Eine tolle Sache

EXBASIC

Schon längst gibt es Programme, die das Arbeiten mit dem Commodore 64 sämtliche Möglichkeiten, manche beschränken sich auf spezielle jedoch Schwerpunkte und besitzt auch Schwachstellen. Exbasic

Daß die auf Vielseitigkeit ausgelegte Hardwarekonzeption des Commodore 64 erst mit einer leistungsfähigeren Sprache beziehungsweise Spracherweiterung voll zum Tragen kommt, steht außer Zweifel. Das bei Commodore seit langem bestehende Mißverhältnis zwischen der hard- und softwaremäßigen Grundausstattung der Geräte war daher auch seit längerer Zeit für einige Software-Anbieter Anlaß, sinnvolle Erweiterungen des Betriebssystems und des Basic-Sprachumfangs zu entwickeln.

Exbasic Level II wurde ursprünglich für die älteren und größeren Commodore-Computer als fest zu installierender Hardware-Zusatz konzipiert. Die vorliegende C 64-Version kann diese Herkunft nicht leugnen, da sie nur bedingt an die neuen Möglichkeiten dieser Maschine angepaßt erscheint. Während zum Beispiel Simons Basic eindeutig neu für den C 64 entwickelt wurde und die grafischen und akustischen Fähigkeiten des C 64 voll unterstützt, hat man mit Exbasic Level II eher ein Produkt für kommerzielle Programme, also zum Beispiel solche aus dem Business-Bereich in der Hand.

Exbasic wird als Erweiterungsmodul (siehe Bild) zusammen mit zwei Handbüchern geliefert. Der Vorteil dieser, gegenüber einer reinen Diskettenversion, teureren Lösung spürt man deutlich, wenn man regelmäßig mit Exbasic arbeitet, da durch die langsame Übertragungsgeschwindigkeit vom Laufwerk zum C 64 für 8 KByte schon eine beträchtliche Ladezeit benötigt wird. Man steckt einfach das Modul in den Expansion-Port und erhält bereits nach dem Einschalten eine entsprechende Startmeldung. Es belegt den Speicherbereich $8000 bis $9FFF, man hat also 8 KByte weniger Speicherplatz zur Verfügung als sonst.

Unbequemer ist da schon die Lektüre der beiden Handbücher: Das stärkere gelbe Buch beschreibt die Exbasic-Versionen für die älteren Commodore-Computer in einem zügig lesbaren, lockeren Stil, mit vielen kleinen Beispielen. Um aber mit der C 64-Version zurecht zu kommen, muß man gleichzeitig das blaue Ergänzungsbuch lesen. Darin findet man neben vielen Verweisen auf das Handbuch Hinweise zu Befehlen, die es in der C 64-Version nicht gibt, zu erweiterten Parametrisierungen von existierenden Befehlen und Beschreibungen von solchen Befehlen, die speziell für den C 64 dazugekommen sind. Man sollte sich also, um Frustration zu vermeiden, beim Studium von dem blauen Ergänzungsbuch leiten lassen. Die Befehlsübersicht in diesem Artikel soll den Überblick erleichtern.

Die einzelnen Gruppen, in die man die Befehle gliedern kann, sollen im folgenden besprochen werden. Dabei verdienen manche Befehle noch eine besondere Erwähnung.

Hilfsfunktionen

Diese dienen dazu, das Erstellen und Testen von Basic-Programmen komfortabler zu gestalten. Es wurden dabei einige sinnvolle Eigenschaften realisiert, die sich nicht direkt durch Schlüsselwörter ausdrücken lassen. So kann man ein Programm komfortabel mit Hilfe der Cursortasten listen: Drückt man am unteren Bildrand die Cursor-down-Taste, so werden weitere Zeilen gelistet, während das Bild nach oben rollt. Für den oberen Bildrand und die Cursor-up-Taste gilt entsprechendes. Wird ein Programm fehlerhaft abgebrochen, so wird nicht nur eine Fehlermeldung abgesetzt, sondern auch die fehlerhafte Zeile mit dem Cursor an der fehlerhaften Stelle angezeigt. Diskettenbefehle werden abgekürzt (entsprechend den Möglichkeiten des DOS 5.1 auf der Demodiskette):

LEVEL II

der interessantesten Spracherweiterungen für den Commodore 64.

erleichtern. Manche unterstützen Bereiche. Jede Erweiterung setzt Level II von Interface Age ist dabei eine

/programmname	lädt ein Programm
↑programmname	lädt ein Programm und startet es
←programmname	speichert ein Programm auf Disk
@	fragt den Fehlerkanal des Laufwerks ab
@$	zeigt das Inhaltsverzeichnis an, ohne das Programm im Speicher zu zerstören
@Befehl	sendet einen Befehl an das DOS

LOAD und SAVE bezeichnen Kassettenbefehle. Diese laufen in Exbasic mit fünffacher Geschwindigkeit ab.

