ZX81 **Basic Version 1.50

for ZXpand

```
10 ONERROR 60
20 RESTORE
30 DATA 1,2,7
40 READ A,B
50 GOTO 40
60 STOP
```

Eigenschaften

**Basic ist eine Erweiterung für das ZX81-Basic, das auf einem ZX81 mit ZXpand lauffähig ist.
Mit **Basic bekommt das ZX81-Basic neue Befehle, die mit dem Mathematik-Symbol des ZX81-Basic ** für Power bzw. Potenz eingegeben werden. Im Listing wird das ** nicht dargestellt, so dass alle Befehle gut lesbar erscheinen.

Properties

**Basic is an extender for the ZX81-Basic, that can be used on a ZX81 with ZXpand.
With **Basic you will have new ZX81-Basic commands that all start with the ZX81 symbol ** like the mathematical power function. The symbol ** will not be shown in the listing to make the new commands look nice.

```
10 **ONERROR
```

Zur Eingabe eines neuen Befehls muss zuerst SHIFT-H eingegeben werden. Dies ergibt das Zeichen ** in der Editor-Zeile. Danach kann der neue Befehl in einzelnen Buchstaben eingegeben werden. Durch diese Syntax ist ein Programm, das unter **Basic geschrieben wurde, auch mit dem normalen ZX81 Basic oder anderen PC-Programmen (ZXtool, ZX81List, etc.) als Listing darstellbar. Dort wird allerdings das ** nicht unterdrückt und erscheint im Listing.
Die neuen Befehle können im Direktmodus, im Zeileneingabemodus und natürlich zur Programmlaufzeit benutzt werden. Ein neuer Befehl wird allerdings bei der Eingabe nicht auf Syntaxfehler untersucht. Fehler werden erst im Programmlauf oder bei der Abarbeitung im Direktmodus festgestellt.
Zusätzlich zu den neuen Befehlen gibt es in **Basic einige Hotkeys, mit denen das Verhalten von **Basic während des Programmlaufs gesteuert werden kann.
Eine weitere Änderung ist, dass ein Programm-Abbruch bzw. BREAK nur noch mit SPACE + NEWLINE erfolgt. Durch dieser Änderung ist das Leerzeichen oder das £-Zeichen über INKEY$ in Programmen nutzbar.
Im Gegensatz zum original Sinclair-Basic hat **Basic nun auch einen blinkenden K-, L-, F- und G-Cursor.
Zusätzlich dazu sind die Fehlermeldungen erweitert. **Basic meldet sich beispielsweise nach einem fehlerfreien Programmlauf statt mit „0/10“ nun mit „OK“. Im Fehlerfall meldet es statt „2/10“ nun „ERROR 2 IN LINE 10“.
Damit kann man sofort sehen, dass **Basic aktiv ist.

To start entering one of the new commands 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 + NEWLINE pressed together. So you can have SPACE or £ be entered with 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.

Systemvoraussetzungen

Für **Basic wird ein ZX81 mit mindestens 16k RAM und entweder 8k RAM im Adressraum 8k-16k oder M1-fähiger RAM im Adressraum 32k-40k benötigt. Ein ZXpand ab ROM-Version 6.7 ist dazu hinreichend.
Eine HRG-fähigkeit des Speichers ist für die Funktion von **Basic nicht notwendig. Selbstverständlich kann **Basic mit HRG-Programme zusammen genutzt werden, solange es keine Adresskonflikte mit HRG-Daten oder HRG-Programmteilen gibt.
**Basic läuft kann auch auf dem Emulator „EightyOne“ bei entsprechend eingestellten Hardware-Optionen für ROM und RAM genutzt werden. Allerdings gibt es unter „EightyOne“ gewisse Funktionseinschränkungen, die beim Einsatzt der realen Hardware nicht auftreten.

Requirements

For **Basic you need a ZX81 with at least 16k RAM and either 8k RAM between 8k-16k or M1 capable RAM above 32k. ZXpand with version 6.7 or newer will be sufficient.
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“ as long as all needed hardware-options for ROM and RAM are set correctly. Unfortunately there are some issues with "EightyOne" that do not occur on the real hardware.

Versionen und Speicherbelegung

Es gibt zwei Varianten von **Basic, die für die unterschiedliche Speicherbereiche generiert sind.

PB8K.P
Diese Variante läd sich in den Speicherbereich zwischen 8k und 16k.

