»Apfelmännchens« Diashow

Mit wenig Aufwand kann man die Grafiken des »Apfelmännchens« zu einer effektvollen und abwechslungsreichen Diashow zusammenfassen.

Natürlich darf bei einem Programm wie dem »Apfelmännchen« (64'er 11/85, Seite 80) der Vorführeffekt nicht zu kurz kommen. Um diesen zu erreichen, verändern Sie die Zeilen des Originalprogramms wie in Listing 1 angegeben und ergänzen das Programm mit Listing 2. Die Bilder müssen dazu die unten aufgeführten Namen haben. Bitte beachten Sie das Leerzeichen vor jeder Zahl:

Bild 1.PIC
Bild 2.PIC
Bild 3.PIC
...
Bild 9.PIC
Bild 10.PIC

Die Anzahl der Bilder wird in Zeile 11 000 festgelegt. Um zum Beispiel die Anzahl auf 20 festzulegen, müßte dort stehen:

```
11000 N=N+1:IFN=21THEN RETURN
```

Das nächste Bild wird jeweils durch Druck auf die Space-Taste gezeigt.

(O. Hobert/og)

```
317 PRINT"{DOWN}"TAB(10)"(B){2SPACE}BIA-SH
    OW"                                    <229>
340 GET A$:IF A$<"1"OR A$>"8"THEN 340      <018>
360 ON VAL(A$)GOSUB 1000,2000,3000,4000,50
    00,5500,7000,10000                     <184>
```

Ⓒ 64'er

Listing 1. Die Veränderungen ...

```
10000 REM DIA-SHOW                         <177>
10010 PRINT"{CLR,DOWN}*** BIA-SHOW ***"    <108>
10020 PRINT"{DOWN}FARBEN{9SPACE}: FUNKTION
      STASTEN"                             <228>
10030 PRINT"{DOWN}MENUE{10SPACE}: '←'"     <150>
10040 PRINT"{DOWN}NAECHSTES BILD : 'SPACE'
      "                                    <229>
10050 PRINT"{2DOWN}WEITER MIT TASTE"       <105>
10060 POKE 198,0:WAIT 198,1:GET A$         <078>
10070 N=0                                  <117>
10080 REM LADEN                            <098>
11000 N=N+1:IF N=10 THEN RETURN            <208>
11005 N$=STR$(N)                           <200>
11010 NB$="BILD"+N$+".PIC"                 <057>
11020 FOR I=0 TO LEN(NB$)-1                <046>
11030 POKE BN+I,ASC(MID$(NB$,I+1,1))       <143>
11040 NEXT                                 <048>
11050 POKE BN+16,LEN(NB$)                  <071>
11060 SYS LO                               <090>
11070 REM EFFEKT                           <114>
11090 SYS M1                               <159>
11100 IF A$="{F5}"THEN POKE C2,(PEEK(C2)+1
      )AND 15                              <020>
11110 CP=PEEK(C3):POKE C3,PEEK(C2):POKE C2
      ,PEEK(C1):POKE C1,CP                 <085>
11120 GET A$:IF A$="←"THEN 11190           <201>
11125 IF A$=" "THEN GOTO 11200             <112>
11130 IF A$="{F1}"THEN POKE C0,(PEEK(C0)+1
      )AND 15                              <009>
11140 IF A$="{F3}"THEN POKE C1,(PEEK(C1)+1
      )AND 15                              <168>
11150 IF A$="{F5}"THEN POKE C3,(PEEK(C3)+1
      )AND 15                              <090>
11160 IF A$="{F7}"THEN POKE C3,(PEEK(C3)+1
      )AND 15                              <229>
11170 SYS SC                               <064>
11180 GOTO 11110                           <078>
11190 SYS M0:RETURN                        <221>
11200 SYS M0:GOTO 11000                    <016>
```

Ⓒ 64'er

Listing 2. ... und die Ergänzungen für eine Diashow mit den Grafiken des »Apfelmännchens«

Autochange für C 128

Mit diesem Utility erkennt Ihr C 128 automatisch beim Boot-Versuch, in welchen Modus — C 64 oder C/PM — er springen muß.