FIND

Damit kann man alle Zeilen vollständig listen lassen, die einen angegebenen Suchstring an irgendeiner Stelle enthalten. So werden zum Beispiel mit FIND REM alle Kommentarzeilen gelistet. Mit FIND A$ kann man sich alle Stellen, an denen auf die Variable A$ zugegriffen wird, listen lassen. Zum Editieren von Programmen ist diese Möglichkeit von nicht zu unterschätzendem Wert.

RENUM

Exbasic unterstützt weniger als Simons Basic das strukturierte Programmieren. Sprünge und Aufrufe von Unterprogrammen sind immer mit der Angabe von Zeilennummern verbunden. Dem trägt aber der Befehl RENUM Rechnung. RENUM nummeriert nicht nur (wie bei Simons Basic) die Zeilennummern neu, sondern ändert auch die Argumente aller Befehle, die sich auf Zeilennummern beziehen, wie GOTO, GOSUB, THEN und so weiter, sinngemäß ab, so daß, wie man es eigentlich auch erwartet, das umnummerierte Programm lauffähig bleibt.

TRACE

Dieser Befehl schaltet einen Modus ein, in dem jede aktuelle Befehlszeile eines laufenden Programms in der ersten Bildschirmzeile kurz angezeigt wird. Bei Zeilen mit mehreren Befehlen steht sogar der Cursor auf dem jeweils aktuellen Befehl. So lassen sich zum Beispiel auch bedingte Anweisungen gut mitverfolgen. Verlangsamung des Vorgangs und auch Einzelschrittbetrieb sind mit der CTRL-Taste möglich. Durch Drücken der Commodore-Taste läuft das Programm wieder schneller. Da die Befehle TRACE und TRACE OFF auch programmierbar sind, kann man sich in einem Programm auch gezielt Teile, die man testen will, »tracen« lassen.

SPACE

Um Speicherplatz zu sparen, eliminiert Exbasic alle Blanks in Programmen außer in Kommentaren und in Texten, die durch Apostrophe begrenzt werden. Durch den Befehl SPACE (ohne Parameter) werden beim Listen (nicht im Programm selbst) wieder Blanks eingefügt, um die Übersichtlichkeit eines Programms zu wahren.

BASIC

Dieser Befehl desaktiviert Exbasic, was außer einem geringen Geschwindigkeitsgewinn bei Standard-Basic-Programmen eigentlich nichts bringt. Er wird allerdings notwendig, wenn man mit den Diskettenbefehlen LOAD"name",8 und LOAD"name",8,1 oder SAVE"name",8 arbeiten will, oder wenn das Programm diese Befehle enthält. Diese Befehle wirken in Exbasic unerfreulicherweise auf Kassette und nicht auf die Disk. Exbasic kann mit PRINT USR(0) wieder gestartet werden.

MERGE

Hiermit kann man ein Programm von Diskette oder Kassette zu einem Programm im Hauptspeicher dazuladen. Das nachzuladende Programm sollte Zeilennummern haben, die von denen des vorhandenen Programms verschieden sind, ansonsten bedarf das Mischprodukt noch einer manuellen Nachbehandlung, da doppelte Zeilennummern auftreten. Die richtige Form des Befehls für das Nachladen von Diskette ist übrigens:
MERGE* "programmname", 8

KEY

Durch die Möglichkeit, die f-Tasten mit beliebigen Texten zu belegen (einschließlich RETURN durch die Taste »←«) werden die guten Editierfähigkeiten von Exbasic abgerundet.

Kontrollstrukturen

Da Exbasic mehr das zeilennummern-orientierte Programmieren unterstützt findet man hier nur spärliche Möglichkeiten. Die von PASCAL her bekannten Konstruktionen wie WHILE ... DO oder REPEAT ... UNTIL wird man hier vergeblich suchen. Das eigentlich schon zum Standard gehörende IF ... THEN ... ELSE gibt es aber auch in Exbasic.

