Turbo-BASIC XL Referenz

über diese Referenz

Diese Turbo-BASIC-Referenz versteht sich als Ergänzung zur ATARI-BASIC-Referenz. Sie behandelt ausschließlich Befehle, die unter ATARI-BASIC nicht verfügbar sind.
Diese Referenz wurde unter Zuhilfenahme des 1. Happy-Computer ATARI-Sonderheftes erstellt.

über Turbo-BASIC XL

Turbo-BASIC XL wurde 1985 von Frank Ostrowski geschrieben und in der Happy-Computer im selben Jahr als Listing zum Abtippen veröffentlicht. Es ist zu 99% kompatibel zum ATARI-BASIC. Programme, die in ATARI-BASIC geschrieben wurden können meist problemlos unter Turbo-BASIC XL ausgeführt werden und werden dabei erheblich beschleuningt.
Da Turbo-BASIC XL aber nicht ATARI-BASIC ist, können Probleme bei folgenden Programmen auftreten:

- Programme, die sich fest auf den ATARI-BASIC-Speicherplan verlassen
- Programme, die sich fest auf die (langsame) Geschwindigkeit des ATARI-BASIC verlassen

Solche Programme sollten entweder an ATARI-Standards oder gleich an Turbo-BASIC XL angepasst werden.

Index

Befehle zur strukturierten Programmierung

IF

```
IF logischer Ausdruck:ENDIF
IF logischer Ausdruck:Befehle:ELSE:Befehle:ENDIF
```

Eine solche Strukturierung ist unter Turbo-BASIC XL möglich. Wird die Bedingung *logischer Ausdruck* erfüllt, so werden alle Befehle bis zum ELSE oder ENDIF ausgeführt. Ist die Bedingung *logischer Ausdruck* nicht erfüllt, so werden alle Befehle zwischen ELSE und ENDIF ausgeführt. Auf ELSE kann verzichtet werden, ENDIF muss gesetzt werden.

Beispiel:
```
IF STRIG(0)=0
     ? "Feuertaste gedrueckt"
ELSE
     ? "Feuertaste nicht gedrueckt"
ENDIF
```

REPEAT

REPEAT: UNTIL *logischer Ausdruck*
Dies ist eine Schleife, die so lange ausgeführt wird, bis die Bedingung *logischer Ausdruck* erfüllt ist. Die Schleife wird mindestens einmal ausgefuhrt, weil die Bedingung erst am Ende der Schleife geprüft wird.

Beispiel:
```
REPEAT
     ? "Feuertaste druecken"
UNTIL STRIG(0)=0
```

WHILE

WHILE *logischer Ausdruck*: WEND
Dies ist eine Schleife, die ausgeführt wird, so lange die Bedingung *logischer Ausdruck* erfüllt ist. Ist die Bedingung bereits zu Beginn der Schleife nicht erfüllt, so wird die Schleife gar nicht ausgeführt.

Beispiel:
```
WHILE STRIG(0)=1
     ? "Feuertaste druecken"
WEND
```

DO

DO: LOOP
Eine bedingungslose Endlosschleife. Alle Befehle zwischen DO und LOOP werden so lange ausgeführt, bis diese Schleife verlassen wird. Dies sollte mit dem Befehl EXIT geschehen.

Beispiel:
```
DO
     ? "Es gibt kein Entkommen!"
LOOP
```

EXIT

EXIT
Mit diesem Befehl werden Schleifen sauber verlassen.

Beispiel:
```
DO
     IF INKEY$="X" THEN EXIT
LOOP
```

***F+**

*F+
*F
Nach diesem Befehl sind >>FOR ... NEXT<<-Schleifen abweisend. Es wird also vor dem ersten Durchlauf der Schleife geprüft, ob der Zähler schon den Endwert erreicht hat. Ausgeschaltet wird diese Funktion mit *F-.

