

PETER'S ASSEMBLERECKE

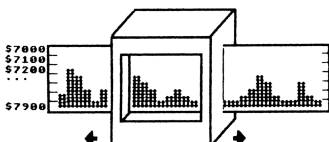
Scrolling Teil 2

In der Assemblerecke dieses Monats dreht sich wieder alles um Scrolling. Nachdem wir die vertikale Bildverschiebung in der Juni-Ausgabe bereits ausführlich besprochen haben, geht es jetzt um horizontales Scrolling.

RAM und BILD

Beim vertikalen Scrolling konnte die Bildinformation wie gewohnt abgelegt werden, d.h. für einen GRAPHICS 2 Bildschirm 20 Bytes für die erste Zeile, anschließend wieder 20 Bytes für die nächste Zeile usw. Die Erweiterung des Bildschirm-RAMs kam einfach dadurch zustande, daß nicht nur 10 (bzw. 12) Zeilen sondern mehrere hundert solcher Zeilen vorgesehen waren. Die vertikale Grobverschiebung wurde durch Veränderung der Bildschirmadresse um Vielfache der Zeilenlänge erzeugt. Man kann sich unschwer vorstellen, daß mit dieser Anordnung der Bildinformation horizontales Scrolling nicht möglich ist. Erhöht man nämlich die Anfangsadresse des Bildschirms um eins, rutscht die erste Zeile zwar nach links, aber das erste Zeichen der zweiten Zeile taucht dann in der letzten Position der ersten Zeile auf.

Sie sehen, die Anordnung der Bildschirminformation ist beim horizontalen Scrolling von entscheidender Bedeutung. Wie in Bild 1 dargestellt, muß



die Information gleich in langen Zeilen angeordnet werden, um dann später den Bildschirm gleich einem Fenster darüber schieben zu können. Bei einer Zeilenlänge von 256 Zeichen wird ab der Anfangsadresse des Bildschirmspeichers die gesamte erste, 256 Bytes umfassende Zeile abgelegt. Jetzt folgt

die ebenfalls 256 Bytes lange zweite Zeile, das heißt wir tun so, als ob der Atari einen Grafikmodus hätte, der 256 Zeichen in einer Zeile darstellen kann.

ANTIC schafft's

Jetzt wird das nächste Problem offensichtlich: Der darzustellende Bildschirmspeicher ist nicht zusammenhängend. Nehmen wir an, das Fenster befindet sich am linken Ende der Zeilen. Die erste Zeile beginnt bei der Adresse \$7000 und endet, da ANTIC im Modus 7 (GRAPHICS 2) nur 20 Zeichen pro Zeile darstellen kann, an der Adresse \$7013. Die zweite Zeile beginnt aber erst bei Adresse \$7100, so daß ein "Loch" von 236 Bytes entstanden ist. Hier zeigt sich, daß ANTIC im wahren Sinne des Wortes "programmierbar" ist: Wir können ein ANTIC-Programm (eine Display-List) schreiben, die jeder einzelnen Bildschirmzeile einen separaten Bildschirmspeicher zuordnet. Dies wird durch Setzen des sogenannten LMS (Load Memory Scan) Bits im ANTIC-Befehl erreicht. Eine ANTIC-Anweisung besteht dann aus dem Befehlscode mit nachfolgender Adresse des gewünschten Bildschirmspeichers im Low/High Format.

Nehmen wir wiederum an, das Fenster befindet sich am linken Ende des darzustellenden Bereiches. Soll das Fenster jetzt um eine Zeichenposition nach rechts gescrollt werden, so muß die Anfangsadresse des Videospeichers jeder Zeile um eins erhöht werden, wobei das zur Vermeidung von Verzerrungen

am Schirm in allen 10 Zeilen möglichst gleichzeitig geschehen sollte. Wir wissen aber alle, daß ein Computer nichts gleichzeitig erledigen kann, sondern alles schön der Reihe nach bearbeitet. Aus diesem Grunde sind zwei Maßnahmen notwendig: So muß die Änderung der Adressen im Rahmen eines schnellen Maschinenprogrammes erfolgen, und außerdem sollte dieses nur in der vertikalen Austastlücke (im VBI) benutzt werden, so daß Grafikstörungen ausgeschlossen sind.

