EXTENDED BASIC für LASER 1101210/310

EXTENDED BASIC beschert Sie mit einer großen Zahl neuer und komfortabler BASIC-Befehle.

Auf der Diskette befinden sich zwei verschiedene Versionen, die sich in der Größe des benötigten Speichers, aber auch in der Anzahl der Befehle unterscheiden.

Die Kurzfassung mit einem Speicherbedarf von ca. 1200 Byte enthält die folgenden 30 Befehle:

AUTO, DELETE, TRON, TROFF, MEM, FRE, ERROR, ERR, ERL, DEFINT, DEFSGN, DEFDBL, DEFSTR, RANDOM, ON, RESUME, VARPTR, SYSTEM, STRING$, POS, VLOAD, MERGE, COMPRESS, RENEW, CINT, CSNG, CDBL, FIX, CALL, MEMSIZE. In der großen Version mit einem Speicherbedarf von ca. 3500 Byte sind zusätzlich die 9 folgenden Befehle enthalten:

RENUM, PLOT, NPLOT, CIRCLE, RECT, PAINT. CPOINT, GCLS, LPEN.

EXTENDED BASIC ist aufwärts-kompatibel mit der Original BASIC-Version Ihres Rechners und auch mit dem BASIC-UP; d. h. dort erstellte Programme können in EXTENDED BASIC übernommen, bearbeitet und auch gestartet werden.

In EXTENDED BASIC übernommene Programme sind unter dem normalen BASIC oder BASIC-UP nicht mehr ablauffähig.

Zwischen den beiden Versionen des EXTENDED BASIC können die Programme ausgetauscht werden. Sie sind in der Kurzversion nur ablauffähig, wenn keine der neun Sonderbefehle der großen Version verwendet wurden.

Geladen wird EXTENDED BASIC mit dem BRUN Befehl.

Es meldet sich nach erfolgreichem Laden mit einem kurzen Vorstelltext und der READY-Meldung. Anschließend können Sie mit der normalen BASIC Bearbeitung beginnen.

Die Eingabe der neuen BASIC-Befehle erfolgt normal, d. h. ohne jegliche Kennzeichnung.

Anmerkung:

Im nachfolgend beschriebenen Syntax der neuen Befehle bedeutet eine in eckige Klammern eingeschlossene Zeichenfolge ([xxx]) einen wahlfreien Eintrag.

Beschreibung der neuen Befehle

1. Zur Unterstützung der Programmbearbeitung **YLOAD** ["name"]

Mit diesem Kommando können Sie ein BASIC-Programm von Kassette lesen, das nicht unter EXTENDED BASIC erstellt wurde. Das Programm wird automatisch an den Adressbereich der jeweiligen Version angepasst. Nach dem Laden können Sie dieses Programm starten, listen, editieren und ggf. mit CSAVE in der EXTENDED BASIC-Fassung wieder auf Kassette speichern. So gespeicherte Programme können dann auch unter EXTENDED BASIC wie gewohnt mit CLOAD oder CRUN wieder geladen oder gestartet werden.

Achtung:

Alle nicht mit EXTENDED BASIC erstellten Programme müssen zunächst mit VLOAD adaptiert werden, auch beim Obergang von einer Version des EXTENDED BASIC auf die andere. Das direkte Laden dieser Programme mit CLOAD oder CRUN führt in aller Regel zu einer Zerstörung des EXTENDED BASIC.

MERGE ["name"]

Mit diesem Kommando kann ein zweites BASIC-Programm an ein bereits im Speicher befindliches angehängt werden.

Dieses Programm muss nicht mit der gleichen BASIC-Version erstellt sein.

Die Zeilennummern des zweiten Programmes sollten jedoch größer als die des im Speicher befindlichen sein, ggf. vorher mit RENUM neu numerieren.
RENUM [n,s) (nur in der Großversion)

RENUM [n,s]
Dieser Befehl dient der Neunumerierung eines BASIC-Programms.

Mit "n" kann der Anfangswert, mit "s" die Schrittweite angegeben werden. Fehlt einer der beiden Parameter, so wird als StandardwertJ0" eingesetzt. Beispiel: RENUM 100,5 Neunumerierung beginnend ab 100 in 5er Schritten.