Beispiel:
```
*F+
FOR X=2 TO 1
     ? X
NEXT X
```
Da X bereits größer als der Endwert 1 ist, wird nach einem *F+-Befehl die Schleife nicht einmal durchlaufen.

PROC

PROC *unterprogramm_name*: ENDPROC
Zwischen diesen beiden Marken befindet sich ein Unterprogramm mit dem Namen *unterprogramm_name*.

Beispiel:
```
PROC unterprogramm_name
     ?"Dieses Unterprogramm wird nur ausgeführt, wenn es mit EXEC unterprogramm_name aufgerufen
wird!"
ENDPROC
```

EXEC

EXEC *unterprogramm_name*
Dies ist das Gegenstück zu PROC. Mit diesem Befehl wird das Unterprogramm aufgerufen.

Beispiel:
```
EXEC unterprogramm_name
?"Diese Zeile wird erst ausgegeben, wenn das Unterprogramm mit ENDPROC verlassen wurde!"
```

ON

ON *zahlen_wert* EXEC *unterprogramm_name_1*,*unterprogramm_name_2*,*unterprogramm_name_3*
Je nach dem Wert in der Variablen *zahlen_wert* wird das erste, zweite oder dritte (oder weitere) Unterprogramm aufgerufen.

Beispiel:
```
? "1. Addieren"
? "2. Subtraieren"
? "3. Multiplizieren"
? "4. Dividieren"
INPUT ZAHL
ON ZAHL EXEC ADDITION,SUBTRAKTION,MULTIPLIKATION,DIVISION
?"Diese Zeile wird erst ausgegeben, wenn das Unterprogramm mit ENDPROC verlassen wurde!"
```

#

#zeilen_name
GO *#zeilen_name*
ON *zahlen_wert* GO *#zeilen_name*
TRAP *#zeilen_name*
RESTORE *#zeilen_name*
Labeldefinition, das # entpricht dem PROC. In Turbo-BASIC XL lässt sich der unstrukturierte Sprungbefehl GOTO wenigstens dadurch lesbarer machen, dass >>GOTO zeilen_nummer<< durch >>GO *#zeilen_name*<< ersetzt wird. Auch bei TRAP und RESTORE können diese Marken verwendet werden.

Beispiel:
```
100 RESTORE #AUTOMARKEN
200 #AUTOMARKEN
210 DATA Ford,Volkswagen,Opel,SEAT,Wartburg
```

POP

POP
POP gilt für Unterprogramme mit GOSUB und EXEC ebenso wie für die Schleifen >>FOR ... NEXT<<, >>REPEAT ... UNTIL<<,

>>WHILE ... WEND<< und >>DO ... LOOP<<.

Beispiel:
```
100 PROC unterprogramm_name
110      POP :GOTO 200
120 ENDPROC
200 ?"Weiter im Programm..."
```

--

--
Eine besondere Form des REM-Befehls. Zeilen mit dieser -- Anweisung werden nicht beachtet. Wird eine solche Zeile mit LIST ausgegeben, so werden anstatt zwei ganze 30 Minuszeichen ausgegeben.

Beispiel:
```
100 --
LIST

100 ------------------------------
```

***L-**

*L-
In Turbo-BASIC XL werden Listungs formatiert ausgegeben, d.h. dass Unterprogramme (PROC ... ENDPROC) oder Schleifen (FOR ... NEXT, REPEAT ... UNTIL) jeweils um zwei Zeichen eingerückt werden. *L- schaltet diese Formatierung aus. Mit *L oder *L+ kann diese Formatierung wieder eingeschaltet werden.

Beispiel:
```
*L-

LIST
100 --

*L+
LIST

100 ------------------------------
```

neue Fehlermeldungen

ERROR - 22 ?NEST
Schachtelungsfehler, tritt auf, wenn das zu einem WHILE gehörende WEND nicht gefunden wird, oder das ENDIF zu einem IF, oder auch, nach >>*F+<<, das NEXT zu einem FOR. Beim Verlassen von Unterprogrammen (durch RETURN oder ENDPROC) werden Schleifen abgebrochen. Dies gilt sowohl - wie gewohnt - für die >>FOR ... NEXT<<-Schleife, wie auch für die anderen, unter Turbo-BASIC XL zur Verfügung stehenden Schleifen.