Feinscrolling

Wie im ersten Teil dieses Artikels soll auch beim horizontalen Scrolling die Feinverschiebung benutzt werden. Wird das Bit 4 (das HSCOL-Bit) jeder betroffenen ANTIC-Anweisung auf eins gesetzt, so kann diese Zeile durch das HSCROL-Register (Adresse \$D404) bis zu 15 Pixel nach rechts verschoben werden. Für längere Scrollwege muß daher wiederum Grob mit Feinscrolling kombiniert werden, und dadurch wird's kompliziert: Wie schon festgestellt, entspricht eine Erhöhung der Zeilenadressen einer Verschiebung des Bildschirmfensters nach rechts. Da aber das Fenster (Ihr Fernsehgerät) im Regelfall ortsfest ist, entspricht dies einer Bewegung des Bildes nach links. Eine Erhöhung des Feinverschiebungsregisters HSCROL bewirkt dagegen eine Verschiebung des Bildes nach rechts, woraus zu folgern ist, daß Grob- und Feinscrolling immer konträr verwendet werden müssen. Eine Verschiebung nach links erfordert eine Verminderung des HSCROL-Registers bei Vergrößerung der Zeilenadressen. Alles klar?

Das Assemblerlisting

Ziel des Assemblerprogrammes soll es sein, zehn GRAPHICS 2 Zeilen über eine Länge von 256 Zeichen horizontal zu scrollen. Der Aufbau des Assemblerlistings entspricht im wesentlichen dem Programm des ersten Teils. Dabei wird wieder eine handgestrickte Display-List und der "deferred" VBI verwendet. Den Anfang bilden einige Defi-

nitionen. Unter anderem wird der Bildschirmspeicher beginnend ab der Adresse \$7000 definiert. Das Programm selbst liegt in Page 6 und wird mit X=USR (1536) aufgerufen. Jetzt folgt die Display-List, die, wie bereits geschildert, aus ANTIC-Anweisungen mit gesetzten LMS- und HSCROL-Bits und den Zeilenadressen besteht. Zwei Variablen halten die momentane Scroll-Position fest:

SHSCR dient als Schattenregister des HSCROL-Registers (das selbst nur geschrieben aber nicht gelesen werden kann). GRBSCR speichert die Entfernung des Fensters vom Zeilenanfang in Zeichen und ist daher für das Grobscrolling verantwortlich.

VSINIT initialisiert die Display-List und das HSCROL-Register so, daß das Fenster ganz links zu liegen kommt. Die Display-List wird aktiviert und die Scroll-Routine HSVBI in den "deferred" VBI eingebunden. Das VBI-Programm testet, ob Joystick Null nach links oder rechts bewegt wurde und ruft dementsprechend die Unterprogramme LINKS und RECHTS auf. Nach getaner Arbeit wird noch das Schattenregister SHSCR ins Hardwareregister HSCROL kopiert und der VBI über XITVBV verlassen.

Im Unterprogramm LINKS wird zuerst abgefragt, ob der Rand des Scrollbereiches erreicht ist, der mit GRBMAX gleich 233 die Zeilenlänge minus der Bildschirmlänge ist. Nun wird das Schattenregister der Feinposition vermindert. Falls Feinscrolling nicht ausreicht, wird die Grobposition um eins erhöht. Die Zeilenadressen selbst werden durch das Unterprogramm SETLMS erzeugt, was sich durch die gewählte Zeilenlänge von 256 Bytes recht einfach gestaltet. SETLMS braucht nur die Grobposition aus GRBSCR zu entnehmen und in den niederwertigen Teil der Zeilenadresse einzutragen. Bei der Verwendung von anderen Zeilenlängen muß dieser Programmteil verändert bzw. erweitert werden. Es ist meist zweckmäßig, eine Tabelle

der Zeilenanfänge anzulegen, dazu jeweils den Offset des Fensters zu addieren und den so errechneten Wert als LMS-Adresse in die Display-List einzutragen.

