Record-Befehl für den C 64

Dieses Programm erleichtert Ihnen den Umgang mit relativen Files.

Bei relativen Files kann auf jeden Satz des Files direkt zugegriffen werden, ohne die vorherigen Sätze lesen zu müssen. Außerdem kann ein eröffnetes File nach Belieben beschrieben und gelesen werden, es ist also zum Beispiel möglich, Satz 12 zu beschreiben, auch wenn bereits höhere Sätze existieren. Ein relatives File wird durch folgenden OPEN-Befehl eingerichtet:
OPEN lfn,ga,sa,"filename,l,"+chr$(sl). Dabei bedeutet:
lfn = logische Filenummer.
ga = Geräteadresse (normalerweise 8)
sa = Sekundäradresse (mindestens 2, höchstens 14)
sl = maximale Satzlänge des relativen Files (mindestens 1, höchstens 254). Diese Angabe ist entscheidend für den späteren Diskettenplatzverbrauch des Files, da auch Sätze, die nicht mit voller Satzlänge beschrieben werden, den gesamten durch sl reservierten Platz verbrauchen. Ist ein relatives File erst einmal eingerichtet, so kann die Satzlänge nicht mehr verändert werden. Es genügt dann der folgende OPEN-Befehl:
OPEN lfn,ga,sa,"filename"

Selbstverständlich muß ein relatives File auch mit CLOSE lfn geschlossen werden. Auf die einzelnen Sätze eines relativen Files wird normal mit PRINT#lfn, INPUT#lfn oder GET#lfn zugegriffen, allerdings muß vor jedem Zugriff auf den gewünschten Satz positioniert werden. Dies übernimmt die vorliegende Basic-Erweiterung.

Durch **RECORD#lfn,sn,(,p)** wird auf den Satz mit der Nummer sn positioniert, wird zusätzlich noch p angegeben, so wird innerhalb des Satzes das p-te Zeichen angewählt. Der neue Befehl muß von THEN durch einen Doppelpunkt abgetrennt werden. Es können folgende Basic-Fehlermeldungen auftreten:
?SYNTAX ERROR: Schreibfehler im Befehl oder die Erweiterung war nicht eingeschaltet.
?ILLEGAL QUANTITY ERROR: Ein Argument lag außerhalb des erlaubten Bereichs.
?FILE NOT OPEN ERROR: Das File mit der logischen Filenummer lfn war nicht eröffnet.
?ILLEGAL DEVICE NUMBER ERROR: Dem File mit der Nummer lfn ist ein Gerät mit einer Adresse kleiner als 8 oder größer als 15 zugeordnet.
Es können außerdem folgende Floppy-Fehlermeldungen auftreten (Sie müssen durch den Fehlerkanal ausgelesen werden):

Bild 2. Symbol-Tabelle

0073	chrget	ffb1	listen	c036	l2
0079	chrgot	ff93	seclis	c041	tab1
007a	chrvek	ffa8	iecout	c047	tab2
0308	decod	ffae	unlist	c04b	record
a437	error	0014	integ	c05a	13
a7ae	schleife	00b8	filenr	c067	simon
ad8a	getrec	00b9	secadr	c06c	13s
aefd	chkcom	00ba	ga	c079	out
b79e	getbyt	c019	endoff	c082	ok
b7f1	comget	c01a	vekt	c091	14
b7f7	fac16	c01c	flag	c09f	illdev
f30f	search	c01d	on	c0a4	15
f31f	parset	c02c	l1	c0da	ende

```
20 FOR I=49152 TO 49375:READ A:POKE I,A:S=
   S+A:NEXT I                                <193>
30 IF S<>25982 THEN PRINT"PRUEFSUMMENFEHLE
   R":STOP                                   <103>
40 SYS 49152:NEW                             <225>
100 DATA 76,29,192,173,28,192,240,17,173,2
    6,192,141,8,3,173,27,192,141,9,3         <071>
101 DATA 169,0,141,28,192,96,228,167,0,169
    ,255,141,28,192,173,8,3,201,75,240       <151>
102 DATA 3,141,26,192,173,9,3,201,192,240,
    3,141,27,192,169,75,141,8,3,169          <045>
103 DATA 192,141,9,3,96,35,68,176,67,69,82
    ,35,68,176,8,165,122,72,165,123          <022>
104 DATA 72,32,115,0,201,100,240,15,160,6,
    217,64,192,208,26,32,115,0,136,208       <102>
105 DATA 245,240,27,32,115,0,160,4,217,70,
    192,208,8,32,115,0,136,208,245,240       <246>
106 DATA 9,104,133,123,104,133,122,108,26,
    192,104,104,32,158,183,32,15,243         <096>
107 DATA 240,5,162,3,76,55,164,32,31,243,1
    65,185,41,15,72,165,186,201,16,144       <130>
108 DATA 5,162,9,76,55,164,201,8,144,247,3
    2,177,255,169,111,133,185,32,147         <244>
109 DATA 255,169,80,32,168,255,32,253,174,
    32,138,173,32,247,183,104,32,168         <230>
110 DATA 255,165,20,32,168,255,165,21,32,1
    68,255,32,121,0,240,7,32,241,183         <238>
111 DATA 138,32,168,255,32,174,255,76,174,
    167                                      <152>
```

