

# »Fenster«-Befehle für den C16

Dieses Programm stellt den im C16-Basic leider nicht vorhandenen »Window«-Befehl zur Verfügung und macht Schluß mit dem umständlichen Hantieren mit den ESC-Funktionen.

**W**er im Befehlssatz des Basic 3.5 nach dem Befehl »Window« sucht, wird nicht fündig. Die groß angekündigte moderne Window-Technik läßt doch zu wünschen übrig. Ein Window, also ein Bildschirmfenster, kann nur mit Hilfe der ESC-Taste definiert werden.

So muß man, um ein Window zu setzen, mit dem Cursor an die linke obere Ecke des gewünschten Bildschirmfensters fahren und die Tasten ESC und T drücken. Die rechte untere Ecke des Fensters wird ebenfalls mit dem Cursor gewählt und mit ESC und B gesetzt. Erst jetzt ist das Fenster definiert.

Im Direktmodus ist dieser Aufwand nicht hinderlich, sondern bringt eher Vorteile, da kein Befehl »WINDOW« am Bildschirm erscheint. Im laufenden Programm ist es jedoch sehr umständlich, ein Fenster zu generieren. Es sind sehr viele PRINT-Anweisungen nötig.

Da die Window-Funktion jedoch in vielen Programmen nützlich ist, lohnt es sich schon, diese Funktion komfortabel

als Basic-Befehl zur Verfügung zu haben. Ein kleines Maschinenprogramm macht's möglich.

Nach Eintippen des DATA-Laders (Listing 1) sollten Sie das Programm als erstes unbedingt abspeichern, da es sich nach Kontrolle der Prüfsumme selbst initialisiert und anschließend der DATA-Lader gelöscht wird.

Haben Sie das Programm abgespeichert, dann können Sie es mit »RUN« starten und haben kurz danach den neuen Basic-Befehl »FENSTER« zur Verfügung.

Der FENSTER-Befehl hat folgendes Format und kann in jedem Basic-Programm verwendet werden:

FENSTER AZ,AS,EZ,ES

Die Parameter haben folgende Bedeutung:

AZ = AnfangsZeile  
AS = AnfangsSpalte  
EZ = EndZeile  
ES = EndSpalte

```

10 REM *****
20 REM *
30 REM * FENSTER FUER C 16 & 116 *
40 REM *
50 REM * CHRISTIAN QUIRIN SPITZNER *
60 REM * GRUBERSTRASSE 53 *
70 REM * 8011 POING *
80 REM *
90 REM *****
100 :
110 :
120 REM *** SPEICHER BEGRENZEN ***
130 :
140 POKE 53,255 : POKE 54,62 : POKE 55,2
55 : POKE 56,62 : CLR
150 :
160 REM *** ERKLAERUNG ***
170 :
180 PRINT "{CLR,DOWN}"TAB(12)"***{SPACE,F
LASHON,RED}FENSTER{FLASHOFF,BLACK }***"
190 PRINT "{2DOWN,6SPACE}FORMAT:"
200 PRINT "{DOWN,6SPACE,PURPLE,RVSON}FENS
TER AZ,AS,EZ,ES{BLACK}"
210 PRINT "{DOWN,6SPACE}AZ = ANFANGSZEILE
"
220 PRINT "{6SPACE}AS = ANFANGSSPALTE"
230 PRINT "{6SPACE}EZ = ENDZEILE"
240 PRINT "{6SPACE}ES = ENDSPALTE"
250 PRINT "{DOWN,6SPACE}NICHT IM DIREKTMO
DUS !"
260 :
270 REM DATA'S EINLESEN ***
280 :
290 FOR I=16128 TO 16383
300 : READ P
310 : Z=Z+P
320 : POKE I,P
330 NEXT I
340 IF Z <> 30097 THEN PRINT "{DOWN,6SPAC
E,FLASHON}DATA ERROR{FLASHOFF}":END
350 :
360 REM *** INITIALISIEREN ***
370 :
380 SYS 16128
390 :