Das Unterprogramm RECHTS arbeitet nach dem gleichen Prinzip, nur wird eben in die andere Richtung gescrollt. Ein Demo des Programmes können Sie dem Listing 2 entnehmen. Hier wird eine Zufalls-Landschaft entworfen, die mit einem Joystick in Port 1 horizontal gescrollt werden kann. Wenn Sie das Programm nach eigenen Vorstellungen modifizieren wollen, sollten sie jedoch das mit MAC/65 (die Editor-Assembler Cartridge tut's auch) verfaßte Assemblerlisting eintippen.

Ein paar Kniffe

Durch das Einschalten der horizontalen Feinverschiebung verändert ANTIC selbsttätig die Anzahl der gelesenen Bytes pro Zeile. Wenn Sie mit dem GRAPHICS 2 Modus arbeiten, bei dem ANTIC normalerweise 20 Bytes pro Zeile holt, werden beim Einschalten des Feinscrollings 24 Bytes gelesen (aber nicht dargestellt). Das ist ja auch bitter nötig, wenn das erste Zeichen nur noch halb sichtbar ist, muß ANTIC ja wissen, welches Zeichen an der 21. Position dargestellt werden soll. Dieses Verhalten führt aber auch dazu, daß die ersten beiden Spalten außerhalb der

Sichtweite bleiben. Einen gewissen Ausgleich bietet hier der "breite" Bildschirmmodus (POKE 559,35), der Scrolling in Cinema-Scope bietet. Hier zeigt sich wieder einmal, daß der Atari in Grafikangelegenheiten der Rolls-Royce unter den Home-Computern ist. Andere Computer (welche wohl?) lösen das Problem auf eine recht simple Art: Der Bildschirm wird um zwei Spalten eingeeengt, so daß beim horizontalen Scrolling genügend Bildschirminformation vorhanden ist.

Ein zweites Problem tritt an 4KByte-Grenzen auf: ANTICs interner Adresszähler ist nur 12 Bit breit und kann daher 4K-Grenzen nicht überspringen. Sollte jedoch ein Videospeicher von mehr als 4KByte Länge erforderlich sein, kann man sich recht gut helfen, indem man als Zeilenlänge eine Zweierpotenz (64, 128 oder wie im Beispiel 256) wählt. Falls das nicht möglich ist, kann man die Zeile, in der die 4K-Grenze auftritt, durch geschickte Wahl der Zeilenanfänge einfach überspringen. Der Videospeicher hat dann zwar "Löcher", aber dieses Problem kann man mit Tabellen ganz gut in den Griff bekommen. Die einzige tatsächliche Beschränkung besagt, daß eine zusammenhängende Zeile nicht länger als 4096 Zeichen sein kann. Und das genügt doch, oder?

Peter Finzel

Listing 1

```

0100 ;*****
0110 ;LISTING 1:
0120 ;
0130 ;Horizontales Fein-Scrolling
0140 ;      in Maschinensprache
0150 ;
0160 ;      Peter Finzel '85
0170 ;*****
0180 ;
0190 ; Konstante
0200 ;
0210 BASIS = $7000
0220 HSMAX = 7
0230 ZEILE = 256
0240 ZZAHL = 10
0250 GRBMAX = 233
0260 MOD = 7+$50
0270 ;
0280 ; Operating System
0290 ;
0300 SDLISTL = $0230 Schattenreg. Display-Listadresse
0310 STICKO = $0278 Schattenreg. f. Joystick Nr. 0
0320 SETVBV = $E45C Routine f. Interruptvektoren
0330 XITVBV = $E462 Abschluss des VBI
0340 ;
0350 ; Hardware
0360 ;
=D404 0370 HSCROL = $D404 Register f. hor. Verschiebung
0380 ;
0390 ;*****
0400 ;Programm-Einsprung
0410 ;*****
0420 ;
0430 ; * = $0600 wie immer in Page 6
0440 ;
0600 68 0450 PLA BASIC Einsprung X=USR(1536)