Listing 1. Basic-Lader »Record-Befehle«.
Bitte beachten Sie die Eingabehinweise auf Seite 6.

50,RECORD NOT PRESENT: Der Satz, auf den positioniert wurde, existiert nicht. Diese Meldung kann ignoriert werden, wenn der Satz beschrieben werden soll, denn durch das Beschreiben wird er eingerichtet. (Und alle nicht existierende Sätze mit kleinerer Nummer ebenfalls.) Aus Geschwindigkeitsgründen empfiehlt es sich, bei der Einrichtung eines Files den höchsten Satz zuerst zu beschreiben.
51,OVERFLOW IN RECORD: Die maximale Satzlänge sl wurde beim Schreiben überschritten. Zu beachten ist, daß das Carriage Return am Ende des Satzes mitzählt.
52,FILE TOO LARGE: Die Diskette ist voll, der letzte Schreibzugriff kann nicht durchgeführt werden.

Aufbau des Programms

Der Basic-Interpreter besitzt einen Zeiger (Adresse $0308, im Listing »decod« genannt), der auf die Routine zur Befehlsinterpretation zeigt. Dieser Vektor wird nach dem Aufruf der Erweiterung zuerst gemerkt und danach auf eine eigene Routine zur Befehlsinterpretation gesetzt. Vor dem Merken wird noch geprüft, ob der Vektor schon auf die eigene Routine zeigt. Damit wird verhindert, daß der Computer bei einem versehentli-

