

PETER'S ASSEMBLERECKE

80 Zeichen per Software

Gleich zwei Themen sind diesmal in der Assemblerecke enthalten: Erstens die Ausgabe von 80 Zeichen pro Zeile in einem GRAPHICS-8 Screen und zweitens kommt die Mini-RAMDisk aus Heft 7/85 nochmal zum Zuge. Diesmal allerdings, auf vielfachen Leserwunsch, in einer für den Kassettenrekorder angepaßten Form.

Nebenbei bemerkt, feiert die Assemblerecke mit dieser Ausgabe ihren ersten Geburtstag. Und es gibt gleich zwei Geburtstagsgeschenke, ein gutes aber auch ein weniger erfreuliches. Fangen wir mit dem schlechteren an: Wie Sie sicherlich bereits bemerkt haben, erscheint die CK jetzt zweimonatlich, so daß es nur noch 6 Assemblerecken im Jahr geben wird. Dafür, und das ist die gute Nachricht, steht in Zukunft mehr Platz zur Verfügung. Daher gibt's diesmal auch gleich zwei Themen.

80 Zeichen

Bestimmt haben Sie schon eine Anzeige für das eine oder andere Textverarbeitungs-Programm gesehen, in der mit 80-Zeichen pro Zeile geworben wird. Genug Grund, um in der Assemblerecke zu zeigen, wie man so etwas machen kann.

Das Programm in Listing 1 kann neben den 80 Zeichen sogar noch verschiedene Schriftgrößen darstellen (s. Bild 1).

Aus eigener Erfahrung wissen Sie, daß Ihr Atari im normalen GRAPHICS-0 Modus nur 40 Zeichen in einer Zeile darstellen kann, also wo liegt der Trick? Nun ja, man muß statt dem gewohnten GR.0 Modus die am höchsten auflösende Grafikstufe 8 nehmen und die Zeichenmatrix (die Grafikpunkte, die das Zeichen darstellen) per Software in den Videospeicher eintragen. Wenn Sie ein eifriger Leser der Assemblerecke sind, dann kennen Sie das Prinzip noch von Heft 4/85, in dem nach gleichem Strickmuster Text und Grafik gemischt wurde.

Damals konnten wir aber auch nur 40 Zeichen in einer Zeile darstellen. Den Grund dafür kann man sich anhand einer einfachen Rechnung klar machen. GR.8 kann in horizontaler Richtung 320 Grafikpunkte (Pixels) darstellen. Ein Zeichen aus dem im ROM abgelegten Zeichensatz ist 8 Pixels breit, so daß $320/8=40$ Zeichen darstellbar sind. Für 80 Zeichen müssen wir zu einer List greifen: Wir machen jedes Zeichen

statt 8 nur noch 4 Pixels breit und können dadurch doppelt so viele Zeichen darstellen.

4 × 8 Matrix

Ein paar Haken hat die Sache natürlich: Statt 1 KByte für den 40-Zeichen-Schirm, braucht die 80-Zeichen-Darstellung immerhin 8 KByte (GR.8). Schlimmer ist aber, daß die kleinen Zeichen bei weitem nicht so gut zu lesen sind. Das hat zwei Gründe: In der 4×8 Matrix lassen sich die Buchstaben längst nicht so schön wie in einer 8×8 Matrix darstellen. Das liegt daran, daß zum nächsten Buchstaben ja noch Platz sein sollte, so daß im Grunde nur 3 Pixels in der Breite zur Verfügung stehen.

Der zweite Grund ist eine Eigenheit von GR.8, die dem einen oder anderen Leser sicherlich schon in einem anderen Zusammenhang aufgefallen ist. Wenn zwei Grafikpunkte waagrecht nebeneinander gesetzt werden, so sind diese wesentlich heller als ein einzelner Punkt. Im ROM-Zeichensatz ist dieses Problem recht einfach umgangen, da immer (prüfen Sie es doch mal nach) zwei Punkte nebeneinander gesetzt sind und so eine gleichmäßige Helligkeit erreicht wird. Bei den kleineren 4×8 Zeichen ist das verständlicherweise nicht denkbar, so daß die Zeichen unterschiedlich hell sind.

Die Darstellung läßt sich etwas verbessern, wenn man den Hintergrund hell und die Zeichenfarbe dunkel stellt. Aber mit Farben und Helligkeitsstufen können Sie im Listing 1 selbst ausgiebig experimentieren. Gegen die schlechter lesbare 4×8 Darstellung läßt sich aber nichts machen, damit muß man sich einfach abfinden.

