ZX81 **Basic Version 1.02 Handbuch

**Basic ist eine Erweiterung für das ZX81-Basic, die ohne Hardwareänderungen auf jedem ZX81 lauffähig ist.

Mit **Basic ist es möglicht , Befehle des ZX81-Basics zu ändern oder mit neuen Befehlen zu ergänzen.

Neuen Befehle werden mit dem Mathematik-Symbol des ZX81-Basic "**" für Power bzw. Potenz eingegeben. Im Listing wird das "**" unterdrückt, so dass der neue Befehl linksbündig erscheint.

Beispiel:

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

Basic erweitert dazu die Syntax um das neue Keyword "". Zur Eingabe eines neuen Befehls muss deshalb zuerst SHIFT-H gedrückt 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 noch 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 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 und gleichzeitig gedrücktem NEWLINE erfolgt. Mit dieser Änderung ist SPACE oder £ 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.

Systemvoraussetzungen

Für **Basic wird ein ZX81 mit mindestens 16k Speicher und originalem Sinclair 8k-Basic-Rom benötigt.

Zusätzlicher Speicher im Bereich 8k-16k oder über 32k (mit M1NOT-Schatung) kann für **Basic genutzt werden. Beim Einsatz von zusätzlichen Speicher bleibt der volle 16k Speicher für Basic frei. Der Speicher muss schreibbar sein, weil **Basic eigene Variablen dort ablegt. **Basic ist also nicht ROM-fähig.

Eine Grafikerweiterung 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“ (Version 1.0 10/03/08) bei entsprechend eingestellten Hardware-Optionen für ROM und RAM genutzt werden.

Speicherbelegung

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

PB8K.P
PB12K.P

Diese Varianten laden sich in den Speicherbereich zwischen 8k und 16k. Damit ist der komplette 16k Speicher für Basic nutzbar.

PB16K.P

Diese Variante ist auf jedem ZX81 lauffähig. Sie läuft komplett im 16k Speicher. Dazu erniedrigt sie automatisch RAMTOP und kopiert sich darüber. RAMTOP muss deshalb vorher auf den Originalwert für 16k Speicher stehen. Andernfalls bricht die Installation mit einem Fehler ab.

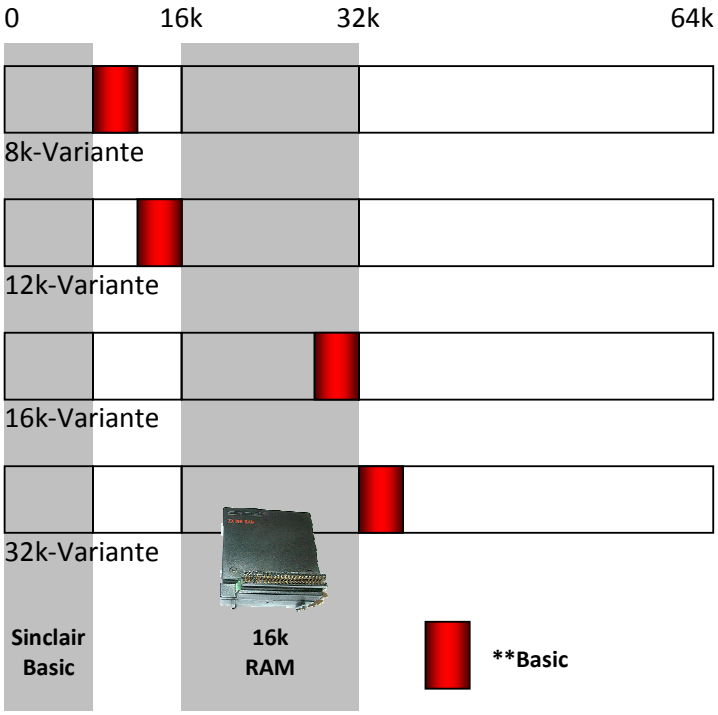
Für das Basic stehen nach der Installation ca. 12kByte zur Verfügung.