Neunumerierung beginnend ab 10 in 10er Schritten.

AUTO [n,s]

Dieses Kommando erzeugt automatisch Zeilennummern vor jeder Programmzeile, die Sie eingeben wollen. Mit "n" können Sie einen Anfangswert und

mit "s" die Schrittweite angeben. Als Standardwert wird ab 10 in 10er Schritten numeriert.

Die AUTO-Funktion verlassen Sie durch Betätigen der CTRL- und BREAK-Tasten.

Stößt der Computer im AUTO-Modus auf eine bereits besetzte Zeile, so wird der Zeileninhalt ausgegeben und kann überschrieben oder durch ausschließliche Betätigung der RETURN-Taste unverändert übernommen werden.

Achtung:

Enthält eine solche, bereits besetzte Zelle einen der
zusätzlichen Befehle des EXTENDED BASIC, so
können diese nicht dargestellt werden. D. h. sie
gehen verloren, wenn man eine solche Zeile editiert
oder mit RETURN zurücksendet.

Sind Sie nicht sicher, so verlassen Sie den AUTO Modus und prüfen Sie die Zeile mit LIST.

Beispiel: AUTO 10,5 Zeilennumerierung ab 10 in 5er Schritten.

DELETE Zeilennummer [-Zeilennummer]

Dieses Kommando löscht die angegebene Zeile bzw. den Zeilenbereich.

Beispiel:
DELETE 20 löscht Zeile 20
DELETE 50-100 löscht Zeilen 50-100

DELETE -40 löscht vom Programmanfang bis zur Zeile 40

COMPRESS

Hiermit werden aus dem Programmtext alle Leerzeichen (außer in Strings) und "Remarks" entfernt.

Dies kann zu einer erheblichen Verringerung der Programmlänge führen und damit zu einer kürzeren Ladezeit.

Sie sollten bei den von Ihnen entwickelten Programmen eine sauber geschriebene und dokumentierte Version für Fehlersuche und Weiterentwicklung auf Kassette halten. Zur Ausführung bietet sich jedoch die komprimierte Form an, da diese schneller ladbar ist, den Speicher optimaler nutzt und auch schneller abläuft. **RENEW** Haben Sie einmal einen Programmtext durch die versehentliche Eingabe von NEW gelöscht, so können sie diesen mit Hilfe des RENEW-Kommandos wiederherstellen.

Dies geht allerdings nur, solange Sie nicht bereits mit einer Neueingabe begonnen haben (RUN oder LIST haben keinen Einfluß).

TRON

Einschalten der TRACE-Funktion. Sie erlaubt es, den Ablauf eines Programmes zur Fehlersuche zu verfolgen. Sobald der Computer eine neue Programmzeile ausführt, wird ihre Zeilennummer in Klammern ausgegeben.

TROFF
Ausschalten der TRACE-Funktion.

Anmerkung: TRON und TROFF können auch innerhalb eines Programmes eingesetzt werden, wenn nur der Ablauf eines bestimmten Programmabschnitts gewünscht wird.

MEMSIZE Adresse

Mit diesem Kommando können Sie die höchste für BASIC zugängliche Adresse festlegen. Damit haben Sie die Möglichkeit, einen Speicherbereich, in den Sie eine Maschinenprogrammroutine ablegen wollen, vor der Zerstörung durch die automatische BASIC-Speicherverwaltung zu schützen.

Als Adresse ist minimal die unterste für BASIC zugängliche Speicheradresse und maximal die oberste Adresse Ihres Speicherausbaus anzugeben.

Die Adressangabe erfolgt dezimal.

Dieses Kommando kann auch innerhalb eines BASIC-Programms verwendet werden. Dabei ist darauf zu achten, dass es als eines der ersten Kommandos erfolgt, da alle Variablen gelöscht werden (enthält implizit den Befehl CLEAR 50).

Beispiel: MEMSIZE 42000

SYSTEM

Mit diesem Kommando wird Ihr Rechner neu initialisiert.

Nach diesem Befehl sind weder das EXTENDED BASIC noch ein evtl. eingegebenes Programm verfügbar.