Zur Anwendung

Wie immer in der Assemblerecke finden Sie das Maschinenprogramm sowohl als Basic-Loader wie auch als Assemblerlisting abgedruckt. Der Basic-Loader enthält zusätzlich ein Beispiel zur Anwendung. Sie können die Zeilen ab 30.000 jedoch in eigene Programme einbauen. Das Assemblerlisting müssen Sie nicht unbedingt eintippen, es ist nur als Hintergrundinformation für erfahrene Leser gedacht. Das Maschinen-

programm befindet sich in Page 6 und kann vom Basic aus mit einem USR-Befehl aufgerufen werden.

Dazu bedarf es aber einiger Vorarbeit. Um das Maschinenprogramm kurz zu halten, wurde zu einem Trick gegriffen: Bevor die USR-Routine mit der Arbeit beginnt, muß die Position des Textes am Bildschirm durch einen PLOT-Befehl markiert werden. Dabei werden nicht alle 320 möglichen horizontalen Positionen ausgewertet, sondern nur ganze Zeichenpositionen, d.h., es werden im 80-Zeichenmodus 80 und im 40-Zeichenmodus 40 Positionen berücksichtigt.

Vor dem PLOT-Befehl schalten Sie zweckmäßigerweise auf die Hintergrundfarbe um (COLOR 0), damit kein Punkt zurückbleibt. Der USR-Anweisung müssen Sie die Adresse eines auszugebenden Strings und dessen Länge mitgeben. Außerdem wird ein Attribut benötigt, mit dem man das Schriftbild beeinflussen kann. Folgende Werte sind zulässig:

0: 40-Zeichen, normale Schrift
64: 40-Zeichen, doppelt hoch
128: 80-Zeichen, normal hoch
192: 80-Zeichen, doppelt hoch

Sie sehen, damit läßt sich allerhand anfangen. Schreibt man eine Zeile mit mehreren USR-Befehlen (s. Demo), dann läßt sich die Schriftgröße auch innerhalb einer Zeile verändern. Hier ein Beispiel zur Ausgabe eines Textes.

```
COLOR 0: PLOT 120, 100
X = USR (1536, ADR
("HALLO"), 5, 128)
```

Diese Befehle schreiben das Wort "HALLO" im 80-Zeichenmodus auf den Schirm. Voraussetzung ist natürlich, daß das Maschinenprogramm im Speicher liegt und GR.8 aufgerufen ist.

Wie läuft's?

Wenden wir jetzt unsere Aufmerksamkeit dem Assemblerlisting zu. Zu Anfang werden die Parameter des USR-Befehls per PLA-Befehle vom Stack genommen und abgespeichert. Aus OLDADR (\$5E) wird die Adresse des Bytes entnommen, welches durch den letzten PLOT-Befehl angesprochen wurde. Dann wird mit Hilfe der

80-ZEICHENKONVERTER

für alle 8-Bit ATARI-Computer 400/800/500XL/600XL/130XE

ERZEUGT 80-ZEICHEN DISPLAY IN DER GRAFIKSTUFE 8

und zusätzlich recht flexibel in den Schriftarten
auch normale Schrift ist möglich

PETER'S ASSEMBLERECKE IN COMPUTER KONTAKT

Speicherzelle OLD COL (\$5B, \$5C) die momentane Schreibspalte ermittelt und ein Merker (FLAG80) für ungerade Spalten gesetzt. Danach wird Zeichen um Zeichen aus dem String geholt, vom ATASCII in den internen Bildschirmcode gewandelt und die Position des Bitmusters im Zeichensatz berechnet. Danach wird das Bitmuster Zeile für Zeile in den Grafikspeicher übertragen, wobei die einzelnen Punkte den eventuell bereits vorhandenen Bildschirminhalt überlagern.

Die 80-Zeichen werden durch das Unterprogramm WANDEL80 ermöglicht. Hier wird das 8-Bit breite Bitmuster des normalen ROM-Zeichensatzes durch den Einsatz zweier kurzer Tabellen auf 4 Bit kom-

primiert und je nach Inhalt von FLAG80 um 4 Bit verschoben. Schließlich müssen ja im 80-Zeichen-Modus pro Byte die Bitmuster zweier Zeichen eingetragen werden.

Die Wandelroutine wird nur aufgerufen, wenn Bit 7 des Attributes gesetzt ist, andernfalls wird im 80-Zeichenmodus ausgegeben. Ist zusätzlich Bit 6 gleich 1, dann wird jede Bitmusterzeile doppelt ausgegeben und man erhält doppelt hohe Zeichen, sowohl im 40- als auch im 80-Zeichenmodus.

Das Assemblerprogramm wurde diesmal wieder mit MAC/65 (Atari-Assembler geht auch!) geschrieben. Wie versprochen, wird abgewechselt, das nächste Listing ist dann für ATMAS-II.