Der C 128 sucht nach dem Einschalten zuerst auf der Diskette, ob er einen Boot-Sektor vorfindet. Ein Boot-Sektor ist ein Datenblock auf der Diskette, der beim Einschalten des Computers automatisch geladen und ausgeführt wird. Wird zum Beispiel ein CP/M-Boot-Sektor gefunden, so versucht er anschließend, das CP/M-Betriebssystem hochzufahren. Man kann diesen Sektor dahingehend umändern, daß er bei C 64-Disketten automatisch die Kontrolle an das C 64-Betriebssystem abgibt. Das Programm (Listing 1) generiert einen Boot-Sektor auf einer beliebigen, im Commodore-Format beschriebenen Diskette. Hierdurch wird der C 128 befähigt, eine C 64-Diskette zu erkennen und automatisch nach dem Einschalten in dessen Modus zu springen. Beim Erstellen dieses Startups (Befehlssektors) werden keine eventuell vorhandenen Dateien auf der Diskette geschädigt. Außerdem erkennt das Programm einen möglicherweise im C 128-Modus installierten Boot-Sektor und weist, wenn bestehend, gesondert darauf hin.

Zum Programm:

In den Zeilen 190 bis 230 wird auf das eventuelle Vorhandensein eines Boot-Sektors auf Spur 1/Sektor 0 geprüft. Sollte die Abfrage negativ ausfallen, wird die Installierung fortgesetzt, indem der Inhalt des späteren Startsektors im Blockpuffer aufgebaut wird (Zeile 380 bis 400). Der Blockpuffer enthält fortlaufend die Kennung »CBM«, gefolgt von vier Null-Byte und dem Text, der beim Booten ausgegeben wird (in NAME$ enthalten). Es schließen sich außerdem noch zwei Null-Byte und der eigentliche Sprungbefehl an, der direkt in den C 64-Modus umschaltet. Zeile 410 schreibt den Autochanger endgültig auf die Diskette und kennzeichnet ihn, wenn nötig, in der BAM als belegt. (Manfred Bauer/dm)

```
10 LC=0:COLOR0,1:COLOR4,1:COLOR5,2
100 NAME$="{CLR,7RIGHT,7DOWN}C64 DISKETT
E"
110 PRINT"{CLR,5DOWN,9RIGHT}AUTOMATISCHE
R WECHSEL"
120 PRINT"{DOWN,11RIGHT}IN DEN 64'ER MOD
US"
130 OPEN15,8,15,"I0":A$=CHR$(0)
140 IFDS<>0THENBEGIN
150 PRINT"{2DOWN,RIGHT}DISKETTEN FEHLER:
 ";DS$
160 CLOSE8:CLOSE15
170 PRINT"{2DOWN,RIGHT}ABBRUCH":END
180 BEND
190 OPEN8,8,8,"#"
200 PRINT#15,"U1:8 0 18 0":PRINT#15,"B-P
";8;5
210 GET#8,BA$
220 PRINT#15,"U1:8 0 1 0"
230 FORI=0TO24:GET#8,S$:BL$=BL$+CHR$(ASC
(S$)):NEXT
240 IF(ASC(BA$)AND1)=0THENBEGIN
250 IFLEFT$(BL$,3)="CBM"THENPRINT"{2DOWN
,6RIGHT}BOOT-SECTOR SCHON VORHANDEN"
270 IFLEFT$(BL$,3)<>"CBM"THENPRINT"{2DOW
N,5RIGHT}PROGRAMM LIEGT AUF BOOT-SECTOR"
280 PRINT"{2DOWN,4RIGHT}INSTALLIERUNG FO
RTSETZEN? (J/N)" :LC=1
290 GETKEYX$
300 IFX$<>"J"THENBEGIN
310 CLOSE8:CLOSE15
320 PRINT"{2DOWN,5RIGHT}ABBRUCH":END:BEN
D
330 BEND
340 PRINT"{DOWN,5SPACE}INSTALLIEREN DES
BOOT-SEKTORS{CTRL-@}H{GREEN}Y{CTRL-A,LIG
.GREEN}"    BITTE ENTFERNEN SIE EINEN EV
TL.
350 PRINT"{5SPACE}VORHANDENEN SCHREIBSCH
UTZ UND{CTRL-@}&{BLUE}H{CTRL-A,LIG.GREEN
}"        DRUECKEN SIE EINE TASTE
370 GETKEYX$
380 PRINT#15,"B-P 8 0"
390 PRINT#8,CHR$(67);CHR$(66);CHR$(77);A
$;A$;A$;A$;NAME$;A$;
400 PRINT#8,A$;CHR$(DEC("20"));CHR$(DEC(
"4D"));CHR$(DEC("FF"));A$
410 PRINT#15,"U2:";8;0;1;0
415 IFLC=0THENPRINT#15,"B-A 0 1 0"
420 CLOSE8:CLOSE15
430 PRINT"{4SPACE}BOOT-SECTOR INSTALLIER
T":END
```

64'er

Listing 1. Autochange 128/64. Bitte im C 128-Modus eingeben.