PB32K.P
Diese Variante läd sich in den Speicherbereich über 32k. Um dort ein Maschinenprogramm laufen zu lassen, muss eine M1NOT-Schatung im ZX81 integriert sein!

Versions and Memory

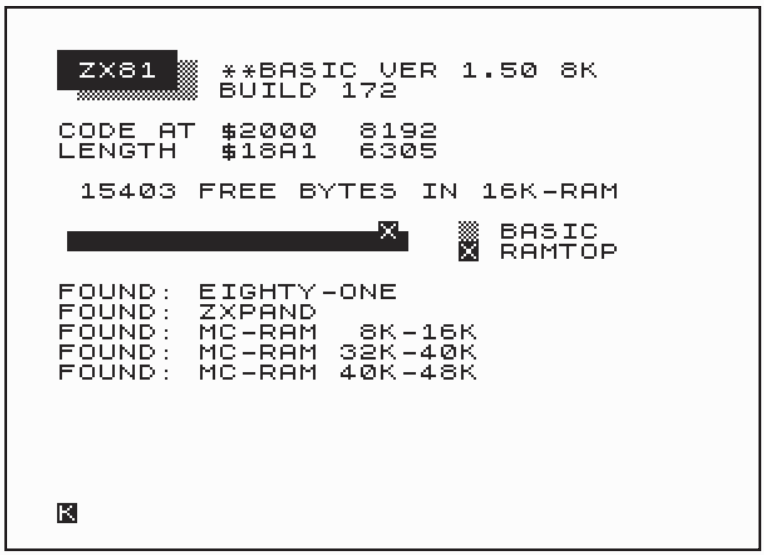
There are two variants of **Basic which are made for different memory-adresses.

PB8K.P
These variant loads between 8k and 16k.

PB32K.P
This variant loads above 32k. To run a machine program there you must have a M1NOT circuit installed in your ZX81!

Start von **Basic

Nach dem Laden der entsprechenden Programmvariante startet diese automatisch und **Basic installiert sich in den vorgesehenen Speicherbereich. Wenn der Speicherbereich nicht verfügbar sein sollte oder nicht für die Programmausführung von Maschinenprogrammen geeignet ist, dann meldet das Installationsprogramm einen entsprechenden Fehler und bricht ab.
**Basic meldet sich nach dem Starten mit seiner Statusmeldung, die die Version von **Basic und den für Basic verfügbaren Speicher anzeigt.
Zusätzlich zeigt **Basic hier an, welche Hardware es bei der Installation vorgefunden hat.
Diese Statusmeldung kann auch jederzeit mit dem Befehl **STATUS angezeigt werden.



Starting **Basic

After loading the program it starts automatically and **Basic then installs itself into the memory. If the memory is not present or not sufficient for running machinecode the installer stops with an errormessage and quits.
After the start **Basic will display its status-screen, which shows the verison of **Basic and the free memory for the basic.
Additionally this screen shows all detected hardware that was found at the installation.
You can get this status-screen antytime by entering **STATUS.

Änderungen und Erweiterungen

Statusbildschirm
Der Statusbildschirm zeigt eine Übersicht des 16k Basic-Speichers. Dabei wird neben der Anzahl der freien Bytes in einem Diagramm die Belegung des Speichers mit Basic-Zeilen, Basic-Variablen und RAMtop dargestellt.
CAT
Der CAT-Befehl von ZXpand zeigt jetzt auch die Dateilängen und ggf. ein gesetztes RO-Flag der Dateien an.
NEW
Der NEW Befehl zeigt nach Ausführung die Statusseite an.

Automaties Verzeichnis
Wenn kein Programm im Basic-Speicher geladen ist, wird bei Eingabe von ENTER bzw. NEWLINE das aktuelle Verzeichnis auf der SD-Karte angezeigt.
Anführungszeichen
Bei den Befehlen **LOAD, **SAVE, **DIR, **CD und **DELETE müssen keine Anführungszeichen um den Pfad- oder Dateinamen angegeben werden.

Changes and Additions

Status-screen
The status-screen shows an overview for the 16k Basic-memory. It shows the free bytes and a diagram to have an overview of the Basic-lines, the variables and RAMtop.