Listing 1

```

10 REM *****
20 REM * AUSGABE VON 80-ZEICHEN
30 REM * Basic-Loader mit Demo
40 REM * P. Finzel 1986
50 REM *****
100 DIM A$(127):GOSUB 30000
110 GRAPHICS 8+16:SETCOLOR 2,8,10:SETC
    OLDR 1,0,0
120 COLOR 1:PLOT 5,5:DRAWTO 314,5:DRAW
    TO 314,186:DRAWTO 5,186:DRAWTO 5,5
130 PLOT 5,150:DRAWTO 314,150
140 RESTORE 1000:FOR I=0 TO 7
150 READ X,Y,ATT,A$
160 COLOR 0:PLOT X,Y:REM wichtig!
170 A=USR(1536,ADR(A$),LEN(A$),ATT)
180 NEXT I
190 GOTO 190
200 REM *
210 REM * und hier ein paar Texte
220 REM * als Demo...
230 REM *
1000 DATA 80,20,64,80-ZEICHENKONVERTER
1010 DATA 40,40,128,fuer alle 8-Bit AT
    ARI-Computer 400/800/600XL/800XL/130XE
1015 DATA 64,70,128,ERZEUGT 80-ZEICHEN
    DISPLAY IN DER GRAPHIKSTUFE 8
1020 DATA 60,100,192,und zusaetzlich
    recht flexibel in den Schriftarten
1030 DATA 16,120,0,auch normale Schrif
    t ist moeglich
1040 DATA 16,168,0,PETER'S
1050 DATA 80,160,64,ASSEMBLERECKE
1060 DATA 192,160,128,IN COMPUTER KONT
    AKT
30000 REM * 80-ZEICHEN ML-PROGRAMM
30010 S=0:RESTORE 30100

```

```

30020 FOR A=1536 TO 1746:READ D:POKE A
    ,D:S=S+D:NEXT A
30030 IF S<>25816 THEN ? "DATEN-FEHLER
    !":STOP
30090 RETURN
30100 DATA 104,104,133,209,104,133,208
    ,104,104,133,210,104,104,133
30110 DATA 214,169,0,133,211,133,205,1
    33,213,165,92,133,216,165,91
30120 DATA 102,216,106,102,216,106,133
    ,212,165,212,201,80,176,103
30130 DATA 106,102,213,165,94,133,203,
    165,95,133,204,160,0,132,207
30140 DATA 164,211,177,208,41,127,201,
    96,176,11,201,32,176,4,9,64
30150 DATA 208,3,56,233,32,10,10,38,20
    7,10,38,207,133,206,24,165,207
30160 DATA 109,244,2,133,207,162,8,160
    ,0,177,206,36,214,16,3,32,147
30170 DATA 6,32,191,6,36,214,80,3,32,1
    91,6,230,206,202,208,230,230
30180 DATA 212,36,214,16,6,36,213,48,4
    ,16,4,230,212,230,205,230,211
30190 DATA 198,210,208,147,96,133,215,
    169,0,133,216,160,3,185,183
30200 DATA 6,36,215,240,7,185,187,6,5,
    216,133,216,136,16,239,165,216
30210 DATA 36,213,48,4,10,10,10,10,96,
    1,4,16,32,1,2,4,8,72,164,205
30220 DATA 17,203,145,203,24,165,203,1
    05,40,133,203,144,2,230,204,104,96

```

Listing 2

```

;*****
; * 80-ZEICHEN MIT GRAPHICS 8 *
; * * *
; * Peter Finzel '86 *
; * * *
; * Assembler: MAC/65 *
;*****
;
; Betriebssystemadressen:
;
OLD COL = $5B X-Pos. des Cursors
OLDADR = $5E Cursoradresse
CHBAS = $02F4 Pagenr. des Zeichensatzes
;
; Belegung der Zeropage
;
; * = $CB
;
BILD * = **2 Zeiger in Hi-Res Bildschirm
XOFFSET * = **1 Abstand in Bytes von OLDADR
ZSATZ * = **2 Zeiger in Zeichensatz
STRING * = **2 Zeiger in Textstring
LAENGE * = **1 Laenge des Strings
SINDEX * = **1 Index in String
SPALTE * = **1 Schreibspalte
FLAG80 * = **1 80-Zeichen Ausgabe
ATTBUT * = **1 Doppelt hohe Zeichen
ZALT * = **1 Zwischenspeicher
HILF * = **1 Hilfsregister
;
;*****
; 80-Zeichen in Graphics 8
; Aufruf: X=USR(1536,Adresse,Laenge,Att)
; Attribut: 0: normal, 40 Zeichen
; 64: doppelt hoch, 40 Zeichen
; 128: normal, 80 Zeichen
; 192: doppelt hoch, 80 Zeichen
;*****

