PETER'S ASSEMBLERECKE

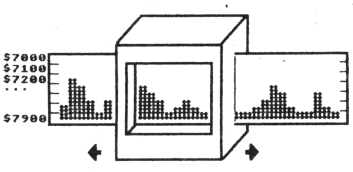
Scrolling Teil 2

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 befinde 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 wahrsten 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 befinde 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 Definitionen. 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