ERROR - 16 ?GOSUB
Zu einem RETURN fehlt GOSUB.

ERROR - 13 ?FOR
Zu einem NEXT fehlt FOR.

ERROR - 23 ?WHILE
Zu einem WEND fehlt WHILE

ERROR - 24 ?REPEAT
Zu einem UNTIL fehlt REPEAT.

ERROR - 25 ?DO
Zu einem LOOP fehlt DO.

ERROR - 28 ?EXEC
Zu einem ENDPROC fehlt EXEC.

ERROR - 29 ?GOSUB
Eine unbekannte Prozedur wurde aufgerufen.

ERROR - 30 ?#
Eine unbekannte Marke wurde verwendet.

ERROR - 27 ?XPROC
(Executing PROC). Diese Fehlermeldung tritt auf, wenn eine PROC-Anweisung ausgeführt wird. Prozeduren dürfen nur von EXEC aufgerufen werden.

ERROR - 26 ?EXIT
EXIT ohne Schleife.

ERROR - 15 ?DEL
Das GOSUB zu einem RETURN, NEXT zu einem FOR, REPEAT zu einem UNTIL ... wurde gelöscht. In Atari- und in Turbo-BASIC XL lassen sich Programme editieren, ohne Variablenwerte oder den Stapel zu zerstören. Wenn dann bei Rückkehr aus einem Unterprogramm (Schleife) die die entsprechende Zeile gelöscht oder verändert wurde, kann dieser Fehler passieren. Dies tritt auch auf, wenn ein in ein Programm eingebautes DEL sich selbst löscht. Alle Fehlernummern, die ATARI-BASIC ohne irgendwelche Texte ausgibt, werden in Turbo-BASIC XL grundsätzlich mit einem kurzen Text ergänzt (beispielsweise >>138 TIMEOUT<<, >>29 ?PROC<< etc.).Ausführliche erläuterungen können dem ATARI-BASIC Referenz Manual oder der DOS-Anleitung entnommen werden. Längere Texte würden, bei den insgesamt 60 zur Verfügung stehenden Fehlern, den Platzbedarf des Interpreters noch wesentlich erhöhen.

Befehle zur Programmbearbeitung

DEL

DEL *von_zeile*,*bis_zeile*
Löscht alle Programmzeilen von *von_zeile* bis *bis_zeile*.

RENUM

RENUM *alt_zeile*,*neu_zeile*,*schritt*
Nummeriert alle BASIC-Zeilen ab *alt_zeile* in *neu_zeile* um. *schritt* ist die Schrittweite, um die die jeweils nächste Zeile erhöht wird.

DUMP

DUMP
DUMP *datei_name$*
Gibt alle verwendeten Variablennamen und Unterprogrammnamen eines Programms auf dem Bildschirm aus. Die Ausgabe kann auch auf den Drucker (*datei_name$*="P:") oder in eine Datei umgeleitet werden.

TRACE

TRACE
TRACE+
Schaltet die TRACE-Funktion ein, d.h. die Nummer jeder ausgeführten Zeile wird auf dem Bildschirm ausgegeben. Mit TRACE- wird diese Funktion wieder ausgeschaltet.

