Rüdiger Bäcker

Text mit Steuerzeichen

Erweiterung des Texteditors beim 68008-NDR-Computer

Dieses Programm bietet den Benutzern des 68008-NDR-Klein-Computers eine nützliche Erweiterung des Texteditors: Man kann nun beim Erstellen von Texten Steuerzeichen für den Drucker einfügen. Diese Steuerzeichen werden dann beim Ausdruck berücksichtigt und als Befehle an den Drucker übergeben.

Da man im Editor keine ESC-Sequenzen in den Speicher eingeben kann, wird hier ein anderer Weg beschritten, um die Steuerung des Druckers zu ermöglichen: Die Steuerzeichen werden in einer Befehlstabelle abgelegt, und im Text wird hinter einem Erkennungszeichen eine Zahl bzw. ein Buchstabe für die jeweilige Steuersequenz gesetzt, z. B. "Text ^2" für die Tabulatorfunktion. Dabei ist die Zahl der möglichen Steuersequenzen nur durch die Anzahl der verwendbaren ASCII-Zeichen begrenzt. Im Regelfall sind bereits die Ziffern 0...9 ausreichend, um alle vorhandenen Druckerfunktionen zu nutzen.

Programm selbst ist relokativ und kann aus der Bibliothek aufgerufen werden. Es bezieht sich immer auf den aktuellen Text, dessen Startadresse mittels der Grundprogrammroutine GETSTX ermittelt wird. Nach dem Aufruf von GETSTX steht die aktuelle Textstartadresse in D0; sie wird dann in A0 übertragen, so daß A0 immer auf die aktuelle Stelle im Text zeigt.

Im Programmteil „loop" wird immer ein Zeichen aus dem Speicher in D0 geholt und geprüft, ob es sich um das Endezeichen des Textes oder um das Erkennungszeichen für eine Steuersequenz handelt. Ist das nicht der Fall, so wird das Zeichen an den Drucker übergeben. Beim Auftauchen des Endezeichens springt das Programm an die Marke „ende" und von dort in das Grundprogramm zurück. Trifft die Routine auf das Erkennungszeichen, so verzweigt das Programm zur Marke „steuer". Hier wird

```
kopf:

dc.l $55aa0180          * Kennung Bibliothek
dc.b 'St Druck'
dc.l start-kopf
dc.l enda-kopf
dc.b 1
dc.b 0,0,0
dc.l 0,0

start:
move #66,d7 ; @getstx   * Startadresse des aktuellen Textes holen
trap #1
movea.l d0,a0           * in a0, a0 ist dann Zeiger auf Position im Text

loop:
move.b (a0)+,d0         * Zeichen von (a0) in d0
cmp.b #0,d0             * ist es das Endezeichen ?
beq ende                * Ja, dann zu Ende
cmp.b #'^',d0           * ist es das Codezeichen fuer Steuerfunktionen
beq steuer              * Ja, dann Steuerfunktion holen
move #22,d7 ; @lo       * sonst Zeichen ausdrucken
trap #1
bra loop                * und weiter

ende:                   * springt zum Hauptmenue zurueck
rts

steuer:                 * Hier werden die Steuerzeichen ausgegeben
move.b (a0),d1          * Zeichen nach ^ in d1
lea beftab(pc),a1       * anf. Befehlstabelle in a1
clr d2                  * Zeiger loeschen

stloop:
cmp.b 0(a1,d2.w),d1     * Befehl 1 ? usw.
beq ausf                * Ja, dann ausfuehren
cmp.b #'^',0(a1,d2.w)   * endezeichen ?
beq endsteuer           * Ja dann war kein definiertes Steuerzeichen
addq #1,d2
bra stloop

endsteuer:
adda.l #1,a0            * auf Zeichen nach Steuercode zeigen
bra loop                * un Text weiter ausgeben

ausf:
lea codetab(pc),a1      * Adresse der Codetabelle in a1
move.b 0(a1,d2.w),d0    * Steuercode holen
move #22,d7 ; @lo       * und an Drucker geben
trap #1
adda.l #1,a0            * auf Zeichen nach Steuercode zeigen
bra loop                * und Text weiter ausgeben

beftab:                 * Hier stehen die Codeziffern der Steuerbefehle
dc.b '1'
dc.b '2'
dc.b '3'
dc.b '4'
dc.b '5'
dc.b '6'
dc.b '7'
dc.b '8'
dc.b '9'
dc.b '0'
dc.b '^'                * Endezeichen der Tabelle

codetab:                * hier stehen die Codes, die jeweils ausgegeben werden
dc.b $9                 * BackSpace      1
dc.b $9                 * Tabulator      2
dc.b $e                 * TAB set        3
dc.b $f                 * TAB clear      4
dc.b $10                * RePeat         5
dc.b $16                * HalfSpace      6
dc.b $00                * Null fuer nicht
dc.b $00                * benutzte codes
dc.b $00                * dto.
dc.b $00                * dto.
dc.b $00                * dto.

enda:
```

Kein Problem bereitet es mit dieser Editor-Erweiterung, Drucker-Steuerzeichen in den Text aufzunehmen

der Buchstabe eingegeben werden, der beim Sortieren gleichwertig mit Ä sein soll, nämlich A:

E 622 'A'

(Die einfachen Anführungszeichen muß man bei IBM-kompatiblen Tastaturen eingeben, indem man zuerst die zugehörige Taste und dann die Leertaste drückt – erst dann erscheint das Zeichen.) Nach dem gleichen Prinzip muß man auch für alle anderen Umlaute mit H zunächst die Adresse in der Tabelle errechnen und an dieser dann mit E den jeweils entsprechenden Großbuchstaben eingeben, und zwar nach folgender Tabelle:

