»Maschinen-Power« mit Basic

Träumen Sie als Basic-Programmierer nicht auch manchmal von so atemberaubenden Geschwindigkeiten, wie sie die Maschinensprache ermöglicht? Wenn ja, dann kommen Sie im folgenden Artikel voll auf Ihre Kosten. Wir werden dem Basic-Interpreter durch die Anwendung von Betriebssystem-Routinen nämlich ganz schön einheizen.

Der C64 ist in der Grundausstattung nicht gerade mit einem herausragenden Basic gesegnet. Will man mehr aus seinem Commodore herausholen, so kann man sich entweder eine der inzwischen zahlreichen Basic-Erweiterungen zulegen, oder (abgesehen davon, überhaupt eine neue Sprache zu erlernen) sich in die Tiefen der Maschinensprache wagen, um sich seine eigenen Befehle und Unterprogramme zu erstellen.

Es gibt jedoch noch eine dritte Möglichkeit, sozusagen den goldenen Mittelweg, nämlich das Ausnutzen der vorhandenen Interpreter- und Betriebssystem-Routinen von Basic aus. Wenn solche Routinen von Maschinensprache-Programmierern erst einmal analysiert worden sind, dann wird auch der Basic-Programmierer in die Lage versetzt, diese Unterprogramme ohne Maschinensprache-Kenntnisse zu verwenden.

Kleine »Byte«ologie

Folgende kleine Tatsachen sollten Ihnen jedoch schon geläufig sein. Wie Sie sicher wissen, speichert Ihr Computer jede Zahl, jeden Buchstaben, jedes Programm, schlicht und einfach alles in Bytes ab. Ein Byte kann bis zu 256 verschiedene Werte annehmen, man kann also Zahlen von 0 bis 255 darin speichern. Um größere Werte verarbeiten zu können, nimmt man einfach zwei Byte zusammen und erhält so die maximale Zahl 65535. Das soll für unsere Zwecke genügen. Das erste Byte enthält den niederwertigen Anteil (nicht, wie man vielleicht annehmen möchte, den höherwertigen Teil), im folgenden deshalb Low-Byte genannt. Entsprechend heißt das zweite, höherwertige Byte, High-Byte.

Folgende Basic-Zeile wandelt eine Zahl in zwei Byte um:

```
H%=INT(X%/256):L%=X%-H%*256
```

```
1 REM BLOCKVERSCHIEBUNG                    <198>
2 REM CREATED BY CHRISTOPH BERGMANN        <242>
3 REM                                      <065>
10 DEF FN H(X)=INT(X/256)                  <244>
20 DEF FN L(X)=X-FN H(X)*256               <019>
100 INPUT"ALTE STARTADRESSE";S             <215>
110 INPUT"ALTE ENDADRESSE+1";E             <089>
120 INPUT"NEUE ENDADRESSE+1";N             <212>
200 POKE 95,FN L(S):POKE 96,FN H(S)        <221>
210 POKE 90,FN L(E):POKE 91,FN H(E)        <202>
220 POKE 88,FN L(N):POKE 89,FN H(N)        <136>
300 SYS 41920:END                          <098>
```

Listing 1. Routine zur Verschiebung von beliebigen Speicherbereichen

X% ist hierbei die zu wandelnde Zahl; L% und H% natürlich das Low- und High-Byte.

Und umgekehrt das Errechnen einer Zahl aus zwei Byte:

```
X%=H%*256%+L%
```

Eine Eigenschaft der Maschinensprache ist es, bei Angaben von Speicherbereichen die Endadresse plus 1 anzugeben. Beispiel: Sie wollen den Bildschirm von 1024 bis 2047 speichern. Dann müssen Sie als Startdresse den Wert 1024 und als Endadresse den Wert 2048 (!) verwenden.