Dies ist sinnvoll, wenn Sie anschließend mit dem normalen BASIC weiterarbeiten wollen oder ein Maschinenprogramm laden wollen.

11. Innerhalb eines Programms

DEFINT Buchstabenbereich
Variablen, die mit einem Buchstaben aus dem defi-

nierten Buchstabenbereich beginnen, werden als ganze Zahlen (Integers) behandelt und abgespeichert.

So definierte Variable belegen im Speicher nur 2 Byte. Beachten Sie, dass eine ganzzahlige Variable nur Werte zwischen -32768 und +32767 annehmen kann.

10 DEFINT N,R,V oder 10 DEFINT A-F **DEFSGN Buchstabenbereich**

Variablen, die mit einem Buchstaben aus dem Buchstabenbereich beginnen, werden als Variable einfacher Genauigkeit behandelt und abgespeichert.

Variablen und Konstanten einfacher Genauigkeit werden mit 7 Stellen gespeichert und mit 6 Stellen ausgegeben.

Beachten Sie, dass alle numerischen Variablen, wenn nicht anders definiert, einfache Genauigkeit haben (Standardform). DEFSGN ist sinnvoll, wenn innerhalb eines Programms vorher anders definierte Variable neu definiert werden sollen.

10 DEFSGN A-E,X
DEFDBL Buchstabenbereich

Variablen, die mit einem Buchstaben aus dem Buchstabenbereich beginnen, werden als Variable doppelter Genauigkeit behandelt und gespeichert. Eine solche Variable wird intern mit 17 Stellen behandelt, wovon 16 ausgegeben werden

10 DEFDBL G
DEFSTR Buchstabenbereich

Variablen, die mit einem Buchstaben dieses Buchstabenbereichs beginnen, werden als Zeichenketten-Variable behandelt und abgespeichert.

Anmerkung

Eine explizite Typfestlegung im Programm, z. B. mit % für Integer oder $ für String, überschreibt eine getroffene Definition.

10 DEFDBL A-E
20 A% 10

30 A$ "ABCD" 40 AX 100D4

A% ~ Integer, A$ = String, AX = doppelte Genauigkeit

ERROR

Dieses Kommando wird benutzt, um in eine ERROR Routine zu verzweigen.

Anwendung:
10 ON ERROR GOTO 100

Im Fehlerfall verzweigt das Programm zur Zeile 100, wo eine entsprechende Fehlerauswertung und -behandlung erfolgen kann.

Wird ERROR in der Kommandoposition angegeben (am Zeilenanfang oder nach einem:), so interpretiert Ihr Rechner dies als SOUND-Befehl.

RESUME (Param)

Dieses Kommando beendet eine Fehlerbehandlungsroutine und gibt an, wo die normale Programmausführung fortgesetzt werden soll.

RESUME 0 oder RESUME ohne Zeilennummer bewirkt, dass das Programm in der Zeile fortgesetzt wird, in der der Fehler auftrat.

RESUME mit Zeilennummer (z. B. RESUME 50) bewirkt eine Programmfortsetzung bei dieser Zeile. RESUME NEXT veranlaßt eine Programmfortsetzung an der Zeile, die nach der fehlerhaften Zeile steht. ERL

Gibt die Zeilennummer zurück, in der der Fehler auftrat. ERR Gibt den Fehlercode des aufgetretenen Fehlers zurück.

Beispielprogramm zur Anwendung der Fehlerbehandlung:

```
10 CLEAR 10
20 ON ERROR GOTO 1000
30 INPUT"GEBE MELDUNG EIN"; M$

40 INPUTJETZT EINE ZIFFER BITTE"; N 50 Z = YN

60 PRINT"EINGABEWERTE KORREKT, VERSUCHE MAL WAS FALSCHES."

70 GOTO 30

1000 IF ERL=30 AND ERR=26 THEN 1040 1010 IF ERL=40 AND ERR=10 THEN 1050 1020 IF ERL=50 AND ERR=20 THEN 1060 1030 ON ERROR GOTO 0: RESUME

1040 PRINT"MELDUNG ISTZU LANG--MAX. 10ZEICHEN": RESUME

1050 PRINT"ZIFFER IST ZU GROSS": RESUME

1060 PRINTJEILEN DURCH 0 IN ZEILE 50 - ZIFFER UNGLEICH 0 EINGEBEN"

1070 RESUME 40
```