DISPOSE

Der Nutzen dieses Befehls erscheint eher zweifelhaft. Er läßt zwar manchmal trickreiche und kurze Programme zu, untergräbt aber das Konzept von strukturierten Programmen. Der Interpreter verwaltet für Schleifen und Unterprogrammsprünge einen Stack, auf dem Rücksprungadressen und Namen von Laufvariablen gehalten werden. Diesen Stack kann man mit DISPOSE manipulieren. So kann man mit DISPOSE NEXT eine Schleife abschließen, um sie dann mit GOTO irgendwo zu verlassen. DISPOSE RETURN schließt eine Unterprogrammebene ohne Rücksprung ab. Man muß dann das Unterprogramm mit einem GOTO verlassen. DISPOSE CLR löscht den gesamten Stack. Die Beispiele im Handbuch zu DISPOSE sind nicht sehr gelungen. Eine vernünftige Anwendung ergibt sich meiner Ansicht nach nur im Zusammenhang mit einer programmierten Fehlerbehandlung.

EXBASIC LEVEL II

Fehlerbehandlung

Die Befehle zur Fehlerbehandlung sind sehr nützlich und sinnvoll.

ON ERROR GOTO

Wenn dieser Befehl einmal abgearbeitet wurde, bricht bei einem späteren Fehler das Programm nicht ab, sondern springt an die bei ON ERROR GOTO angegebene Zeilennummer. Dort kann der Benutzer seine eigene Fehlerbehandlungsroutine unterbringen. Dazu stehen ihm zur Information die Variablen EC (Error Code) und EL (Error Line) zur Verfügung. EC enthält die Nummer des Fehlers gemäß einer Liste im Handbuch, EL die Zeilennummer der Zeile, in der der Fehler aufgetreten ist. Die Fehlerbehandlungsroutine wird mit einer der drei RESUME-Varianten verlassen (siehe Tabelle).

Ein- und Ausgabebefehle

Diese machen neben den Hilfsfunktionen die eigentliche Stärke von Exbasic aus. Mit wenigen, aber leistungsfähigen Befehlen ist eine vielfältig formatierbare Ausgabe möglich. Man kann auch sehr einfach gegen Fehlbedienung gesicherte Eingabemasken realisieren.

INPUTFORM

Dieser Befehl gestattet es, neben der Ausgabe eines Textes, ein Eingabefeld in Position, Länge und Farbe zu definieren. Gleichzeitig wird mit diesem Befehl eine Eingabe über dieses Feld angefordert. Dabei ist es nicht möglich, über das festgelegte Feld hinauszuschreiben. Eingegeben werden können alle Zeichen außer den Cursor-Steuerzeichen. Mit DEL kann man einzelne Zeichen löschen, mit »←« den ganzen Text.

PRINT USING

Formatierte Ausgabe von Dezimalzahlen wird hiermit ermöglicht. Es wird ein Formatstring ausgegeben, der beliebig Text sowie Formatzahlen enthalten darf. Bei der Ausgabe werden dann die Formatzahlen durch aktuelle Zahlen aus der Variablenliste ersetzt. Eine Formatzahl setzt sich aus Formatzeichen "#", "*", ",", ".", "+" und "–" zusammen. Dabei stehen "#" und "*" für Dezimalstellen. Beispielsweise führt PRINT USING "DM ###,##", 2000/3 zu DM 666,67

Diese Befehle sind mit Exbasic Level II möglich

Konventionen: X,X1,Y und so weiter bezeichnen allgemeine numerische Ausdrücke, ansonsten werden Parameternamen klein geschrieben. »ad« steht für Adresse (0...65535), »fz« für Farbzahl (1...16). Runde Klammern müssen mit eingegeben werden, eckige und spitze Klammern werden nicht eingegeben. Parameter in eckigen Klammern müssen angeführt werden, während Parameter in spitzen Klammern optionell sind.