CAT
Now the CAT-command of ZXpand shows the filelength and the readonly-flag if set.
NEW
After a NEW command the status-screen will be shown automaticly.
Automatic Directory
If there are no Basic lines in the memory then pressing the ENTER or NEWLINE key will show the current directory of the SD-card.
Quotes
The commands **LOAD, **SAVE, **DIR, **CD und **DELETE do not need quotes for the file or pathname. If a filename uses special characters then qoutes can be used.

Variablen

**Basic benutzt automatische Variablen. Diese können nicht gelöscht oder überschrieben werden bieten aber nützliche Informationen für ein Basic-Programm an.
ERRNUM
Enthält die Nummer des letzten aufgetretenen Fehlers. Oder den Wert 0, wenn es keinen Fehler gab. Kann in der Fehlerbehandlung von ONERROR genutzt werden, um verschiedene Reaktionen auf die einzelnen Fehlerarten zu generieren.
ERRLINE
Enthält die Zeilennummer, in der der letzte Fehler auftrat. Oder den Wert 0, wenn es ein Fehler in der Eingabezeile war.
DFILE
Enthält die aktuelle Adresse des Text-Bildschirm Speichers. Die Länge des Speicherbereichs ist normalerweise 24 x 33 = 792 Bytes. Diese Adresse kann für PEEK, POKE oder als Startadresse für **BSAVE oder **BLOAD genutzt werden.
HRGFILE
Enthält die aktuelle Adresse des HRG-Bildschirm Speichers. Die Länge des Speicherbereichs ist normalerweise 192 x 32 = 6144 Bytes. Diese Adresse kann für PEEK, POKE oder als Startadresse für **BSAVE oder **BLOAD genutzt werden. Wenn das HRG-Modul nicht verfügbar ist, enthält die Variable den Wert 0.
FDATAS
Enthält die Anzahl der Daten in einer mit **FOPEN geöffneten Datei. Wenn die Datei leer oder nicht existent ist, dann enthält die Variable den Wert 0.

Variables

**Basic uses some automatic variables. You can not delete or change the values of this variables. For a Basic-program the variables can be useful.
ERRNUM
will hold the number of the last error that occured. If there was none then it is 0. You can use it for a differentiated reaction when using ONERROR.

ERRLINE
has the line number where the last error occured. Or will be 0 if the error was in the entryline.

DFILE
shows the adress of the text-screen. The regular screensize is 24 x 33 = 792 bytes. You can use this for PEEK and POKE into the screen or as start address for **BSAVE or **BLOAD commands.

HRGFILE
shows the actual address of the HRG-screen. The regular screensize is 192 x 32 = 6144 bytes. You can use this for PEEK and POKE into the screen or as startaddress for **BSAVE or **BLOAD commands. If there is no active HRG-module then it will have the value 0.

FDATAS
stores the number of data values of a file after opening it with **FOPEN. If the file is empty or does not exist then its value is 0.

HotKeys

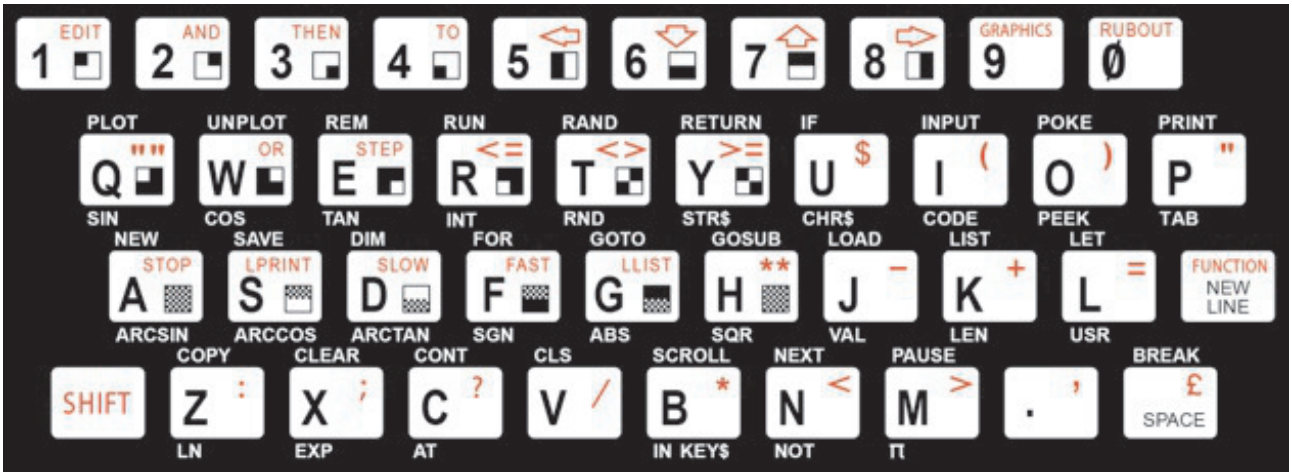
SHIFT + T	= Trace an/aus
SHIFT + S	= Singlestep an
SHIFT + SPACE	= Pause
SPACE + NEWLINE	= BREAK
SPACE + B	= Blinkstil des Cursors ändern