So, soviel zum nötigen »Grundwissen«. Jetzt geht's los mit dem ersten Tip! Eine häufige Aufgabe, die der Computer zu erfüllen hat, ist das Verschieben von Speicherbereichen. Wollen Sie zum Beispiel einen deutschen Zeichensatz auf dem C64 realisieren, so müssen Sie zuerst den normalen Zeichensatz auf einen freien Platz kopieren und dann dort die gewünschten Zeichen in Umlaute ändern. Eine andere Anwendung ist das Kopieren des Basic-ROMs und des Betriebssystem-ROMs (Kernel) in das darunterliegende RAM, um dort dann Änderungen vorzunehmen (zum Beispiel Eindeutschen des Befehlssatzes). Dabei müssen allerdings über 16000 Byte gelesen und wieder geschrieben werden.

```
1 REM DEMO ZUR CURSORPOSITIONIERUNG        <250>
2 REM CREATED BY CHRISTOPH BERGMANN        <242>
3 REM                                      <065>
10 POKE 53280,0:POKE 53281,0               <138>
20 PRINT"{CLR}";                           <214>
30 FOR X=0 TO 6 STEP.1                     <036>
40 Y=SIN(X)+1                              <216>
50 POKE 211,X*6:POKE 214,Y*11              <184>
60 SYS 58732:POKE 783,1:SYS 58634          <038>
70 PRINT"{WHITE}6{GREY 3}4{GREY 2}E{GREY 1
   }R"                                     <007>
80 NEXT:END                                <051>
```

Listing 2. Dieses Programm demonstriert die Cursor-Positionierung auf dem Bildschirm

In Basic dauert das eine ganze Weile. Aber der Interpreter stellt uns eine Routine zur Verfügung, die genau dasselbe in Sekundenschnelle für uns erledigt. Hierzu muß man in die Speicherstelle 95 das Low-Byte und in 96 das High-Byte der Anfangsadresse des zu verschiebenden Bereichs angeben, sowie entsprechend in 90 und 91 die Endadresse. In 88 und 89 schließlich muß die Endadresse (!) des Zielbereichs angegeben werden. Rufen Sie danach die Roumit SYS 41920 auf (Das Programm in Listing 1 verschiebt einen beliebigen Speicherblock). In den Zeilen 10 und 20 sehen Sie übrigens eine sehr schöne und komfortable Methode zur Umwandlung einer Adresse in das Low-Byte und High-Byte.

Speicher »herumschieben«

Vom Verschieben von Speicherbereichen zum Verschieben des Bildschirms. Sie können den Bildschirm jederzeit, ohne die Cursorposition zu verändern, nach oben scrollen, indem Sie einfach die entsprechende Routine mit SYS 59626 aufrufen.

An dieser und an den folgenden Routine sehen Sie übrigens, wie leistungsfähig das Betriebssystem des Commodore 64 zum Beispiel in der Bildschirmsteuerung ist. Man muß es nur zu nutzen wissen. Wenn Sie eine oder mehrere Zeilen auf dem Bildschirm löschen wollen, so können Sie dies ebenfalls dem Betriebssystem überlassen. Einfach die zu löschende Zeile in die Speicherzelle 781 POKEn und die entsprechende Routine mit SYS 59903 starten. Schon ist die gewünschte Zeile vom Bildschirm verschwunden. Wollen Sie mehrere Zeilen löschen, so können sie das in einer FOR-NEXT-Schleife Das sieht dann folgendermaßen aus:

```
10 FOR T=A TO E : POKE 781,T : SYS 59903 : NEXT T
```

A ist hierbei die Anfangszeile und E entsprechend die Endzeile. Eine weitere, sehr wichtige Unterroutine ist die »Cursor

Setzen/Holen«. Damit kann man den Cursor auf jede beliebige Position des Bildschirms setzen. Einfach die Zeile in 214 und die Spalte in 211 schreiben und... Halt! Machen Sie nicht den Fehler und rufen Sie, wie schon oft in Zeitschriften geschrieben, sofort die Routine auf. Erstens sollten Sie mit POKE 783,1 die Routine auch auf »Setzen« schalten, und zweitens müssen sie vorher nämlich erst die »Cursorposition berechnen«-Routine aktivieren. Dies geschieht mit SYS 58732. Dann können Sie gefahrlos mit SYS 58634 das erste Unterprogramm starten (Das Programm in Listing 2 demonstriert dies mit einer kleinen Sinuskurve).