Hilfsfunktionen:	
FIND [string]⟨,bereich⟩	Zeilen, die den String enthalten, listen
AUTO ⟨zeilennr.⟩⟨,schrittweite⟩	automatische Zeilennummernvorgabe
DEL [bereich]	Bereich löschen
RENUM ⟨startnr.⟩⟨,schrittweite⟩	Zeilen umnummerieren
TRACE/TRACE OFF	aktuellen Befehl anzeigen
DUMP	nichtindizierte Variablen mit Wert anzeigen
MATRIX	indizierte Variablen mit Wert anzeigen
LETTER	Umschalten in Klein/Großschrift
LETTER OFF	Umschalten in Großschrift/Blockgrafik
MEM	Speicherbelegung anzeigen
HIMEM [ad]	obere Speichergrenze für Basic setzen
SPACE/SPACE OFF	beim Listen Blanks mit ausgeben
HELP	Exbasic-Schlüsselwörter auflisten
HELP*	Standard-Basic-Schlüsselwörter auflisten
BASIC	Rückkehr ins Standard-Basic
PRINT USR (0)	Exbasic Level II reaktivieren
MERGE ⟨,programmname⟩	weiteres Programm in bestehendes einkopieren
KEY	F-Tastenbelegung anzeigen
KEY [nummer],[string]	F-Tastenbelegung definieren
KEY ON	Standardbelegung herstellen
Kontrollstrukturen:	
IF...THEN...ELSE	bedingte Verzweigung
RESTORE [zeilennummer]	DATA-Zeilen-Zeiger auf angegebene Zeile setzen
ON X RESTORE [zeilennummernliste]	RESTORE mit berechneter Zeilennummer
DISPOSE	Manipulationen des Stack
DISPOSE CLR	Abschluß aller Schleifen und Unterprogramme
DISPOSE RETURN	Abschluß einer offenen Unterprogrammebene ohne Rücksprung
DISPOSE NEXT [variablenname]	Abschluß der Schleife mit der angegebenen Laufvariablen sowie aller Schleifen innerhalb dieser, aber kein Sprung aus der Schleife
ON ERROR GOTO [zeilennummer]	Fehlerausgang definieren
RESUME	Rückkehr nach Fehlerbehandlung zur Zeile, in der der Fehler aufgetreten ist
RESUME NEXT	Rückkehr nach Fehlerbehandlung zu der Zeile, die auf die Zeile folgt, in der der Fehler aufgetreten ist
RESUME [zeilennummer]	beliebiger Rücksprung nach Fehlerbehandlung
Ein-/Ausgabebefehle:	
INPUTLINE ⟨"text";⟩ [stringvariable]	Eingabe sämtlicher Zeichen
INPUTFORM ⟨"text";⟩[stringvariable] ⟨,maximallänge⟩⟨,fz⟩	Eingabe beliebiger Zeichen außer Cursorsteuerzeichen in das Feld mit angegebener Länge
PRINT@ [position],"text"	Ausgabe an beliebiger Bildschirmposition
PRINT USING ["format"],[varliste]	formatiertes Drucken
SPACE [spalte1,zeile1,spalte2, zeile2]⟨,code⟩⟨,fz⟩	Die ersten 4 Parameter bestimmen den linken oberen bzw. den rechten unteren Eckpunkt eines Rechtecks, das mit dem durch seinen Bildschirmcode gegebenen Zeichen gefüllt wird.
STRING$ (anzahl, string od. X)	Vervielfachung des angegebenen Strings oder des durch den ASCII-Code X gegebenen Zeichens
HARDCOPY	Ausgabe des Bildschirminhalts (nur Blockmodus)
Stringbefehle:	
INSTR (String 1, string 2 ⟨position⟩)	sucht string2 in string1 ab position und liefert, falls vorhanden, die Position von string2 in string1, sonst 0
EVAL (string)	Auswertung eines durch string dargestellten numerischen Ausdrucks
EXEC (string)	Ausführung eines im string enthaltenen Basic-Befehls

EXBASIC LEVEL II

Grafik- und Farbbefehle:	
CURSOR [fz]	Ausgabefarbe setzen
GROUND [fz]	Hintergrundfarbe setzen
BORDER [fz]	Rahmenfarbe setzen
COKE [position,zeichencode,fz]	Durch Code bestimmtes Zeichen an Bildschirmposition setzen
CEEK [position,modus]	modus = S: Bestimmung des Zeichencodes des Zeichens an gegebener Position modus = C: Bestimmung des Farbcodes
HPLOT X⟨,fz⟩	horizontalen Balken der Länge X plotten
VPLOT X⟨,fz⟩	vertikalen Balken der Höhe X plotten
SET (spalte,zeile)	Punkt setzen (Blockgrafik 80x50)
RESET (spalte,zeile)	Punkt löschen
POINT (spalte,zeile)	Test, ob Punkt gesetzt
Tonerzeugung:	
VOLUME X	Lautstärke des SID setzen (0-15)
ADSR [stimme,kurvenform,a,d,s,r],⟨pulsweite⟩	Klangparameter setzen
PLAY [stimme,tonhöhe]	Ton anschlagen
Numerische Funktionen:	
MAX (X1,X2,...,Xn)	*Maximum*
MIN (X1,X2,...,Xn)	*Minimum*
FRAC (X)	*Nachkommastellen einer Dezimalzahl*
ROUND (X,Y)	*Rundung von X auf Y Nachkommastellen*
ODD (X)	*Test des ganzzahligen Teils von X auf ungerade*
RND (X)	*Für X >=2* ganze Zufallszahlen zwischen 1 und X, sonst wie Standard-RND
HEX$ (X)	Umwandlung von X in Hexadezimalstring
DEC (Hexadezimalstring)	Umwandlung in Dezimalform
Verbindungen zur Maschinensprache:	
CALL (parameterliste)	Aufruf eines Maschinenprogramms mit Parametern über speziellen Einsprungvektor
DEF CALL = [ad]	Definition des CALL-Vektors
DOKE [ad],X	Doppelbyte-POKE
DEEK (ad)	Doppelbyte-PEEK
VARPTR (variablenname)	Anfangsadresse, ab der der zugehörige Variablenwert im Speicher liegt
Sonstige Befehle:	
SEC X	X Sekunden Verzögerung
PAUSE X	X/16 Sekunden Verzögerung
LOCK	Manuelle Umschaltung Schreibmaschinenmodus/Grafikmodus sperren
LOCK OFF	LOCK aufheben
SWAP [varname1],[varname2]	Werte zweier Variablen vertauschen