Solange SHIFT + SPACE gleichzeitig gedrückt wird, hält die Abarbeitung von Programmzeilen an. Beim Loslassen der Tasten läuft das Programm normal weiter.

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 are pressed simultaneously the program will generate a pause. Releasing the keys will make the program continue running.



Neue Befehle

**?
**HELP
Zeigt alle Befehle von **Basic am Bildschirm an.

**STATUS
Gibt die Version von **Basic und den momentan freien Speicherbereich aus. Die angezeigt Speichermenge ist der verbleibende Adressraum zwischen den Variablen und dem Prozessorstapel.

**TRACEON
**TRACEOFF
Der Befehl **TRACEON startet den Trace-Modus. Mit **TRACEOFF wird der Trace-Modus wieder abgeschaltet. Im Trace-Modus wird die aktuell abgearbeitete Programmzeilen-Nummer am rechten Bildschirmrand angezeigt. Die Zeilennummern rollen dabei automatisch nach oben, so dass immer die letzten 20 Zeilennummern im Bildschirm ablesbar sind.

**STEPON
**STEPOFF
Der Befehl **STEPON startet den Singlestep-Modus. Mit **STEPOFF wird der Singlestep-Modus wieder abgeschaltet. Es wird jeweils nur eine Programmzeile ausgeführt und dann automatisch eine Pause gemacht. Nach Drücken von S wird jeweils eine weitere Zeile abgearbeitet. Der Singlestep-Modus kann mit SPACE frühzeitig beendet werden.

New Commands

**?
**HELP
Shows all command of **Basic on the screen.

**STATUS
Shows the version of **Basic and the remaining memory. This is calculated as the difference between calculator stack and variables.

**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 line numbers will scroll upwards to build a tracelist showing permanently the last 20 lines.

**STEPON
**STEPOFF
With **STEPON you start the single-step mode. With **STEPOFF this mode will be switched off again. 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 [Zeilennummer]**
****READ Variable[,Variable, ...]**

Diese Befehle dienen der Datenspeicherung. In der **DATA-Zeile können ein oder mehrere String- oder numerische Ausdrücke mit Komma getrennt stehen. In einem Programm können mehrere **DATA-Zeilen stehen. Die einzelnen **DATA-Zeilen müssen dabei nicht direkt aufeinanderfolgen. Es können beliebige andere Basic-Zeilen zwischen den **DATA-Zeilen stehen.

**RESTORE stellt den Lesezeiger auf eine **DATA-Zeile. Ist keine Zeilennummer angegeben so wird die erste **DATA-Zeile im Programm selektiert. Durch Angabe einer Zeilennummer kann der darauffolgende **READ auf unterschiedliche Daten zugreifen. Vor der Benutzung von **READ muss ein **RESTORE ausgeführt sein.

**READ liest einen oder mehrere Werte aus der **DATA-Zeile in die angegebene Variable ein. Eine Stringvariable kann nur mit einem String-Ausdruck eingelesen werden. Eine numerische Variable kann nur mit einen numerischen Ausdruck eingelesen werden.

Ist der letzte Ausdruck einer **DATA Zeile eingelesen so wird der folgende **READ den ersten Ausdruck aus der daraufolgenden **DATA-Zeile einlesen. Wenn keine weiteren Daten folgen, oder ein falscher Datentyp benutzt wird, bricht **READ mit einer Fehlermeldung ab.

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

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

**RESTORE lets the next **READ start with the selected line. If **RESTORE is used without a line number then the first **DATA line in the program is selected. Before any **READ is used make sure that a **RESTORE command is executed before.