Für die zweite Funktion der Unterroutine, nämlich »Cursorposition holen«, müssen Sie lediglich POKE 783,0 und SYS 58634 eingeben, danach steht die Zeile des Cursors in der Speicherzelle 781 und die Spalte in 782. Damit läßt sich zum Beispiel ein Menü, aus dem man mit einem Cursor auswählen kann, sehr gut realisieren.

Sicher ist es Ihnen schon einmal passiert, daß Sie alle möglichen Parameter zur Bildschirmausgabe verändert haben (zum Beispiel Rahmen-, Hintergrundfarbe, Lage des Bildschirms, und so weiter...) und Sie nun nicht mehr wissen, wie die Anfangswerte ausgesehen haben. Abgesehen davon ist es sehr mühsam, das alles »per Hand« wieder zurückzustellen. Durch Aufruf der Routine »Bildschirm-Reset« mit SYS 58648 wird alles wieder in den Ausgangszustand (wie nach dem Einschalten, also dunkelblauer Hintergrund, hellblaue Rahmen- und Zeichenfarbe, und so weiter...) zurückgesetzt. Dies sollten Sie übrigens zu Beginn jedes Programms einmal durchführen, da ja der Benutzer vorher alles verstellt haben könnte.

Wenden wir uns nun ab von der Bildschirmsteuerung, hin zu anderen Betriebssystem-Routinen. Zur Fehlersuche zu gebrauchen ist eine Routine, die die aktuelle Zeilennummer, in der sich das Programm gerade befindet, ausgibt. Durch SYS 48578 erfolgt die Ausgabe »in xxxx«, wobei es sich bei »xxxx« um die aktuelle Zeilennummer handelt.

Das laufende Basic-Programm wird danach ganz normal fortgesetzt.

Wenn Sie nicht Fehler verhindern oder aufspüren, sondern produzieren wollen, so hilft Ihnen die folgende Routine weiter. POKEn Sie einen Wert zwischen 1 und 30 in die Speicherzelle 781 und starten Sie das Unterprogramm mit SYS 42039. Schon bricht Ihr Programm mit der der Nummer entsprechenden Fehlerausgabe ab.

```
1 REM SPEICHERBEREICH ABSPEICHERN                <181>
2 REM CREATED BY CHRISTOPH BERGMANN              <242>
3 REM                                            <065>
10 DEF FN H(X)=INT(X/256)                        <244>
20 DEF FN L(X)=X-FN H(X)*256                     <019>
100 INPUT"STARTADRESSE";S                        <085>
110 INPUT"ENDADRESSE+1";E                        <230>
120 INPUT"NAME{8SPACE}";N$                       <146>
150 SYS(57812)N$,8                               <215>
200 POKE 193,FN L(S):POKE 194,FN H(S)            <206>
210 POKE 174,FN L(E):POKE 175,FN H(E)            <222>
300 SYS 62957:END                                <198>
```

Listing 3. Speichern eines beliebigen Speicherbereichs auf Kassette oder Diskette

Alle Kanäle schließen

Nun wieder zu etwas Nützlichem: Wenn Sie in einem Programm mehrere Ein- oder Ausgabekanäle mit OPEN eröffnet haben, so ist es etwas mühsam, alle Kanäle wieder mit CLOSE zu schließen. Das Betriebssystem hat hierfür ein Unterprogramm, das alle möglichen Kanäle auf einmal schließt. Die Startadresse der Routine lautet 62255.

Wie Sie sicher wissen, kann ein Programm nicht nur mit LOAD"Name",8 (Hier wird das Programm immer ab dem Basic-Anfang, der bei 2049 liegt, geschrieben), sondern auch mit LOAD"Name",8,1 geladen werden. Bei letzterem lädt das Betriebssystem ein Programm an die angegebene Stelle im Speicher. Die Startadresse wird beim SAVEn mitgespeichert. Dies kann man sehr nutzbringend anwenden, denn es muß sich ja nicht unbedingt um ein (in diesem Fall Maschinen-)Programm handeln. Man kann vielmehr jeden beliebigen Speicherbereich speichern, zum Beispiel Sprite-Daten, den Bildschirmspeicher, HiRes-Grafiken, Variablenwerte, etc. Es ergeben sich wirklich viele Anwendungsmöglichkeiten.