Auch wenn man keine EXBASIC-Level II-Befehle in seine Programme einfügen will, also nur das Standard-Commodore-Basic benutzt, ist EXBASIC-Level II eine sehr komfortable Editierhilfe

SPACE mit Parameterangabe

Im Gegensatz zu SPACE ohne Parameter (siehe oben) kann man mit SPACE plus Parameterangabe sehr schnell ganze Rechteckbereiche mit einem Zeichen beschreiben. Man kann damit zum Beispiel große farbige Rechtecke erzeugen oder Teilbereiche des Bildschirms gezielt löschen.

Stringbefehle

EVAL und EXEC erlauben es, Strings als numerische Ausdrücke beziehungsweise als Basic-Befehle aufzufassen. Damit ist es möglich, in ein laufendes Programm eine Funktion einzugeben, um zum Beispiel deren Nullstellen zu ermitteln, ohne das Programm abzubrechen. Denkbar wäre auch die Simulation eines leistungsfähigen Taschenrechners.

Grafik- und Farbbefehle

Es werden leider nur die Möglichkeiten unterstützt, die sich aus den Grafikzeichen der Tastatur ergeben. Dazu gehören Balken, deren Länge beziehungsweise Höhe sehr fein abgestuft werden kann, deren Breite aber immer acht Rasterpunkte beträgt. Da diese Balkengrafik allerdings sehr schnell ist, könnte man sie beispielsweise zur quasi-analogen Anzeige von Meßwerten in Realzeit benutzen.

Weitere Befehle

Die Befehle zur Tonerzeugung ersparen eine lange Reihe von PEEKs und POKEs, auch muß man die Adressen der SID-Register nicht kennen. Man kommt jedoch nicht umhin, sich mit den Einzelheiten des Sound-Chips zu befassen. Eine Reihe weiterer Befehle erleichtert das Zusammenspiel zwischen Basic- und Maschinenspracheprogrammen erheblich. Viele Programmieraufgaben lassen sich nur in Maschinensprache vernünftig lösen. Diese Maschinensprache versieht man aber meistens mit Rahmenprogrammen in Basic, die dazu dienen, komfortabel mit dem Benutzer zu kommunizieren (in Basic liegen komfortable E/A-Routinen eben schon vor) und dazu, das Maschinenprogramm mit Parametern zu versorgen und es aufzurufen. Letzterem dienen die Befehle HEX$, DEC, CALL, DOKE, DEEK und VARPTR. Man halte sich vor Augen, daß zum Beispiel zur Änderung eines 16-Bit-Wertes der Befehl:
DOKE AD,X genügt, während man in Standard-Basic die etwas umständliche Sequenz:
POKE AD,X—256*INT(X/256)
POKE AD+1,INT (X/256)
benötigt.

Zusammenfassende Beurteilung

Wer die vollen Grafik- und Klangmöglichkeiten des C 64 (einschließlich Sprites) mit von Basic ausnutzen will, der sollte Simons Basic oder eine andere spezielle graphische Befehlserweiterung wählen. Auch was strukturierte Programme anbelangt, fährt man mit Simons Basic besser.

Wer allerdings Basic-Programme (unter anderen auch schon vorhandene) sehr komfortabel editieren und testen will, wer maskierte Eingabe und formatierte Ausgabe benötigt, für Business-Programme, für komfortable und schnelle Rahmenprogramme zu Maschinenspracheprogrammen, für den ist Exbasic Level II sicher das geeignetere Werkzeug.

(Thomas Krätzig)