**READ gets one or more values 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 Variable, Port**
****OUT Port, Value**

Mit diesen Befehlen können Portein- und Portausgaben über Z80 I/O-Adressen ausgeführt werden. Die Portadresse kann dabei als 16 Bit Wert angegeben werden.

Hier ist **IN ein Befehl und keine Funktion und benötigt deshalb die Angabe eines Variablennamens!

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

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

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

****ONERROR Zeilennummer**

Mit dem **ONERROR Befehl können Laufzeitfehler im Basic-Programm behandelt werden. Alle auftretenden Fehler außer D (BREAK) und 9 (STOP) werden damit abgefangen und führen nicht nicht mehr zu einem Programmabbruch. Tritt ein Fehler auf, so wird zu der in dem **ONERROR Befehl definierten Zeilennummer wie mit einem GOTO verzweigt. Die Fehlernummer und die Zeilennummer, in der der Fehler auftrat, werden dabei automatisch in den Variablen ERRNUM und ERRLINE gespeichert und können zur Fehlerbehandlung benutzt werden.

Nach Behandlung des Fehlers kann zu dem unterbrochene Programmteil mit CONT zurückgekehrt werden.

Mit **ONERROR 0 wird die Option beendet und alle weiteren Fehler führen dann wieder zu einem Programmabbruch.

Vorsicht! Ein Laufzeitfehler in den Programmzeilen, die einen Fehler behandeln sollen, kann zu einer Endlosschleife führen. Zur Kontrolle ist hier die Trace-Funktion hilfreich.

****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 Fehlernummer**
Mit dem **ERROR Befehl kann ein Laufzeitfehler im Basic-Programm mit einer genau definierten Fehlernummer generiert werden. Es können nicht nur, wie im Basic üblich, die Fehler 0 bis 15 bzw. F erzeugt werden sondern es können auch zusätzlich die Fehlernummern 16 bis 35 entsprechend den Fehlercodes G bis Z ausgelöst werden.
Dies kann zum Debuggen nützlich sein oder zum Auslösen und Testen der Onerror Funktion.

****ERROR errornumber**
With **ERROR you can generate an error with precise error number 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.

****SAVE Name[,Zeilennummer]**
Speichert das aktuelle Basic-Programm auf die SD-Karte. Der Name muss nicht, kann aber in Anführungszeichen stehen.
Wenn eine Zeilennummer angegeben ist, dann wird ein Autostart bei der angegebenen Zeile angelegt ohne dass aus dem Programmlauf gespeichert werden muss. Wenn dort eine 0 angegeben ist, wird ein Autostart beim Speichern aus einem Programm heraus verhindert.

****SAVE Name[,Linenumber]**
Saves the actual Basic-program into a file on the SD-card. The filename needs not to be in quotes.
If a line number is given then the program will be stored with an autostart at this line. If the linenumber is given a 0 then there will be no autostart even if the command comes from a running program line.

****LOAD Name[,Zeilennummer]**
Läd die angegebene Datei von der SD-Karte in den Basic-Speicher. Der Name muss nicht, kann aber in Anführungszeichen stehen.
Wenn Linenum angegeben ist, dann wird ein Autostart bei der angegebenen Zeile ausgeführt. Der Autostart wird auch dann ausgeführt, wenn dies im abgespeicherten Programm nicht vorgesehen war. Wenn dort eine 0 angegeben ist, wird ein Autostart immer verhindert.

****LOAD Name[,Linenumber]**
Loads the named file from SD-card into the Basic memory. The filename needs not to be in quotes.
If a linenumber is given then the program will execute an autostart at this line. This can be done even if the stored program has no autostart in it. If the line number is given as 0 then there will be no autostart after load at all.

~~BLOAD Name,Adresse[,Länge]~~**
Läd die angegebene Datei als binäre Daten an die angegebene Adresse in den Speicher. Wenn eine Länge angegeben ist, dann werden dementsprechend maximal so viele Daten geladen. Wenn keine Länge angegeben ist, dann wird die komplette Datei geladen.
Als Adresse kann hier z.B. die Variable DFILE oder HRGFILE genutzt werden.

~~BLOAD Name,Address[,Length]~~**
Loads the given file as binary data to the memory at the given address. If the length is given then this will be the maximum data to be loaded. If the lenght is not given then the complete file will be loaded.
Here you can use the variables DFILE or HRGFILE as addresses for loadings screens.

