhandler.

compatibility

Sinclair Basic

All Basic commands of the Sinclair 8k-ROM are still valid. Much of the ROM code like the editor, the arithmetics, the videosystem and the character-set is used by **BASIC. Therefore **BASIC looks and feels like the Sinclair Basic does.

All the functions and positions of the system variables, the Basic lines, the video-screen and the Basic-variables are identically used.

A program written with **BASIC is compatible to Sinclair Basic as long as no **-command is used. If a **-command is tried to execute in Sinclair-Basic it will only stop with an error.

compatibility with HRG-ms version 2.7

**Basic-16k can not be used with HRG-ms. Both programs require the same memory over RAMTOP.

**Basic-32k can be used with some restrictions with HRG-ms. Graphic-bank 5 must not be used, because it will overwrite **Basic there.

**Basic-8k can be used with HRG-ms without any restrictions. All graphic-banks can be used.

compatibility with other programs

Pure Basic-programs can be run with **Basic without any restrictions.

Programs written in machine-code can be run as well. But like with Sinclair-Basic these programs might cause problems or crash or even make **Basic crash.

example programs

Trace

In this example you can view an unpredictable program going.

```
  10 REM TESTPROGRAMM
  20 TRACEON
  30 GOTO 100+500*RND
 100 GOTO 30
 200 GOTO 30
 300 GOTO 30
 400 GOTO 30
 500 GOTO 30
1000 TRACEOFF
1010 PRINT "FERTIG"
```

SingleStep

Here you can singlestep the program after half the loop is done.

```
  10 REM TESTPROGRAMM
  20 TRACEON
 100 FOR X=0 TO 100
 150 IF X=50 THEN STEPON
 200 PRINT ".";
 300 NEXT X
 320 STEPOFF
 340 TRACEOFF
1010 PRINT "FERTIG"
```

When in singlestep-mode each key S will execute one next line. If you press SPACE the singlestep-mode will be terminated and the program continues in normal manner.

Data, Restore, Read

This program shows the use of **READ.

```
  10 REM TESTPROGRAMM
  20 DATA 3
  30 DATA "PAUL",27
  40 DATA "ESTHER",41
  50 DATA "PETER",35
1000 REM
1010 RESTORE 20
1020 READ END
1030 FOR M=1 TO END
1040 READ N$
1050 READ AGE
1060 PRINT N$;" IST ";AGE
1070 NEXT M
```

At first it gets the number of the valid datas and then reads the data in a loop. Here the data is a mix of strings and numbers.

Onerror

The following is an example how to catch a wrong input with **ONERROR. It makes a simple numeric calculator than will not stop executing on a not numerical or overflowed arithmetical expression.

To stop the program you have to enter STOP or make a BREAK when the program prints a value.

```
  10 REM CALCULATOR
  20 ONERROR 100
  30 GOTO 1000
 100 PRINT "ERROR ";ERRNUM
 110 SCROLL
 110 GOTO 1020
1000 REM
1010 SCROLL
1020 INPUT A
1030 PRINT A
1040 GOTO 1010
```

Error

The following one generates error 29 (T) if you wait too long. If you press B it will terminate with error 13 (D or BREAK).

```
  10 REM ERROR-DEMO
1000 FOR X=0 TO 500
1010 IF INKEY$="B" THEN ERROR 13
1020 NEXT X
1030 ERROR 29
```