*B

*B
*B+
Nach diesem Befehl wird das Drücken der BREAK-Taste wie ein Fehler behandelt, der mit TRAP abgefangen werden kann. Mit *B- wird dieser Befehl wieder aufgehoben. Nach RUN ist *B immer ausgeschaltet.

neue Befehle

DPOKE

DPOKE *speicher_adresse*,*zahlen_wert*
Doppel-Byte-POKE
Speichert den 16-Bit-Wert (zwei Byte) *zahlen_wert* in die Speicherzellen *speicher_adresse* und *speicher_adresse*+1. In ATARI-BASIC musste für die gleiche Funktion folgender Umweg gegangen werden:
POKE *speicher_adresse*,*zahlen_wert*-256*INT(*zahlen_wert*/256):POKE *speicher_adresse*+1,INT(*zahlen_wert*/256)

MOVE

MOVE *quell_speicher_adresse*,*ziel_speicher_adresse*,*groesse_speicher_block*
Blocktransfer
Kopiert *groesse_speicher_block* Bytes von *quell_speicher_adresse* nach *ziel_speicher_adresse*.

-MOVE

-MOVE *quell_speicher_adresse*,*ziel_speicher_adresse*,*groesse_speicher_block*
Blocktransfer, Rückwärts ausgeführt
Kopiert *groesse_speicher_block* Bytes von *quell_speicher_adresse* nach *ziel_speicher_adresse*. Jetzt wird allerdings erst das letzte zu verschiebene Byte kopiert um bei Überlappungen die Zerstörung des zu verschiebenen Inhalts zu verhindern.

BPUT

BPUT #datei_nummer,*quell_speicher_adresse*,*groesse_speicher_block*
Blockschreiben
Speichert *groesse_speicher_block* Bytes ab Adresse *quell_speicher_adresse* auf Diskette ab.

BGET

BGET #datei_nummer,*ziel_speicher_adresse*,*groesse_speicher_block*
Blocklesen
Liest *groesse_speicher_block* Bytes in Adresse *quell_speicher_adresse* von Diskette ein.

%PUT

%PUT *zahlen_wert*
Dieser Befehl speichert eine 6 Byte große Zahl auf Diskette ab.

%GET

%GET *zahlen_wert*
Zum Lesen von mit %PUT gespeicherten Zahlen.

FILLTO

FILLTO x,y
Kurzschreibweise, schneller und übersichtlicher als >>POSITION x,y:XIO 18,#6,0,0,"S:"<<

FCOLOR

FCOLOR n
Wählen der Farbe für FILLTO. In ATARI-Basic heißt dieser Befehl >>POKE 765,n<<.

CLS

CLS
CLS #6
Bildschirmlöschen
Mit diesem Befehl wird der Bildschirm gelöscht. Mit CLS #6 kann auch der Grafikbildschirm (bei GRAPHICS n, n>0) gelöscht werden. Entpricht dem drücken der Tastenkombination CONTROL-CLEAR bzw. ? CHR$(125) (oder ? #6;CHR$(125)).

PUT

PUT *zahlen_wert*
Gibt das Zeichen mit dem ASCII-Wert *zahlen_wert* auf dem Bildschirm aus. Entspricht dem Befehl ? CHR$(n).

GET

GET *zahlen_wert*
Wartet auf einen Tastendruck. Der ASCII-Wert der gedrückten Taste wird an *zahlen_wert* zurückgegeben. Für die gleiche Funktion musste in ATARI-BASIC erst ein Kanal für das Gerät "K:" (Keyboard, Tastatur) geöffnet werden.

DIM

DIM *variablen*
Im Gegensatz zu ATARI-BASIC werden alle Variablen gelöscht, die mit dem DIM-Befehl dimensioniert werden.

INPUT

INPUT "text";*variable*
INPUT "text",*variable*
Ein Text nach dem INPUT-Befehl erspart sonst nötige PRINT-Befehle. Folgt auf "text" ein Komma anstatt eines Simikolons, wird das manchmal störende Fragezeichen nicht mehr ausgegeben.

TEXT

TEXT x,y,*variable*
Druckt einen Text auf einem Grafikbildschirm aus. x und y bilden dabei die Position auf dem Bildschirm. Die Variable *variable* darf ein Zahlenwert oder eine String-Variable sein. Es ist nicht möglich, mehrere, durch Komma oder Semikolon getrennte Variablen zu verwenden.