Bild 1. Befehlscodierung

```
*= 49152
chrget   = $73      ;holt nächstes Zeichen
chrgot   = $79      ;holt letzes Zeichen
chrvek   = $7a      ;Chrgetzeiger
decod    = $0308    ;Vektor für Befehlsdec.
error    = $a437    ;Fehlermeldung ausgeben
schleife = $a7ae    ;Interpreterschleife
getrec   = $ad8a    ;Recordnummer holen
chkcom   = $aefd    ;prüft auf Komma
getbyt   = $b79e    ;holt Byte in x
comget   = $b7f1    ;chkcom + getbyt
fac16    = $b7f7    ;FAC nach 16-Bit wandeln
search   = $f30f    ;sucht logische Filenr.
parset   = $f31f    ;setzt Fileparameter
listen   = $ffb1    ;LISTEN senden
seclis   = $ff93    ;Sekundärad. nach LISTEN
iecout   = $ffa8    ;Ausgabe auf IEC-bus
unlist   = $ffae    ;UNLISTEN senden

integ    = $14      ;Integer Wert
filenr   = $b8      ;logische Filenummer
secadr   = $b9      ;Sekundäradresse
ga       = $ba      ;Gerätenummer
```

```
      * = 49152 ;Startadresse

        jmp on          ;zum Einbinden          c000 4c 1d c0

;******* Erweiterung abschalten *******
        lda flag        ;schon abgeschaltet? c003 ad 1c c0
        beq endoff                           c006 f0 11
        lda vekt        ;Vektor wieder her-  c008 ad 1a c0
        sta decod       ;stellen             c00b 8d 08 03
        lda vekt+1                           c00e ad 1b c0
        sta decod+1                          c011 8d 09 03
        lda #0          ;Flag fuer Abge-     c014 a9 00
        sta flag        ;schaltet setzen     c016 8d 1c c0
endoff rts                                   c019 60
vekt    .word $a7e4                          c01a e4 a7
flag    .byte $00                            c01c 00

;******* Erweiterung einschalten ******
on      lda #$ff        ;Flag fuer Einge-    c01d a9 ff
        sta flag        ;schaltet setzen     c01f 8d 1c c0
        lda decod                            c022 ad 08 03
        cmp #<record                         c025 c9 4b
        beq 11                               c027 f0 03
        sta vekt                             c029 8d 1a c0
11      lda decod+1                          c02c ad 09 03
        cmp #>record                         c02f c9 c0
        beq 12                               c031 f0 03
        sta vekt+1                           c033 8d 1b c0
12      lda #<record                         c036 a9 4b
        sta decod                            c038 8d 08 03
        lda #>record                         c03b a9 c0
        sta decod+1                          c03d 8d 09 03

        rts                                  c040 60
;****** Interpreterdarstellung von ****
;****** RECORD#                    ****
tab1    .byte $23,$44,$b0,$43,$45,$52        c041 23 44 b0 43 45 52
tab2    .byte $23,$44,$b0,$08                c047 23 44 b0 08

;****** Test auf RECORD#           ****
record
        lda chrvek      ;Chrgetzeiger        c04b a5 7a
        pha             ;retten              c04d 48
        lda chrvek+1                         c04e a5 7b
        pha                                  c050 48
        jsr chrget                           c051 20 73 00
        cmp #$64        ;Simon's Basic       c054 c9 64
        beq simon       ;TOKEN               c056 f0 0f
        ldy #6                               c058 a0 06
13      cmp tab1-1,y                         c05a d9 40 c0
        bne out                              c05d d0 1a
        jsr chrget                           c05f 20 73 00
        dey                                  c062 88
        bne 13                               c063 d0 f5
        beq ok                               c065 f0 1b
simon   jsr chrget                           c067 20 73 00
        ldy #4                               c06a a0 04
13s     cmp tab2-1,y                         c06c d9 46 c0
        bne out                              c06f d0 08
        jsr chrget                           c071 20 73 00
        dey                                  c074 88
        bne 13s                              c075 d0 f5
        beq ok                               c077 f0 09
;****** Weiter mit normalem Basic  ****
out     pla                                  c079 68
        sta chrvek+1                         c07a 85 7b
        pla                                  c07c 68
        sta chrvek                           c07d 85 7a
        jmp (vekt)                           c07f 6c 1a c0
;****** eigentlicher RECORD#-Befehl ***
ok      pla             ;Zeiger von Stack    c082 68
        pla                                  c083 68
        jsr getbyt  ;holt Byte nach x        c084 20 9e b7
        jsr search  ;sucht log. Filenr.      c087 20 0f f3
        beq 14      ;gefunden                c08a f0 05
        ldx #3      ;file not open           c08c a2 03
        jmp error   ;ausgeben                c08e 4c 37 a4
14      jsr parset  ;Fileparameter setzen    c091 20 1f f3
        lda secadr  ;Sekundaeradresse        c094 a5 b9
        and #$0f                             c096 29 0f
        pha         ; = Kanalnummer          c098 48
        lda ga                               c099 a5 ba
        cmp #16                              c09b c9 10
        bcc 15      ;ga < 16                 c09d 90 05
illdev  ldx #9      ;illegal device nr.      c09f a2 09
        jmp error                            c0a1 4c 37 a4
15      cmp #8                               c0a4 c9 08
        bcc illdev  ;ga <  8                 c0a6 90 f7
        jsr listen                           c0a8 20 b1 ff
        lda #$6f    ;15 and $60              c0ab a9 6f
        sta secadr                           c0ad 85 b9
        jsr seclis                           c0af 20 93 ff
        lda #'p     ;positionieren           c0b2 a9 50
        jsr iecout  ;senden                  c0b4 20 a8 ff
        jsr chkcom                           c0b7 20 fd ae
        jsr getrec  ;Recordnummer holen      c0ba 20 8a ad
        jsr fac16   ;FAC nach 16-Bit         c0bd 20 f7 b7
        pla         ;Kanalnummer             c0c0 68
        jsr iecout                           c0c1 20 a8 ff
        lda integ   ;Recordnr. lo            c0c4 a5 14
        jsr iecout                           c0c6 20 a8 ff
        lda integ+1 ;Recordnr. hi            c0c9 a5 15
        jsr iecout                           c0cb 20 a8 ff
        jsr chrgot  ;letztes Zeichen         c0ce 20 79 00
        beq ende    ;kein dritter Param.?    c0d1 f0 07
        jsr comget  ;Komma und Parameter     c0d3 20 f1 b7
        txa                                  c0d6 8a
        jsr iecout  ;Position in Record      c0d7 20 a8 ff
ende    jsr unlist                           c0da 20 ae ff
        jmp schleife;zur Interpreterschl.    c0dd 4c ae a7
```