~~BSAVE Name,Adresse,Länge~~**
Schreibt den Speicherinhalt ab der angegebenen Adresse mit der angegebenen Länge als binäre Daten in die angegebene Datei.
Als Adresse kann hier z.B. die Variable DFILE oder HRGFILE genutzt werden.

~~BSAVE Name,Adresse,Länge~~**
Writes a file with binary data beginning at the given address and having the given length.
Here you can use the variables DFILE or HRGFILE as addresses for storing screens.

****DIR [Pfad]**
Zeigt das aktuelle oder mit dem Pfad angegebene Inhaltsverzeichnis an.
Der Pfad muss nicht, kann aber in Anführungszeichen stehen.

****DIR [Path]**
Shows the actual directory or the directory of the given path. Path might be but needs not to be in quotes.

****CD Pfad**
Wechselt in das mit Pfad angegebene Verzeichnis. Der Pfad muss nicht, kann aber in Anführungszeichen stehen.

****CD Path**
Changes to the given path. Path might be but needs not to be in quotes.

~~MD Pfad~~**
Erzeugt das mit Pfad angegebene Verzeichnis. Der Pfad muss nicht darf aber in Anführungszeichen stehen.

~~MD Path~~**
Creates a folder with name Path. Path might be but needs not to be in quotes.

DELETE Name
Löscht eine Datei oder ein Verzeichnis. Der Name muss nicht, kann aber in Anführungszeichen stehen.
Es muss immer die Extension mit angegeben werden. Ein Verzeichnis kann nur dann gelöscht werden, wenn es leer ist.

DELETE Name
Deletes a file or directory. The name might be but needs not to be in quotes.
There must be an extension in the name. A directory can only be deleted if it is empty.

~~FOPEN DateiRefNummer,Name[,Methode]~~
Öffnet die mit dem Namen angegebene Datei unter der vorgewählten DateiRefNummer. Generiert die Variable FDATAS. Darin steht die Anzahl der Daten. Bei 0 ist die Datei leer oder nicht existent.
Mit Methode = "B" kann die Datei im Binärmodus geöffnet werden. In diesem Modus können einzelne Bytes ohne Datenstruktur aus der Datei gelesen oder in sie geschrieben werden. Die Variable FDATAS zeigt dann die Dateilänge in Bytes an.
Wenn FOPEN mit DateiRefNummer = 0 aufgerufen wird, dass wird damit der Zugriff auf das Inhaltsverzeichnis erzeugt. Diese Datei kann nicht im Binärmodus geöffnet und nicht beschrieben werden! Die Datenstruktur ist dann immer ein Vielfaches von String und Nummer, entsprechend dem Dateinamen und der dazugehörenden Dateilänge.

~~FOPEN HdlNum,Name[,Methode]~~

~~FCLOSE DateiRefNummer~~
Schließt die mit der DateiRefNummer angegebene Datei. Wenn kein DateiRefNummer angegeben ist, dann werden alle geöffneten Datein geschlossen.

~~FCLOSE HdlNum~~

~~FRESTORE DateiRefNummer,Offset~~
Setzt den Lesezeiger auf den mit Offset angegebenen numerischen Wert oder String. Ist die Datei im Binärmodus geöffnet, dann wird der Lesezeiger auf das Byte mit dem entsprechenden Offset gesetzt.

~~FRESTORE~~

~~FREAD DateiRefNummer,Variable[,Variable...]~~
Ein FREAD mit einer numerischen Variablen läd den in der Datei abgespeicherten Zahlenwert. Im Binärmodus läd es den numerischen Wert des Bytes.
Ein FREAD mit einer String-Variablen läd den in der Datei abgespeicherten String. Im Binärmodus läd es das Zeichen des Bytes.

~~FREAD~~

~~FWRITE DateiRefNummer,Variable[,Variable...]~~
Ein FWRITE mit einem numerischen Wert schreibt den Zahlenwert in die Datei. Im Binärmodus schreibt es ein Byte mit dem entsprechenden Wert an das Ende der Datei.
Ein FWRITE mit einem String schreibt den String in die Datei. Im Binärmodus werden entsprechend der Stringlänge viele Bytes in die datei geschrieben.