CIRCLE

CIRCLE x,y,r
CIRCLE x,y,xr,yr
Zeichnet einen Kreis mit einem Radius von r Pixeln um den Punkt x,y. Wahlweise können auch Elipsen gezeichnet werden, dann müssen zwei Radien xr und xy angegeben werden.

PAINT

PAINT x,y
Füllt eine geschlossene Fläche mit der aktuellen Farbe. Dieser Befehl benötigt mitunter mehrere hundert Bytes, so dass es unter Umständen zur Fehlermeldung ERROR - 2 MEM kommen kann.

TIME$

TIME$=*string_variable$*
Setzt die interne Uhr des ATARI auf den Wert string_variable. Die Variable string_variable muss das Format HHMMSS (HH=Stunden, MM=Minuten, SS=Sekunden) beinhalten.

PAUSE

PAUSE *zahlen_wert*
Dieser Befehl hält die Ausführung des Programms für *zahlen_wert*/50 Sekunden an.

DSOUND

DSOUND stimmen,*frequenz,verzerrung,lautstaerke*
Ähnlich dem normalen SOUND-Befehl. Der ATARI kann jeweils zwei seiner vier Stimmen zu einer Zusammenfassen. Dann stehen ihm 65536 verschiedene Frequenzen (16bit Auflösung) zur Verfügung. Der Variable *frequenz* können also 65536 verschiedene Werte (0 bis 65535) zugeordnet werden.

SOUND

SOUND
DSOUND

Dieser Befehl ohne irgendwelche Werte schaltet alle Soundkanäle aus.

CLOSE

CLOSE
Der CLOSE-Befehl ohne Werte schließt alle I/O-Kanäle von #1 bis #7. Achtung! Es wird in den Grafik-Modi (3-15) auch der Kanal #6 geschlossen, so dass danach keine PLOT, DRAWTO oder CIRCLE Befehle mehr funktionieren. Der TEXT-Befehl funktioniert noch, da er nicht über IO

neue Rechenfunktionen und spezielle Variablen

DPEEK

DPEEK(*speicher_adresse*)
Doppel-Byte-PEEK
Gibt den 16-Bit-Wert der Speicheradressen *speicher_adresse* und *speicher_adresse*+1 zurück.

INKEY$

INKEY$
Enthält den Wert der gerade gedrückten Taste. Wird keine Taste gedrückt, so ist INKEY$="", ist also leer.

INSTR$

INSTR(*string_variable$,such_string_variable$*)
INSTR(*string_variable$,such_string_variable$,zahlen_wert*)
Ermittelt die Position des Suchtextes *such_string_variable$* innerhalb des Textes *string_variable$*. Auf Wunsch wird der Text *string_variable$* erst ab Position *zahlen_wert* durchsucht.

UINSTR$

UINSTR(*string_variable$*,*such_string_variable$*)
UINSTR(*string_variable$*,*such_string_variable$*,*zahlen_wert*)
Wie INSTR, nur werden jetzt Groß- und Kleinschreibung ignoriert.

ERR

ERR
Gibt den Code des zuletzt aufgetretenen Fehlers wieder, entspricht PEEK(195).

ERL

ERL
Gibt die Zeilennummer wieder, in der der letzte Fehler aufgetreten ist. Entspricht DPEEK(186), bzw. PEEK(186)+PEEK(187)*256 in ATARI-BASIC.

TIME

TIME
Enthält die Zeit in 50stel Sekunden und kann nicht direkt geändert werden (TIME=n ist nicht möglich).

TIME$

TIME$
Enthält die Uhrzeit, die mit TIME$="HHMMSS" eingestellt wurde.

FRAC

FRAC(*zahlen_wert*)
Ermittelt den Nachkommaanteil einer positiven oder negativen Zahl.
während -0.3-INT(-0.3)=0.7 ist, ist FRAC(-0.3)=-0.3

TRUNC

TRUNC(*zahlen_wert*)
Ermittelt den ganzzahligen Anteil einer positiven oder negativen Zahl.
während INT(-0.3)=1 ist, ist TRUNC(-0.3)=0

RND

RND
Die Random-Funktion kann in Turbo-BASIC XL abgekürzt werden (in ATARI-BASIC: RND(0)).