Listing 2. Source-Code von »Record-Befehl«

chen Doppelaufruf abstürzt. Die eigene Routine rettet zunächst den Zeiger, der auf das augenblickliche Zeichen im Basic-Text zeigt, und ruft dann die CHRGET-Routine (Befehlscodierung siehe Bild 1, Symboltabelle Bild 2) auf. Diese Routine holt das nächste Zeichen aus dem Basic-Text. Sie überliest automatisch Leerzeichen. Wenn das Ende einer Zeile oder ein Doppelpunkt erreicht wird, so wird die Z-Flag des Prozessors gesetzt. Ein Problem ergibt sich dadurch, daß in »RECORD#« der Simons Basic-Befehl »REC« vorkommt. Er wird bei Verwendung von Simons Basic also in eine interne Darstellung beginnend mit $64 umgewandelt. Das Programm testet daher, ob $64 vorliegt und verzweigt bei positiver Antwort. In beiden Fällen wird nun getestet, ob der Befehl »RECORD#« vorliegt. Unter dem Namen tab 1 (beziehungsweise tab 2 für Simons Basic) steht (rückwärts) die interne Darstellung des Befehls »RECORD«. Dabei ist zu beachten, daß »OR« wiederum nicht im ASCII-Code abgelegt ist, sondern als Token. Wird nur eine Abweichung gefunden, so wird der gemerkte CHRGET-Zeiger zurückgeholt und zur normalen Interpreterroutine gesprungen, anderenfalls tritt eine eigene Routine in Funktion (sie beginnt beim Label OK). Sie holt die Satznummer und, wenn angegeben, die Position im Satz aus dem Basic-Text und sendet die Informationen an die Floppy über die Sekundäradresse 15 (Kommandokanal). Die Floppy erwartet die Werte in folgender Form: »p« als Zeichen, das positioniert werden soll; die Sekundäradresse, mit der das relative File eröffnet wurde; die Satznummer in der Reihenfolge niederwertiges — höherwertiges Byte und, falls angegeben, die Position im Satz. Es werden folgende Routinen des Interpreters und des Betriebssystems verwendet:

getbyt : Holt einen beliebigen Ausdruck in das x-Register.
search : Sucht nach der logischen Filenummer in x.
parset : Holt logische Filenummer, Geräteadresse und Sekundäradresse in die Speicherzellen $b8,$ba und $b9
listen : Teilt dem angesprochenen Gerät mit, daß es Daten empfangen soll.
seclis : Sendet die Sekundäradresse zum angesprochenen Gerät.
iecout : Sendet den Wert im Akkumulator zum angesprochenen Gerät.
chkcom : Prüft auf Komma im Basic-Text.
getrec : Holt einen numerischen Ausdruck in den Fließkomma-Akkumulator.
fac16 : Wandelt den Wert im Fließkomma-Akkumulator in eine 16-Bit-Zahl und speichert sie in $14 und $15.
comget : =chkcom + getbyte.
unlist : Beendet die Übertragung an die Floppy.

Das Programm wurde mit dem Assembler »ASSI« übersetzt, eine Übernahme auf andere Assembler dürfte nicht schwierig sein. Der Basic-Lader (Listing 1, Source-Code: Listing 2) legt das Programm ab der Adresse 49152 ($C000) ab, schaltet die Erweiterung ein und löscht sich dann selbst, er muß also vorher auf Diskette abgespeichert werden. Die Erweiterung wird durch SYS49152 ein- und durch SYS49155 ausgeschaltet. STOP/RESTORE schaltet die Erweiterung nicht ab.

(Bernward Bretthauer/rg)