Umlaut:	Code:	Eingeben:
Ä	8E	'A'
Ö	99	'O'
Ü	9A	'U'
ä	84	'A'
ü	81	'U'
ß	E1	'S'

Das ist zwar ein wenig umständlich, aber es lohnt sich. Das geänderte Programm wird anschließend so wieder auf die gleiche Diskette gespeichert:

W 100 0 186 9

Statt 186 wird bei Ihnen sicher eine andere Zahl nötig sein, nämlich der schon beim L-Befehl eingesetzte Sektor. Mit Q kann man dann aus DEBUG aussteigen und z. B. mit Hilfe von EDLIN oder auch mit der Anweisung von COPY CON- TEST.TXT erzeugen, in die man ein paar Zeilen eingibt, von denen einige mit Umlauten beginnen und deren letzte ein CTRL-Z enthält. Die Anweisung TYPE TEST.TXT | SORT zeigt dann sofort, ob alles richtig gemacht wurde: Ä muß bei A, Ö bei O und Ü bei U erscheinen.

Außer einem „deutschen" Sortierprogramm verfügen Sie jetzt aber vor allem auch über eine gehörige Portion Wissen über den Umgang mit einzelnen Disketten-Sektoren und -Clustern!

Simulations-Programm

Ein in Holland an der TU von Enschede entwickeltes und hierzulande von der Frankfurter Firma ICS für MS-DOS-Rechner erhältliches Programmpaket namens Tutsim gestattet die Simulation analoger dynamischer Systeme im Rechner. Dabei werden unterschiedliche Grafikkarten und Matrixdrucker zur Ausgabe der errechneten Diagramme unterstützt.

Die Eingabe analoger Abläufe erfolgt, wie von der Regelungstechnik gewohnt, in Form von Funktionsblöcken. Die wichtigsten davon sind in der *Tabelle* zusammengestellt. Wie ein fertiges Simulations-Modell aussieht, zeigt das *Bild*: Hier ist ein Ball realisiert, der auf den Boden fällt und dabei mehrmals wieder zurückprallt. Er gehorcht dabei folgender Differentialgleichung:

$$x'' = -g - (W*x'+K(x-R))/m \text{ für } x<R$$

Hierbei ist R der Ball-Radius, m seine Masse, g die Erdbeschleunigung und x die Höhe über dem Boden.
An diesem Beispiel wird wohl deutlich, daß ein gehöriges Maß an mathematischem und regelungstechnischem Wissen dazugehört, um selbst Modelle mit Tutsim realisieren zu können. Hinzu kommt, daß das derzeit nur in Englisch erhältliche Handbuch dieses Wissen einfach voraussetzt.
Das Programm selbst, ein EXE-File, bietet zahlreiche Help-Menüs an, wenn man einmal ein Kommando vergessen hat. Es arbeitet allerdings nicht bildschirm-, sondern zeilenorientiert, d. h. ohne Eingabemasken. Ferner ist es jederzeit möglich, das Programm durch CTRL-C bei Verlust aller eingegebener Daten zu verlassen – hier ist also Vorsicht geboten. Insgesamt bot sich der Eindruck, daß die Software weitgehend in typischer Hochschul-Manier geschrieben und dann von einem Großrechner auf den IBM-PC umcompiliert wurde. Fe.

```
GIVE COMMAND(H=HELP): L


MODEL-FILE: ball.sim
DATE:         9     28    1985

TUTSIM    ***IBM-PC***
MODEL LISTING

TIMING: 500.000E-06       2.5000
PLOTBLOCKS AND RANGES
      0     0.0000        2.5000
      5     0.0000        1.0000


MODEL:
   9.8100           1 CON
                    2 SUM       1       6
   0.0000           3 REL       1       2       2
                                9
   0.0000           4 INT      -3
   0.9000000        5 INT       4
   1.0000           6 ATT       7       8
  10.000E+03        7 GAI       9
  10.0000           8 GAI       4
                    9 SUM       5     -10
   0.1000000       10 CON
```

Simulationsmodell eines hüpfenden Balls in der Tutsim-Eingabesprache. Außer für MS-DOS gibt es das Programm auch für den Apple-II und den C-64

Einige Tutsim-Funktionsblöcke

ABS	Absolutwert
ADL	Algebraische Verzögerung
ATT	Abschwächer
CLK	Taktgenerator
DFF	Daten-Flipflop
DIF	Differentation
DIV	Divisions-Funktion
EUL	Euler-Integrator
EXP	Exponent
FNC	Funktions-Generator
GAI	Verstärker
GSQ	Quadrat der Leitfähigkeit
HLT	Haltfunktion
INT	Integrator
INV	Logischer Inverter
LIM	Begrenzer
LOG	Logarithmierer
MAX	Maximum-Funktion
MIN	Minimum-Funktion
MUL	Multiplizierer
NAN	NAND-Verknüpfung
NOI	Rauschgenerator
NOR	NOR-Verknüpfung
PID	PID-Regler
PLS	Puls-Funktion
REL	Relais-Block
SIN	Sinus-Funktion
SPL	Sample-and-Hold-Funktion
SQT	Quadratwurzel-Funktion
SUM	Summierer
TIM	Zeitfunktion
XOR	Exklusiv-Oder-Funktion