Fehlernummern und entspr. Meldungen
00 = NEXT WITHOUT FOR
02 = SYNTAX ERROR
04 = RETURN WITHOUT GOSUB
06 = OUT OF DATA
08 = ILLEGAL FUNCTION CALL
10 = OVERFLOW
12 = OUT OF MEMORY
14 = UNDEFINED LINE
16 = SUBSCRIPT OUT OF RANGE
18 = REDIMENSIONED ARRAY
20 = DIVISION BY ZERO
22 = ILLEGAL DIRECT OPERATION
24 = TYPE MISMATCH
26 = OUT OF STRING SPACE
28 = STRING TOO LONG
30 = STRING FORMULA TOO COMPLEX
32 = CAN'T CONTINUE
34 = NO RESUME
36 = RESUME WITHOUT ERROR
38 = UNPRINTABLE ERROR
40 = MISSING OPERAND
42 = BAD FILE DATA
44 = DISK BASIC COMMAND

MEM Gibt die Größe des verfügbaren freien Speicherbereichs aus. Beispiel: 200 IF MEM < 200 THEN 700 Wenn MEM als Kommando genutzt werden soll, muss es zusammen mit PRINT ausgeführt werden. PRINT MEM gibt die Zahl der Bytes im Speicher aus, in denen kein Programm, Variablen oder Zeichenketten stehen.

FRE [Param] FRE (0) ist gleichbedeutend mit MEM (s.o.) FRE ("") liefert den für Zeichenvariablen (Strings) noch verfügbaren Platz. Dieser Wert errechnet sich aus dem mit CLEAR reservierten Bereich abzüglich der darin bereits abgelegten Zeichenvariablen. Beispiele: CLEAR 100: PRINT FRE Ausgabe: 100 CLEAR 100: A$="ABCDE" PRINT FRE ("") Ausgabe: 95

VARPTR (Variable) Die Adresse, an der der Wert der angegebenen Variablen steht, wird übergeben. Beispiel: 10 K = VARPTR (A) Für die verschiedenen Variablentypen hat "K" folgende Bedeutung: a) Ganzzahlige Variable (Integer) (A%) K= LSB K+1 = MSB b) Variable einfacher Genauigkeit (A) K = LSB K+1 = nächst. Wertbyte K+2 = MSB K+3 = Exponent c) Variable doppelter Genauigkeit (DEFDBL A) K = LSB K+1 = nächst. Wertbyte K+ 6 MSB K+7 Exponent d) Zeichenketten-Variablen (A$) K = Länge des Strings K+1 LSB der Anfangsadresse K+2 MSB der Anfangsadresse LSB ist das am wenigsten signifikante Byte MSB ist das am höchsten signifikante Byte

ON n GOTO Zeilel,Zeile2...,Zeilen ON n GOSUB Zeilel,Zeile2 Zeilen Dieser Befehl erlaubt, gleichzeitig mehrere Sprungziele anzugeben. Gesprungen wird in Abhängigkeit vom Inhalt der Variablen "n". Der Wert von "n" muss dabei zwischen 0 und 255 liegen. Wird ein ON - GOTO oder ON - GOSUB-Befehl ausgeführt, so wird zunächst der ganzzahlige Anteil von "n" ermittelt. Nun sucht der Rechner das n-te Element in der Zeilennummernliste und springt zu dieser Zeilennummer. Ist "n" gleich 0 oder größer als die Anzahl der angegebenen Zeilennummern, so wird der auf ON GOTO bzw. ON - GOSUB folgende Befehl ausgeführt. Bei "n" kleiner als 0 tritt ein Fehler auf. Bei einem ON - GOSUB wird nach Rücksprung aus

dem Unterprogramm (RETURN) das Programm an dem auf ON-GOSUB folgenden Befehl fortgesetzt. Beispiel: ohne ON

50 IF C = 1 GOTO 200 60 IF C = 2 GOTO 300 70 IF C = 3 GOTO 400 80 IF C = 4 GOTO 500 90 IF C < 1 OR C > 4 GOTO 1 00:REM NAECHSTER BEFEHL

Mit ON

50 ON C GOTO 200,300,400,500 100...

Zur Anwendung von ON in Verbindung mit ERROR siehe Beschreibung des ERROR-Befehls.