~~FWRITE~~

~~Datenstrukturen~~
Die Datenstrukturen für numerische Werte und Strings sind wie hier beschrieben aufgebaut:
Numerischer Wert: 0x7E,Byte0,Byte1,Byte2,Byte3,Byte4
String: 0x4E,AnzHigh,AnzLow,Byte0,...,Byte(Anz-1)

~~Datastructures~~

PITCHF
PITCHC
PITCHM

Definiert die Methode, mit der die Tonhöhen für BEEP und PING angegeben werden.

Nach PITCHF wird eine Tonhöhe als Frequenz in Hz interpretiert. Dies ist die Standardeinstellung nach NEW. Der erlaubte Wertebereich liegt zwischen 25Hz und 9000 Hz.

Nach PITCHC wird eine Tonhöhe als Halbtonabstand zum mittleren C (C4) interpretiert. Der erlaubte Wertebereich ist -40 bis 61.

Nach PITCHM wird eine Tonhöhe in Halbtönen entsprechend der MIDI-Definition interpretiert. Der erlaubte Wertebereich ist 20 bis 121.

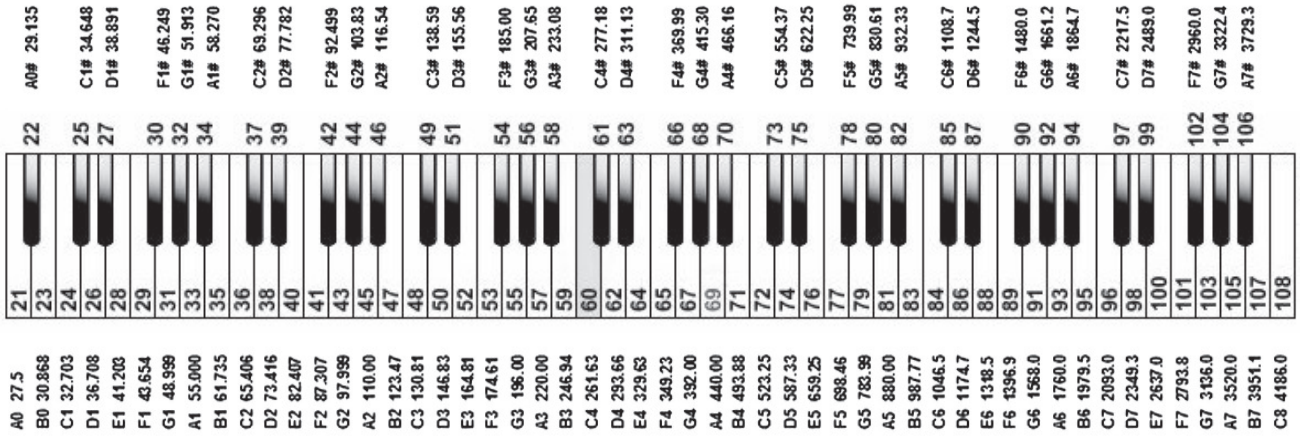
PITCHF
PITCHC
PITCHM

Defines to which scale the pitch for BEEP and PING refers.

After PITCHF the scale is in Hertz. This is the default after NEW. Allowed values are between 25Hz and 9000Hz.

After PITCHC the scale is halftone relative to middle C (C4). Allowed values are -40 to 61.

After PITCHM the scale is MIDI like. Allowed values are 20 to 121.



BEEP Zeit, Pitch

Erzeugt einen konstanten Ton für die angegebene Zeit in Sekunden. Wenn als Zeit ein negativer Wert angegeben ist, dann wird der Ton unbegrenz lange ausgegeben. Wenn als Zeit der Wert 0 angegeben ist, dann wird der Ton ausgeschaltet.

BEEP Time, Pitch

Makes a stable tone with the given pitch for the given time. If time is given a negative value then the tone will last. If time is given as 0 then any tone is switched off.

PING [Pitch]

Erzeugt einen abklingenden Ton. Wenn keine Tonhöhe angegeben ist, wird der Kammerton A (440 Hz, C4 +9, Midi 69) erzeugt.

PING [Pitch]

Makes a single ping with the given pitch. If no pitch is given then it uses standard pitch A (440Hz, C4 +9, Midi 69).

WAIT [Zeit]

Wartet die vorgegebene Zeit (0-500) in Sekunden. Wenn keine Zeit angegeben ist, wartet der Befehl solange, bis eine Taste betätigt wird. WAIT kann nicht im Direktmodus benutzt werden.

WAIT [Time]

Waits for the given time (0-500) in seconds. If no time is given then it waits until a key is pressed. WAIT must not be used in direct mode.