Das Maß der Dinge

Das exakte Ausmessen eines Unterprogrammes ist eine große Hilfe, um Laufzeiten zu verringern. Meist steht jedoch keine hinreichend genaue Uhr zur Verfügung. Wir bieten Ihnen zwei Lösungen an, eine für Maschinensprache- und eine für Basic-Programme.

Um die Laufzeiten von Programmen genau messen zu können, reicht die Stoppuhr für den 100-Meter-Lauf nicht mehr aus. Vor allem bei Maschinenprogrammen, bei denen die menschliche Reaktionsfähigkeit weit hinter der geforderten Genauigkeit bleibt. Doch auch die eingebaute interruptgesteuerte Uhr TI$ im C 64 versagt, wenn es um das Auszählen sehr kurzer Maschinenprogramme geht. Ladevorgänge können damit überhaupt nicht erfaßt werden, da TI$ während des Ladens stillsteht. Abhilfe schaffen hier die Timer der CIAs.

Taktzyklen zählen

Um Maschinensprache-Programme zu messen, ist es am besten, die Taktzyklen zu zählen. Dies liefert einen gut zu vergleichenden Wert. Geben Sie zunächst das Programm »Taktzyklen« (Listing 1) mit dem MSE ein und speichern Sie es. Aufgerufen wird die Routine durch den Befehl

SYS 49152,Startadresse <RETURN>

Die anzugebende Startadresse entspricht der des zu untersuchenden Maschinenprogrammes. Dieses Unterprogramm wird nun ausgeführt und die dafür benötigte Zeit bestimmt. Während dieses Ablaufes ist der Bildschirm ausgeblendet, er nimmt die Farbe des Rahmens an. Am Ende des zu messenden Programmes muß ein RTS stehen. Nach Programmende gibt das Programm den gemessenen Wert aus und springt in den Direkt-(READY.-)Modus. Der angezeigte Zahlenwert drückt die Anzahl der Systemtakte aus, die während der Abarbeitung des Maschinen-Unterprogrammes durch den Mikroprozessor verstrichen sind. Da ein solcher Takt etwa 1,015 Mikrosekunden dauert, läßt sich durch Multiplizieren mit diesem Faktor die benötigte Zeit in Mikrosekunden errechnen. Bei eingeschaltetem Bildschirm vergrößert sich das Ergebnis um etwa 5 bis 6 Prozent.

Beschreibung des Programms »Taktzyklen«

Die beim Aufruf des Programms übergebene Startadresse des zu messenden Unterprogramms wird als Operand für einen Sprungbefehl gespeichert. Anschließend wird die Möglichkeit der Unterbrechung durch einen IRQ ausgeschaltet, weil sonst ein falsches Resultat geliefert werden kann. Nach dem Ausblenden des Bildschirms lädt das Programm beide Timer des CIA 2 mit dem Maximalwert und koppelt sie miteinander. Sobald die Zähler gestartet sind, erfolgt der Sprung in das Unterprogramm, nach dessen Ende die Zähler sofort wieder gestoppt werden. Schließlich werden neben der Einrichtung des Normalzustandes — den Bildschirm und den IRQ betreffend — die Timer-Bytes ausgewertet. Dazu invertiert das Programm sämtliche Bits der von den Timern gelieferten Werte, da beide Registerpaare heruntergezählt werden. Zusätzlich werden noch elf Takte subtrahiert, die zum Starten beziehungsweise Stoppen der Timer und zum Aufruf des Unterprogramms nötig sind. Damit erhält man einen Meßbereich, der zwischen 1 und 2^{32}-11 Taktzyklen (entsprechend einer Zeit von 1 µs bis 1 Stunde 12 Minuten) liegt.

Laufzeit-Meß-System »LMS«

Das LMS wurde entwickelt, um die Laufzeit von Basic-Programmen zeilenbezogen messen zu können. Weiterhin sollte