POS (0)

Es wird die momentane Position der Schreibmarke (Cursor) in der Zeile ermittelt und zurückgegeben. 10 A = POS (0)

Das Argument (0) muss angegeben werden, ist aber ohne Bedeutung.

RANDOM

Bei Ausführung dieses Befehls wird ein neuer Ausgangswert für die Ermittlung der Zufallszahlen mit RND (x) gesetzt.

STRING$ in, Zeichen oder Ziffer)

Erzeugt eine Zeichenkette, die aus n mal dem angegebenen Zeichen besteht.

Beispiel:
10 A$ = STRING$ (10,"-")

nach Ausführung A$=" ".....

Anstelle des Zeichens kann auch der entsprechende ASCII-Code direkt als Ziffer angegeben werden. 10 A$ = STRING$ (8,35) nach Ausführung A$=" **CINT (X)** Berechnet die nächste ganze Zahl, die kleiner als das Argument X ist. Das Argument muss zwischen -32768 und +32767 liegen. Beispiel: A = CINT (2,6) ergibt A 2

A = CINT (-2,6) ergibt A 3 **CSNG** (X) Erzeugt die einfache genaue Darstellung von X. Berechnet wird eine 6stellige Zahl mit 4/5 Rundung bei doppelt genauem X.

CDBL (X)

Erzeugt die doppelt genaue Darstellung von X. Das Ergebnis erhält 17 Stellen, wovon die Stellen, die das Argument X enthalten, signifikant sind.

FIX (X)
Trennt die Nachkommastellen von X ab.

A = FIX (1,5) ergibt 1 A = FIX (-1,5) ergibt A
CALL Adresse

Sprung zu der angegebenen Adresse,

Dieser Befehl ist eine komfortable Möglichkeit zum Aufruf von Maschinenprogrammroutinen aus einem BASIC-Programm.

Im Gegensatz zu USR wird die Startadresse direkt dezimal im Befehl angeben.

Das Maschinenprogramm muss mit RET beendet werden, eine Parameterübernahme ist nicht möglich.

Beispiel: CALL 457 löscht den Bildschirm (CLS) III. Zusätzliche Grafik-Befehle

Die unten angeführten Befehle sind nur in der Großversion des EXTFNDED BASIC enthalten.

Sie werden nur ausgeführt, wenn sich der Rechner im Grafik-Modus befindet (MODE (1». Ansonsten wird eine Fehlermeldung ausgegeben.

GCLS [n]

Löscht den Bildschirmspeicher für hochauflösende Grafik mit der als "n" angegebenen Farbe. "n" kann die Werte von 1-4 annehmen.

Ist "n" nicht angegeben, wird der Bildspeicher mit der Hintergrundfarbe gelöscht.

Beispiel:

10 MODE (1): GCLS 3 malt den Bildschirm blau aus. **CPOINT** (x,y)

Erweiterung des POINT-Befehls.

Hiermit erhalten Sie nicht nur die Information, ob ein bestimmter Punkt im Speicher der hochauflösenden Grafikgesetzt ist oder nicht, sondern Ihnen wird auch der jeweilige Farbcode (1-4) mitgeteilt (1=Hintergrund).

Beispiel:

10 MODE (1): COLOR 3,0 20 SET (20,30)

30 A = CPOINT (20,30)

Nach Ausführung dieser Befehlsfolge enthält A den Wert 3.