```

```

400 REM *** FENSTER-BEFEHL AUFRUFEN ***
410 :
420 FENSTER 4,1,25,40
430 :
440 REM *** BASIC-LOADER LOESCHEN ***
450 :
460 NEW
470 :
480 REM *** DATA'S ***
490 :
500 DATA 169,024,141,008,003,169,063,141
510 DATA 009,003,162,000,189,205,063,032
520 DATA 210,255,232,224,043,208,245,096
530 DATA 162,000,032,115,004,221,209,063
540 DATA 208,007,232,224,007,208,243,240
550 DATA 006,032,121,004,076,217,139,032
560 DATA 183,251,032,115,004,032,132,157
570 DATA 142,249,063,032,145,148,032,132
580 DATA 157,142,250,063,032,145,148,032
590 DATA 132,157,142,251,063,032,145,148
600 DATA 032,132,157,142,252,063,174,249
610 DATA 063,224,000,208,003,076,161,148
620 DATA 224,024,016,249,174,251,063,224
630 DATA 026,016,242,236,249,063,048,237
640 DATA 174,250,063,224,000,240,230,224
650 DATA 041,016,226,172,252,063,224,041
660 DATA 016,219,236,250,063,048,214,162
670 DATA 000,222,249,063,232,224,004,208
680 DATA 248,173,249,063,133,205,173,250
690 DATA 063,133,202,169,027,032,210,255
700 DATA 169,084,032,210,255,173,251,063
710 DATA 133,205,173,252,063,133,202,169
720 DATA 027,032,210,255,169,066,032,210
730 DATA 255,169,019,032,210,255,032,193
740 DATA 251,032,121,004,076,217,139,000
750 DATA 000,000,000,000,000,147,013,032
760 DATA 130,070,069,078,083,084,069,082
770 DATA 132,032,040,067,041,032,067,072
780 DATA 082,073,083,084,073,065,078,032
790 DATA 081,085,073,082,073,078,032,083
800 DATA 080,073,084,090,078,069,082,013
810 DATA 013,000,000,000,000,000,000,000

```

064'er

Listing 1. Der DATA-Lader zum »Fenster«-Befehl

Gibt man im Programm beispielsweise die folgende Zeile 10 ein, so wird ein Fenster von Spalte 5 der fünften Zeile bis einschließlich Spalte 35 der 15. Zeile definiert.

```
10 FENSTER 5,5,35,15
```

Das Fenster kann durch zweimaliges Drücken der HOME-Taste oder durch Generieren eines neuen Fensters gelöscht werden.

Dem Fenster-Befehl müssen auf jeden Fall vier Parameter folgen. Für die Parameter AZ und EZ sind nur Zahlen zwischen 1 und 25 zulässig. AS und ES dürfen nur zwischen 1 und 40 liegen. Außerdem ist zu beachten, daß AZ auf keinen Fall größer als EZ, und AS nicht größer als ES gewählt wird.

Der Fenster-Befehl darf auch nicht im Direktmodus verwendet werden. Sollten diese Bedingungen nicht beachtet werden, erscheint die Fehlermeldung »SYNTAX ERROR«.

Das Beispielprogramm Fenster-Demo (Listing 2) generiert zufällige Textfenster und füllt diese farbig aus.

Der FENSTER-Befehl verkürzt ein Basic-Programm bei häufiger Anwendung gegenüber der konventionellen Methode über PRINT-Anweisungen ganz erheblich, so daß die vom Maschinenprogramm beanspruchten 255 Byte trotz des mageren C 16-Speichers kaum ins Gewicht fallen dürften.

(Christian Quirin Spitzner/ev)

```
100 REM FENSTER-DEMO
110 :
120 COLOR 0,1 : COLOR 4,1 : SCNCLR
130 :
140 DO
150 : AZ = INT(RND(TI)*12)+1
160 : EZ = AZ*2+1
170 : AS = INT(RND(TI)*20)+1
180 : ES = AS*2
190 :
200 : FENSTER AZ,AS,EZ,ES
210 :
220 : PRINT "{CLR}";
230 : COLOR 1,RND(TI)*15+2,RND(TI)*8
240 : E = (ES-AS+1)*(EZ-AZ+1)-1
250 : FOR I=1 TO E
260 :   PRINT "{RVSON,SPACE,RVOFF}";
270 : NEXT
280 : PRINT "{LEFT,RVSON}";CHR$(148); "{SPACE,RVOFF}";
290 LOOP
```

64'er

Listing 2. Ein Demo-Programm zum »Fenster«-Befehl

# Zeichensatz selbstgemacht

**Wollen Sie einen eigenen Zeichensatz definieren? »Charmachine« ermöglicht Ihnen dies in einer komfortablen Weise. Da Sie auch mehrfarbige Zeichen erzeugen können, ist es besonders für Grafiken interessant.**

**D**ie Möglichkeiten des C 16/116 im Bereich der hochauflösenden Bitmap-Grafik sind schier riesig. Doch für viele Programme, die grafischer Unterstützung bedürfen, sind die nach dem Befehl GRAPHIC 1-4 noch vorhandenen 2 KByte nicht ausreichend. In diesem Fall muß der ebenfalls hochauflösende und/oder mehrfarbige Zeichensatz aushelfen. »Charmachine« dient dazu, diesen bequem editieren zu können.