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! Für Basic steht dann der komplette 16k Speicher zur Verfügung.

Adressbelegung

Das folgende Diagramm zeigt die Lage der verschiedenen Varianten im Adressraum.



*Start von **Basic*

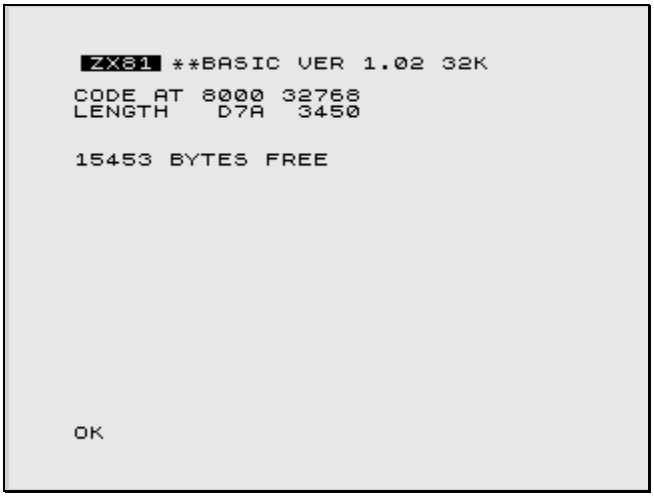
Installation

Nach dem Laden der entsprechenden Programmvariante muss diese mit RUN gestartet werden.

**Basic installiert sich damit automatisch in den vorgesehenen Speicherbereich und ist sofort aktiviert.

Danach kann das Basic-Ladeprogramm mit NEW gelöscht werden.

Nach jedem NEW meldet sich **Basic mit seiner Statusmeldung, die die Version von **Basic und den für Basic verfügbaren Speicher anzeigt.



Diese Statusmeldung kann auch mit dem Befehl **STATUS angezeigt werden.

Batteriegepufferter Speicher

Wenn **Basic in einem Batterie-gepufferten Speicher liegt, dann kann es nach einem Reset oder Power-Up ohne Nachladen direkt ab der Startadresse der installierten Variante wiedergestartet werden. Je nach benutzten Speicherbereich ist das dann einer der folgenden Befehle:

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

**Basic ist dann sofort wieder aktiv. Bei der 16k-Variante ist dies nicht möglich, weil sie nach einem Reset vom Speichertest überschrieben wird.

HotKeys

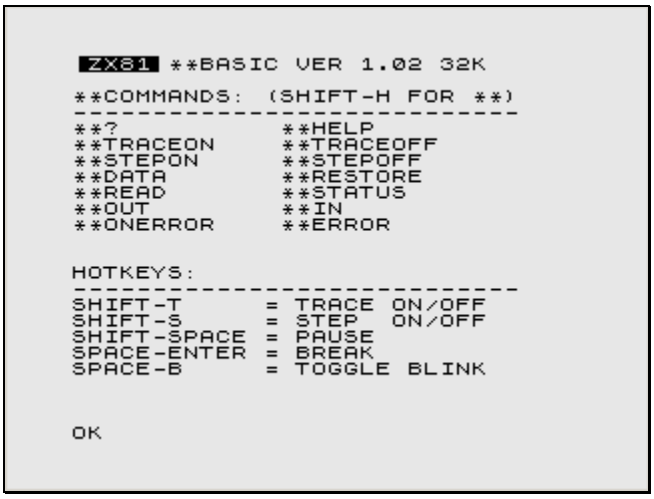
Für die Debug-Funktionen: BREAK, Pause, Trace und Singlestep gibt es die folgenden Hotkeys:

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

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

Hilfefunktion

Wird in der Eingabezeile **? (SHIFT-H und dann SHIFT-C) eingegeben, so wird die integrierten Hilfefunktion am Bildschirm angezeigt.



Damit wird kurz gezeigt, welche Befehle wie geschrieben werden und welche Hotkeys zur Verfügung stehen.

neue Befehle

****?**
****HELP**
Zeigt alle neuen 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 auch mit SPACE frühzeitig beendet werden.