YM Register, Wert

Setzt das angegebene Register (0-15) des YM2149 mit dem angegebenen 8-bit Wert (0-255).

YM Register, Value

Sets the named register (0-15) of the YM2149 to the given 8-bits value (0-155).

YMX Register, Wert

Setzt das angegebene Doppelregister des YM2149 (0,2,4,11) mit dem angegebenen 16-bit Wert (0-65535). Hier dürfen nur die Register 0,2,4 und 11 benutzt werden.

YMX Register, Value

Sets the named double register (0,2,4,11) of the YM2149 to the given 16-bits value (0-65535). Only registers 0,2,4 and 11 can be used.

Beispielprogramme

TRACEON.P

In dem Programm TRACEON.P wird gezeigt, wie die Trace-Funktion arbeitet. Während des Programmlaufs kann mit der Tastenkombination SHIFT+SPACE der Programmfluss temporär angehalten werden. Mit SHIFT+S kann der Singlestepmodus ebenfalls aktiviert werden.

Examples

TRACEON.P

Here you find TRACEON.P where you see how the trace mode can be activated. With SHIFT+SPACE you can pause the program temporarily. With SHIFT+S you can enter the single step mode and watch the program jump around.

```
   1 REM
   2 REM
   3 REM **TRACEON
   4 REM
   5 REM
  10 TRACEON
1000 GOTO 1000 + 8000*RND
2000 GOTO 1000 + 8000*RND
3000 GOTO 1000 + 8000*RND
4000 GOTO 1000 + 8000*RND
5000 GOTO 1000 + 8000*RND
6000 GOTO 1000 + 8000*RND
7000 GOTO 1000 + 8000*RND
8000 GOTO 1000 + 8000*RND
9000 GOTO 1000 + 8000*RND
```

MENU.P

Das Programm MENU.P ermöglicht ein automatisches Laden von z.B. **Basic nach einem Reset oder Kaltstart von ZXpand. Dazu muss es mit der Autostart-Option des **SAVE Kommandos abgespeichert werden.
Nach einem Reset sucht ZXpand automatisch nach einem Programm mit dem Namen "MENU.P" und startet dieses. Dieses Programm startet daraufhin "PB8K.P" wenn keine Taste gedrückt wird.

MENU.P

This little program MENU.P gives you the option to have **Basic start automatically on a reset or power-up. You have to save this program with the autostart option of the **SAVE command.
After a reset or powering up ZXpand looks for a file with name "MENU.P" and loads this automatically. This one then loads "PB8K.P" then if no key is pressed.

```
1 IF INKEY<>"" THEN STOP
2 LOAD "PB8K"


**SAVE MENU.P,1
```

ONERROR.P

Das Programm ONERROR.P demonstriert mehrere neue Kommandos.

In den DATA Zeilen sind hier neben Strings auch numerische Werte abgelegt. Diese werden mit den READ Kommandos zuerst in eine String-Variable eingelesen. Hier wird ganz bewusst eine Fehlersituation provoziert, indem ein numerischer Wert als String gelesen werden soll.

Dieser Fehler führt dann in der darauffolgenden Fehlerbehandlung zu einem Sprung in den Bereich, der die numerischen Werte einliest.

Interessanterweise geht dabei der Wert, der den Fehler erzeugte, nicht verlohren!

ONERROR.P

The program ONERROR.P shows multiple new commands.

The DATA lines have strings and numerical values stored in. The first READ loop takes them all as strings. Here an error is programmed to happen. The errorhandler then generates the jump to the READ loop which gets the numerical values.

Remember that the first numerical value that generates the error is not lost at all and will be output like normal.

```
    1 REM
    2 REM **ONERROR
    3 REM **DATA
    4 REM **RESTORE
    5 REM **READ
    6 REM
   10 ONERROR 9000
1000 DATA "THE","QUICK"
1010 DATA "BROWN","FOX"
1020 DATA "JUMPS","OVER"
1030 DATA "THE","LAZY"
1040 DATA "DOG"
1050 DATA 1,2,3,4,5
1060 REM
2000 RESTORE
2010 READ A$
2020 PRINT A$
2030 GOTO 2010
2040 READ A
2050 PRINT A
2060 GOTO 2040
2070 REM
9000 IF ERRLINE = 2010 THEN GOTO 2040
9010 STOP
```