```

```

;
;      *= $0600    1st in PAGE 6
;
PLA      ;# der Args
PLA      Stringadresse
STA STRING+1
PLA      LSB
STA STRING
PLA      MSB der Laenge
PLA      Laenge
STA LAENGE (max. 255 Zeichen!)
PLA
PLA      Attribut
STA ATTBUT ablegen
;
; Vorbesetzungen, Schreibspalte berechnen
;
LDA #0      Index in String
STA SINDEK  zuruecksetzen
STA XOFFSET
STA FLAG80
LDA OLDCOL+1 Position des
STA HILF    Graphik-Cursors
LDA OLDCOL  durch vier teilen
ROR HILF    ergibt die Scheib-
ROR A       spalte
ROR HILF
ROR A
STA SPALTE
;
; Schleife zur Ausgabe beginnt
;
WEITER    LDA SPALTE  Zeilenende?
CMP #80    dann nichts mehr
BCS WBOENDE  ausgeben! -->
ROR A      Flag fuer ungerade
ROR FLAG80 Spaltenzahl (Trick!)
LDA OLDDADR Cursoradresse des
STA BILD    letzten PLOT-Punktes
LDA OLDDADR+1 uebernehmen
STA BILD+1
;
; Zeichencode aus String, ATASCII in
; internen Code umrechnen
;
LDY #0
STY ZSATZ+1 Vorbesetzung
LDY SINDEK
LDA (STRING),Y ASCII-Zeichen im Akku
AND #$7F    Inversbit maskieren
CMP #96     jetzt Umwandlung in Bildschirmcode
BCS W_ENDE  Code stimmt -->
CMP #32     ist Graphikzeichen?
BCS ALPHA  nein, ist Alphazeichen -->
ORA #64     plus 64
BNE W_ENDE  Umwandlung fertig -->
ALPHA     SEC
SBC #32     Korrektur Buchstaben und Zahlen
;
; Adresse des Bitmusters im Zeichensatz
;
W_ENDE    ASL A      Bildschirmcode mal
ASL A      acht ist Zeiger in
ROL ZSATZ+1 den Zeichensatz
ASL A
ROL ZSATZ+1
STA ZSATZ  LSB ist fertig
CLC
LDA ZSATZ+1
ADC CHBAS  Basisadresse des
STA ZSATZ+1 Zeichensatzes dazu
;
; Schleife zur Ausgabe eines Zeichens
;
ZEILE     LDX #8      acht Zeilen/Zeichen
LDY #0
LDA (ZSATZ),Y eine Zeile des Zeichens
BIT ATTBUT 40er Modus?
BPL M2     ja-->
JSR WNDEL80 in 80er umwandeln
M2        JSR DATOUT  ausgeben
BIT ATTBUT doppelt hoch?
BVC M3     nein-->
JSR DATOUT
M3        INC ZSATZ    naechstes Zeichenmuster
DEX
BNE ZEILE  naechst. Zeile
;
;
INC SPALTE naechste Spalte
BIT ATTBUT 40-Zeichen-Modus?
BPL M4     ja-->
BIT FLAG80 ungerade Adresse?
BMI M5     ja, dann Index weiter
BPL M6     nein, Index bleibt!
M4        INC SPALTE  fuer 40er Modus 2 Spalten
M5        INC XOFFSET VideoRam weiter
M6        INC SINDEK  Zeiger in String
DEC LAENGE noch Zeichen da?
BNE WEITER jawohl -->
WBOENDE   RTS
;
; *****
; 40er Zeichensatz in 80er umrechnen
; *****
;
WNDEL80   STA ZALT    40er Zeichenmuster
LDA #0     neues Muster
STA HILF   vorbesetzen
LDY #3     vier Bit bearbeiten
;
SPALTE80  LDA BITTAB,Y Spalten-Maske
BIT ZALT   Bit gesetzt?
BEQ WBONIX nein -->
LDA UBOTAB,Y Umrechentabelle
ORA HILF
STA HILF
WBONIX    DEY         ;schon alle Spalten?
BPL SPALTE80 nein-->
LDA HILF    eben berechnetes Muster
BIT FLAG80
BMI UNGER   ungerade X-Position-->
ASL A       sonst in ungerade
ASL A       Position shiften
ASL A
ASL A
UNGER     RTS
;
; Tabellen zur Umwandlung in 80er Z-Satz
;
BITTAB    .BYTE $01,$04,$10,$20
UBOTAB    .BYTE $01,$02,$04,$08
;
; *****
; Bild-Daten in Screen eintragen
; Zeiger auf naechste Graphikzeile
; *****
DATOUT     PHA         ;Akku merken...
LDY XOFFSET
ORA (BILD),Y mit vorh. Bild verknuepfen
STA (BILD),Y und eintragen
CLC
LDA BILD    naechste Zeile ist
ADC #40     40 Bytes weiter
STA BILD
BCC DAT1    MSB nicht vergessen
INC BILD+1
DAT1       PLA
RTS         Akku restaurieren

```