Um so einen Bereich zu speichern, müssen Sie folgendes eingeben: Als erstes »SYS(57812)"Name",g« zum Eröffnen des Programmfiles. Wenn Sie für »g« den Wert 8 verwenden,

```
1 REM EINGABE - UNTERPROGRAMM                    <250>
2 REM CREATED BY CHRISTOPH BERGMANN              <242>
3 REM                                            <065>
60000 SYS 42336:E$="":Z=512                      <059>
60010 P=PEEK(Z):IF P THEN E$=E$+CHR$(P):Z=
      Z+1:GOTO 60010                             <135>
60020 RETURN                                     <131>
```

Listing 4. Unterprogramm zur Eingabe einer Zeichenkette in eine Stringvariable

so speichern Sie das Programm auf die Diskette, beim Wert 1 auf Kassette. Danach müssen Sie die Startadresse des Speicherbereichs in die Speicherstellen 193 und 194 schreiben, sowie die Endadresse in 174 und 175. Zum Schluß rufen Sie die Routine zum Speichern mit SYS 62957 auf (Alle Basic-Zeiger bleiben dabei unverändert). Das Programm in Listing 3 speichert einen angegebenen Bereich wie oben beschrieben.

Im folgenden ein sehr nützliches Beispiel: Wenn Sie in einem Programm Sprites verwenden, so werden Sie die zugehörigen Daten wahrscheinlich in DATA-Zeilen geschrieben haben, diese dann mit einer FOR-NEXT-Schleife wieder auslesen und an einen bestimmten Speicherplatz POKEn. Das dauert bei vielen Sprites nicht nur sehr lange, es verbraucht auch ungefähr vier- bis fünfmal soviel Platz, wie eigentlich nötig wäre. Wenn Sie die Sprite-Daten allerdings auf Diskette speichern und dann direkt in den Speicher laden, so umgehen Sie beide Nachteile. Dies können Sie am besten folgendermaßen bewerkstelligen:

Laden Sie Ihr »altes« Programm und starten Sie es. Nachdem die Sprite-Daten an die richtige Stelle gePOKEt wurden, können Sie es unterbrechen und löschen. Tippen Sie nun das Programm in Listing 3 ab und starten Sie es. Als Start- und Endadresse geben Sie die entsprechenden Werte für Ihre Sprites ein. Nach Eingabe eines Namens werden die Daten als Programmfile gespeichert. Laden Sie nun abermals Ihr »altes« Programm. Daraus können Sie nun die DATA-Zeilen und die FOR-NEXT-Schleife entfernen. Fügen Sie als erste Zeile folgendes ein:

```
1 IF A=0 THEN A=1 : LOAD "Name",8,1
```

Als »Name« verwenden Sie natürlich den Namen, den Sie beim Speichern angegeben haben. So, jetzt ist Ihr neues Programm fertig und Sie können es speichern.

Wenn Sie sich schon einmal über den INPUT-Befehl des Commodore-Basic geärgert haben, weil er verschiedene Zeichen (zum Beispiel Doppelpunkt, Komma, führende Leerzeichen und so weiter) einfach nicht übernimmt, dann ist hier die Abhilfe: Rufen Sie mit SYS 42336 die Eingaberoutine des Betriebssystems auf. Diese Routine schreibt alle Zeichen in einer logischen Bildschirmzeile (maximal 80 Zeichen) in den Basic-Eingabepuffer ab Adresse 512. Daraus kann man nun mit einer einfachen Schleife die Eingabe einlesen. Das Ende wird mit einem CHR$(0) gekennzeichnet (Das Programm in Listing 4 ist ein Beispiel dafür.)

So, das war's mit den Tips. Viel Spaß und Freude damit.

(Christoph Bergman/ks)