```

```

0601 4C2A06 0460 JMP HSNIT Sprung zum MP-Anfang
0470 ;
0480 ;*****
0490 ;Display-List fuer horizontal
0500 ;scrollendes 'GRAPHICS 2'-Display
0510 ;10 Zeilen ANTIC-Modus 7
0520 ;*****
0530 ;
0604 707070 0540 DLIST .BYTE $70,$70,$70 ;3 Leerzeilen
0607 57 0550 .BYTE MOD ;ANTIC-Befehl
0608 0070 0560 LMSADR .WORD BASIS ;Adresse Bildspeicher
060A 57 0570 .BYTE MOD
060B 0071 0580 .WORD BASIS+ZEILE
060D 57 0590 .BYTE MOD
060E 0072 0600 .WORD BASIS+ZEILE*2
0610 57 0610 .BYTE MOD
0611 0073 0620 .WORD BASIS+ZEILE*3
0613 57 0630 .BYTE MOD
0614 0074 0640 .WORD BASIS+ZEILE*4
0616 57 0650 .BYTE MOD
0617 0075 0660 .WORD BASIS+ZEILE*5
0619 57 0670 .BYTE MOD
061A 0076 0680 .WORD BASIS+ZEILE*6
061C 57 0690 .BYTE MOD
061D 0077 0700 .WORD BASIS+ZEILE*7
061F 57 0710 .BYTE MOD
0620 0078 0720 .WORD BASIS+ZEILE*8
0622 57 0730 .BYTE MOD
0623 0079 0740 .WORD BASIS+ZEILE*9
0625 41 0750 .BYTE $41 ;ANTIC-JMP-Befehl
0626 0406 0760 .WORD DLIST ;zum Anfang der Disp.-List
0770 ;
0780 ;*****
0790 ; Interne Variable
0800 ;*****
0810 ;
0628 00 0820 SHSCR .BYTE 0 ;Schattenregister HSCROL
0629 00 0830 GRBSCR .BYTE 0 ;Zaehler fuer Grobscrolling
0840 ;
0850 ;*****
0860 ;Initialisierungsroutine zum
0870 ;Einrichten der neuen Disp.-List
0880 ;und der VBI-Routine
0890 ;*****
0900 ;
0910 HSNIT LDA #0 Grobscrolling
0920 STA GRBSCR auf Null
0930 JSR SETLMS und Displaylist vorbereiten
0940 LDA #DLIST&255 neue Disp.-List
0950 STA SDLISTL einschalten
0960 LDA #DLIST/256
0970 STA SDLISTL+1
0980 LDA #DLIST Fein-Scrolling in Ausgangs-
0990 STA HSCROL position setzen
1000 STA SHSCR auch Schattenregister!
1010 LDY #HSVBI&255 Scrolli-Routine in
1020 LDX #HSVBI/256 den VBI einfuegen
1030 LDA #7 deferred VBI genuegt
1040 JSR SETVBV ohne Kommentar...
1050 RTS
1060 ;
1070 ;*****
1080 ;VBI-Routine f. hor. Scrolling
1090 ;*****
1100 ;
1110 HSVBI CLD ;$502 rechnet binar
1120 LDA STICKO Joystick 0
1130 AND #8 nach rechts?
1140 BNE TLINKS nein, versuche links-->
1150 ;
1160 JSR RECHTS Bildschirm nach rechts
1170 JMP VBIEND Fertig!==>
1180 ;
1190 TLINKS LDA STICKO
1200 AND #4 nach links gezogen?
1210 BNE VBIEND nein, kein Scrolling!
1220 ;
1230 JSR LINKS nach links scrollen>>>
1240 ;
0666 AD2806 1250 VBIEND LDA SHSCR Schattenregister
Fein-Scrolling
0669 BD04D4 1260 STA HSCROL in Hardwarereg. uebertragen
066C 4C62E4 1270 JMP XITVBV Schluss fuer heute!
1280 ;
1290 ;*****
1300 ; UP Bildschirm nach links
1310 ;*****
1320 ;
066F AD2806 1330 LINKS LDA SHSCR Zeichengrenze erreicht?
0672 D007 1340 BNE LNK1 nein -->
1350 ;
0674 AD2906 1360 LDA GRBSCR Weiteres Scrolling
0677 C9E9 1370 CMP #GRBMAX nach links moeglich?
0679 B013 1380 BCS LNKEND nein, kein Scrolling ->
067B CE2806 1390 LNK1 DEC SHSCR Ein Pixel nach links
067E AD2806 1400 LDA SHSCR reicht Feinscrolling?
0681 100B 1410 BPL LNKEND jawohl, dann fertig
1420 ;
0683 A907 1430 LDA #HSMAX Feinverschiebung zurueck
0685 BD2806 1440 STA SHSCR
0688 EE2906 1450 INC GRBSCR Grobposition weiter
068B 20AF06 1460 JSR SETLMS in Displaylist eintragen
068E 60 1470 LNKEND RTS
1480 ;
1490 ;*****
1500 ; UP Bildschirm nach rechts
1510 ;*****
1520 ;
068F AD2806 1530 RECHTS LDA SHSCR Zeichengrenze erreicht?
0692 D005 1540 BNE RHT1 nein -->
1550 ;
0694 AD2906 1560 LDA GRBSCR Rand erreicht?
0697 F015 1570 BEQ RHTEND ja, kein Scrolling ->
0699 EE2806 1580 RHT1 INC SHSCR ein Pixel rechts
069C AD2806 1590 LDA SHSCR Feinscrolling ausreichend?
069F C90B 1600 CMP #HSMAX+1