****DATA D1[, D2, D3, .. Dn]**
****RESTORE [Zeilennummer]**
****READ Variable**
Diese Befehle dienen der Datenspeicherung. In der **DATA-Zeile können ein oder mehrere String- oder numerische Ausdrücke ggf. 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 den Wert eines Ausdrucks 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.

**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!

**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 werden 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.

**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.

Kompatibilität

Sinclair Basic

Alle Basic-Befehle des Sinclair 8k-Rom sind weiter nutzbar. Große Teile des Roms wie z.B. der Zeileneditor, die Arithmetik, die Videoausgaberoutine und auch der Zeichensatz werden von **Basic mitbenutzt. Deshalb gleicht **Basic vom grundsätzlichen Verhalten dem Sinclair-Basic.

Die Funktionen und die Lage von Systemvariablen, der Programmzeilen, des Bildspeichers und der Basic-Variablen ist identisch mit dem Sinclair-Basic.

Ein mit **BASIC geschriebenes Programm ist mit Sinclair-Basic lauffähig, solange keine **-Befehle benutzt werden. Ein **-Befehl führt unter Sinclair-Basic zu einer Fehlermeldung.

Kompatibilität mit HRG-ms Version 2.7

**Basic-16k kann nicht mit HRG-ms benutzt werden. Beide Programme liegen im selben Adressraum über RAMTOP.

**Basic-32k kann mit Einschränkungen HRG-ms benutzten. Die Grafik-Bank 5 darf nicht benutzt werden, weil in diesem Speicherbereich **Basic liegt.

**Basic-8k kann ohne Einschränkungen HRG-ms nutzen. Es könne alle Grafik-Bänke ohne Gefahr benutzt werden.

Kompatibilität mit anderen Programmen

Reine Basic-Programme sind uneingeschränkt unter **Basic lauffähig.

Maschinensprache-Programme können ebenfalls ausgeführt werden. Allerding können, wie im originalen Basic auch, Maschinenspracheprogramme möglicherweise die Funktionalität von **Basic behindern, stören oder **Basic sogar zum Absturz bringen.

Beispielprogramme

Trace

Der Lauf des folgenden Programms wird mit **TRACEON sichtbar gemacht.

```
  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"
```

Ohne Trace kann der Programmlauf in diesem Beispielprogramm nicht eindeutig vorhergesagt werden. Mit TRACE wird er sofort sichtbar.

SingleStep

In einigen Fällen ist es hilfreich, ein Programm zusätzlich zum TRACE Zeile für Zeile im Einzelschritt ausführen zu lassen und es dazu gezielt anzuhalten. Das ist mit **STEPON möglich.

```
  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"
```

Wenn x den Wert 500 erreicht, wird ab Zeile 150 jede weitere Zeile nur nach dem Druck der Taste S ausgeführt.

Mit einem Druck auf die Taste SPACE kann zu jeder Zeit der Einzelschritt frühzeitig beendet werden. Das Programm läuft ab dann wieder durchgängig weiter.

Data, Restore, Read

Das folgende Programm zeigt, wie **READ benutzt wird.

```
  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
```

Hier wird zuerst der numerische Wert für die Anzahl der Datensätze eingelesen. Danach werden in einer Schleife die Datensätze bestehend aus dem String für den Namen und dem numerischen Wert für das Alter eingelesen.

Onerror

Das folgende Beispiel zeigt, wie eine falsche Eingabe in einer INPUT-Zeile abgefangen werden kann.

```
  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
```

Um das Programm anzuhalten, muss entweder ein STOP eingegeben werden oder während der Zahlenausgabe ein BREAK ausgelöst werden. Bei allen anderen Fehleingaben läuft das Programm weiter.

Error

Im folgenden Beispiel wird ein Fehler mit dem Code 29 (T) erzeugt, wenn eine gewisse Zeit abgelaufen ist. Bei rechtzeitiger Betätigung der Taste B wird Fehler 13 (D bzw. BREAK) erzeugt.

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