RAND

RAND(*zahlen_wert*)
Ermittelt eine Zufallszahl zwischen einschließlich 0 und ausschließlich *zahlen_wert*. Dabei kann der Wert *zahlen_wert* auch negativ sein.

HEX$

HEX$(*zahlen_wert*)
Wandelt die Dezimalzahl *zahlen_wert* in eine Hexadezimalzahl um. Die Zahl ist als zwei- (0-255) oder vierstellige (256-65535) Zahl formatiert.

DEC

DEC(*hex_zahl$*)
Wandelt die hexadezimale Zahl *hex_zahl$* in eine Dezimalzahl um. Beinhaltet *hex_zahl$* keine gültige Zahl, so ergibt das 0.

$xxxx

$0-$FFFF
Die Zahlen 0-65535 in hexadezimaler Schreibweise.

&

&
binäres UND

!

!
binäres ODER

EXOR

EXOR
binäres Exklusiv-ODER

DIV

zahlen_wert_1 DIV *zahlen_wert_2*
Division ohne Rest. Ist nicht immer gleich INT(*zahlen_wert_1*/*zahlen_wert_2*), ist aber immer gleich TRUNC(*zahlen_wert_1*/*zahlen_wert_2*)

MOD

zahlen_wert_1 MOD *zahlen_wert_2*
Bestimmung des Divisionsrestes, entspricht *zahlen_wert_1*-*zahlen_wert_2**TRUNC(*zahlen_wert_1*/*zahlen_wert_2*).

%0, %1, %2, %3

%0, %1, %2, %3
Die Zahlen 0 bis 3 in fest vordefinierten Konztanten. Diese nehmen weniger Speicherplatz ein, als die Zahlen 0 bis 3 (ein Byte statt 7 Bytes).

""

""
In Stringvariablen oder in mit PRINT oder TEXT ausgegebenen Texten können Anführungszeichen platziert werden, in dem die Anführungszeichen während der Zuweisung zweimal hintereinander geschrieben werden.

Beispiel:
```
? "Auf dem Schild stand ""Beladen verboten!""!"
ergibt
Auf dem Schild stand "Beladen verboten!"!
```

DOS-Befehle

DIR

DIR
DIR *pfad$*
Gibt das Inhaltsverzeichnis einer Diskette aus, auf Wunsch wird das Laufwerk (und der Pfad) in *pfad$* berücksichtigt. Wird *pfad$* nicht angegeben, so wird das Verzeichis von "D:*.*" ausgegeben.

RENAME

RENAME *datei_name$*
Benennt eine Datei um. *datei_name$* muss folgende gültige Werte besitzen: "alter_dateiname,neuer_dateiname".

DELETE

DELETE *datei_name$*
Löscht die Datei *datei_name$*.

LOCK

LOCK *datei_name$*
Schützt die Datei *datei_name$* vor dem Überschreiben.

UNLOCK

UNLOCK *datei_name$*
Hebt den Schreibschutz der Datei *datei_name$* wieder auf.

BLOAD

BLOAD *datei_name$*
Lädt eine Maschinenroutine in den Speicher.

BRUN

BRUN *datei_name$*
Lädt eine Maschinenroutine in den Speicher und startet diese, wenn eine RUN-Adresse in der Datei enthalten ist.

AUTORUN.BAS

Autostart
Mit Turbo-BASIC XL ist es möglich, nach dem Einschalten des Computers ein BASIC-Programm automatisch zu laden und zu starten. Dazu benennen Sie Ihr BASIC-Programm einfach in AUTORUN.BAS um. Vergessen Sie dabei nicht, dass das eigentliche Turbo-BASIC XL mit dem Namen AUTORUN.SYS auf der Diskette vorliegen muss.