PLOT [Cn,) Plot-Informationen Setzt einen Punkt oder zeichnet eine Linie auf dem Bildschirm. Die PLOT-Informationen können mehrere Formate annehmen. a) PLOT X,Y setzt den Punkt mit den Koordinaten X und Y (entspr. dem SET-Befehl). b) PLOTX,YTO Xl, Yl zieht eine Linie von X,Y zu Xl, Yl c) PLOT X,Y TO Xl,Yl TO X2,Y2 TO .... Xn,Yn zieht eine Linie von XX zu Xl,Yl von dort zu X2X2 usw. bis Xn,Yn. d) PLOT to Xl,Yl TO zieht eine Linie vom Zielpunkt des letzten PLOT-Befehls zu X1X1 usw. Für "Cn" kann wahlweise eine Farbe angegeben werden. Diese Angabe überschreibt für diesen Befehl die mit COLOR getroffene Farbwahl (n = 1 -4). Wird "Cn" nicht angegeben, so wird die Farbe des letzten COLOR-Befehls verwendet. Beispiel: 50 PLOT C2, 20,30 TO 70,50 zieht eine blaue Linie von 20,30 zu 70,50

NPLOT

Entspricht dem PLOT-Befehl, nur dass die Linie zurückgesetzt wird.

CIRCLE [Cn,] X,Y,R Zieht einen Kreis, dessen Mittelpunkt die Koordinaten X und Y hat und dessen Radius R beträgt. Überschreitet einerder3 Werte 255, wird eine Fehlermeldung ausgegeben. Mit Cn kann wie beim PLOT-Befehl eine Farbe gewählt werden, die die mit COLOR getroffene Wahl überschreibt.

RECT [Cn,] Xl,Yl TO X2,Y2 Zeichnet ein Rechteck auf dem Bildschirm, wobei Xl und Yl die Koordianten der oberen linken Ecke und X2X2 die Koordinaten der unteren rechten Ecke angeben. Zusätzliche Farbwahl mit Cn wie bei PLOT und CIRCLE.

PAINT X,Y,Bl[,B2,B3] Mit diesem Befehl können Sie Flächen auf dem Bildschirm ausmalen. Ausgangspunkt sind die Koordinaten X,Y. Mit Bl, B2

und B3 können bis zu drei Begrenzungsfarben angegeben werden. Bl wird dabei gleichzeitig als Füllfarbe benutzt.

Beispiel:

Es soll ein gelber Kreis auf rotem Hintergrund gezeichnet werden, der blau auszumalen ist. Der Kreismittelpunkt hat die Koordinaten X=64 und Y= 32, der Radius soll 15 betragen.

10 MODE (1): COLOR 3,0 :'Umschalten auf Grafik 20 GCLS 4 :'Bildschirm mit rot löschen

30 CIRCLE C2, 64, 32, 15 :'Kreis zeichnen

40 PAINT 64, 32, 3, 2 :'Kreis blau ausmalen 50 GOTO 50 :'Endlosschleife

IV. Benutzung eines Lichtstiftes (LIGHTPEN)

Dieser Befehl ist auch nur in der Großversion vorhanden

(BRUN "LPEN" oder RUN "DEMO")

LPEN (X,Y [,A])

Dieser Befehl ermittelt Koordinaten und wahlweise auch die Bildadresse des Bildschirms, auf der ein Lichtstift aufgesetzt ist.

Für X und Y sind numerische Variable anzugeben, in die die XY-Koordinaten vom LPEN-Befehl übertragen werden.

Parameter "A" ist wahlweise. Ist hier ebenfalls eine numerische Variable definiert, so wird in dieser zusätzlich die Bildschirmadresse übergeben.

Ist der Lichtstift noch nicht richtig aufgesetzt, oder stimmen die Helligkeitsverhältnisse des Bildschirms nicht für eine Auswertung, so werden die X,Y-Variablen auf 255 gesetzt und A auf -`I.

Die Werte für X,Y und A werden entsprechend dem jeweiligen Modus ermittelt (Grafik oder Text) d. h.

MODE (0) X= 0-31 Y= 0-15A= 0-511

MODE (1) X = 0-127 Y = 0-63 A = 0-2047 Beispiel:

Im Textmodus soll an jeder Berührungsstelle des Lichtstiftes das Zeichen "- auf dem Bildschirm erscheinen.

```
10 CLS
20 LPEN (X,Y,A)
30 IF X = 255 THEN 20

40 SOUND 33, 1: PRINT @ A,""' 50 GOTO 20
```