Nun zum Umgang mit dem Programm. Nachdem Sie das Programm (Listing) eingetippt und gestartet haben, erscheint in der linken oberen Ecke des Bildschirms ein 8x8 Punkte großes Feld. Rechts daneben sind die Befehle von »Charmachine« zu sehen. Im unteren Teil des Bildschirms befindet sich ein Feld, in dem alle 256 möglichen Zeichen abgebildet sind. Es handelt sich dabei um die positiven Zeichen beider Zeichensätze (also des Groß-Klein- und des Groß-Grafik-Satzes). Die negativen Zeichen beim C 16/116 können nicht gesondert verändert werden, sie zeigen immer genau die Negative ihrer entsprechenden positiven Zeichen.

In diesem Feld können Sie einen Cursor bewegen und sich das zu verändernde Zeichen aussuchen. Das Zeichen unter dem Cursor wird unterhalb des 8x8-Feldes hochauflösend (hiRes) und mehrfarbig (multicolor) dargestellt. Steht nun der Cursor auf dem richtigen Zeichen, so drücken Sie die Taste P (P=pick). Das Zeichen wird augenblicklich im 8x8-Punktefeld dargestellt, und der Cursor befindet sich ebenfalls dort.

Sie können nun das Zeichen ändern mit SPACE für Punkt löschen (SPACE = unplot), Z für Punkt setzen (Cursor nach rechts) und X für Punkt setzen (Cursor nach unten) (Z/X = plot). Man kann das ganze Feld löschen mit C (=Clear) und invertieren mit R (=reverse). Ist das Zeichen fertig editiert, so kehren Sie mit M (=select mode) in den Wählmodus zurück. Hier können Sie mit den Tasten 1 und 2 (=colors) die Multicolorfarben des Zeichens über »multik« ändern sowie den Zeichensatz speichern mit S (=SAVE) und neu laden mit L (=LOAD von Diskette). Der Befehl C (=COPY) erlaubt Ihnen, das sich noch im 8x8-Feld befindliche Zeichen in ein anderes Zeichen zu kopieren, das mit dem Cursor ausgesucht wird.

Wenn Sie einen fertigen gespeicherten Zeichensatz in ein eigenes Programm einbauen wollen, so müssen Sie diesen mit LOAD "name" 8,1 laden und dann den Befehl NEW eingeben. An den Anfang Ihres Programms müssen Sie die Befehle setzen, die in Zeile 10 von »Charmachine« (Listing) stehen. Sie dienen dazu, den Zeichensatz vor dem Überschreiben durch Strings zu schützen. Der C 16/116 hat dann noch 10 KByte Speicher frei. Um den Zeichensatz schließlich einzuschalten, müssen Sie folgende POKes eingeben:

```
POKE 65298,0 (Flag für Zeichensatz aus ROM oder RAM),
POKE 65299,56 (Anfangsadresse des Zeichensatzes im RAM),
POKE 65287,8+(16)+128
```

Beim letzten Befehl steht 8 für 40 Zeichen, (16) für Multicolor (wahlweise), 128 für Darstellung der Zeichensätze Groß+Klein und Groß+Grafik. Ohne 128 wird nur einer dieser Zeichensätze, aber mit seinen negativen Zeichen, dargestellt.

Der undefinierte Zeichensatz liegt im Speicherbereich von 14336 bis 16383 (\$3800 bis \$3FFF).

Nun zu einigen Besonderheiten des Programms. Beim Kopieren des Zeichensatzes vom ROM ins RAM greift das Programm auf den Monitor des C 16/116 zurück. Vom Programm aus wird in den Monitor gesprungen und dort mit dem Befehl T (=TRANSDIRE) der Zeichensatz von ROM ins RAM geschrieben. Um in den Monitor und wieder zurück zu kommen, wird der »programmierte Direktmodus« benutzt, wobei die erforderlichen Befehle auf den Bildschirm gedruckt werden und durch Auffüllen des Tastaturpuffers mit dem ASCII-