```

```

06A1 900B      1610      BCC RHTEND
                1620 ;
06A3 A900      1630      LDA #0      Feinscrolling
06A5 BD2806    1640      STA SHSCR    ruecksetzen
06A8 CE2906    1650      DEC GRBSCR    Grob-Pos. zurueck
06AB 20AF06    1660      JSR SETLMS   in Disp.-List eintragen
06AE 60        1670      RHTEND RTS
                1680 ;
                1690 ;*****
                1700 ; UP Eintragen der LMS-Adressen
                1710 ;*****
                1720 ;
06AF A200      1730      SETLMS LDX #0  Index in Disp.-List
06B1 AD2906    1740      LDA GRBSCR    Grobscroll-Position
06B4 9D0806    1750      NXTLMS STA LMSADR,X im zweiten Byte
06B7 EB        1760      INX          ; jeder ANTIC-Anweisung
06B8 EB        1770      INX          ; eintragen
06B9 EB        1780      INX
06BA E01E      1790      CPX #ZZAHL*3 alle Zeilen bearbeitet?
06BC 90F6      1800      BCC NXTLMS  noch nicht ---
06BE 60        1810      RTS

```

Listing 2

```

100 REM * DEMO-Horizontales Scrolling
110 REM * Peter Finzel 1985
120 REM *
130 GOSUB 1000:REM * Maschinenpgm.
200 A=USR(1536):REM * Scrolling ein
300 REM LANDSCHAFT ENTWERFEN
310 BASIS=28672
320 FOR I=0 TO 255
330 A=INT(RND(0)*8)+1
340 FOR S=0 TO A:POKE BASIS+S*256+I,0:
NEXT S
350 FOR S=A+1 TO 8:POKE BASIS+S*256+I,
3:NEXT S
360 POKE BASIS+9*256+I,67
390 NEXT I
400 GOTO 400

```

```

1000 REM * Maschinenprogramm...
1010 S=0:RESTORE 1100
1020 FOR A=1536 TO 1726:READ D:POKE A,
D:S=S+D:NEXT A
1030 IF S<>15960 THEN ? "DATEN-FEHLER!
":STOP
1090 RETURN
1100 DATA 104,76,42,6,112,112,112,87,0
,112,87,0,113,87,0,114,87,0
1110 DATA 115,87,0,116,87,0,117,87,0,1
18,87,0,119,87,0,120,87,0,121
1120 DATA 65,4,6,0,0,169,0,141,41,6,32
,175,6,169,4,141,48,2,169,6
1130 DATA 141,49,2,169,0,141,4,212,141
,40,6,160,78,162,6,169,7,32
1140 DATA 92,228,96,216,173,120,2,41,8
,208,6,32,143,6,76,102,6,173
1150 DATA 120,2,41,4,208,3,32,111,6,17
3,40,6,141,4,212,76,98,228,173
1160 DATA 40,6,208,7,173,41,6,201,233,
176,19,206,40,6,173,40,6,16
1170 DATA 11,169,7,141,40,6,238,41,6,3
2,175,6,96,173,40,6,208,5,173
1180 DATA 41,6,240,21,238,40,6,173,40,
6,201,8,144,11,169,0,141,40
1190 DATA 6,206,41,6,32,175,6,96,162,0
,173,41,6,157,8,6,232,232,232
1200 DATA 224,30,144,246,96

```