Sollte es tatsächlich noch Leser geben, die mit dem Begriff Player-Missile-Grafik noch nichts anfangen können? Das ist eigentlich kaum zu glauben. Aber wissen Sie auch, dass man damit noch viel mehr anfangen kann, als gemeinhin bekannt ist? So lassen sich z.B. vier Player in je sieben verschiedenen Farben darstellen. Auch ist es möglich, Figuren abzubilden, ohne den regulären PM-Speicher zu benutzen. Wenn dies für Sie neu ist, sollten Sie jetzt unbedingt weiterlesen!

Normalerweise gehen Sie beim Einschalten der PM-Grafik so vor: Ein Speicherbereich von 1 bzw. 2 KByte wird reserviert und per Programm gelöscht. Über das Register PMBASE erfolgt die Mitteilung an den ANTIC-Chip, wo sich dieser Bereich im Speicher befindet. Nun wird die DMA ein- (Register SDMCTL) und die Darstellung der Player mit dem GTIA-Register GRACTL zugeschaltet. Die vertikale Position eines Objektes ist davon abhängig, an welcher Stelle sein Bit-Muster in den PM-Speicher geschrieben wird. Dagegen ist die horizontale Lage nur durch das zugehörige HPOS-Register festgelegt.

Die PM-Grafik ist ein Produkt der Zusammenarbeit zweier Chips im Rechner: Der eine, ANTIC, hat nur die Aufgabe, die für die Bildaufbereitung nötigen Daten aus dem Speicher zu holen. Der zweite, GTIA, übernimmt diese Daten und erzeugt daraus das Videosignal.

Es geht aber auch anders. So ist es möglich, PM-Grafik nur mit Hilfe von GTIA zu erstellen, ganz ohne Zutun der ANTIC-DMA. Es gibt nämlich fünf Register, mit denen man die Form jedes Players und der Missiles festlegen kann. GTIA erzeugt diese Form (genauer gesagt, das Bit-Muster) so lange am Bildschirm, bis ein neues Muster in dieses Register geschrieben wird. Die horizontale Position, die Farbe und die Breite des Musters lassen sich durch die bekannten Register verändern.

Wenn man daher einmal einen Wert, sagen wir $FF, in das Formregister schreibt, wird ein farbiger Balken erzeugt, der vom oberen bis zum unteren Rand des Bildschirms reicht. Dieser Balken lässt sich mit den HPOS-Registern sehr leicht verschieben und deshalb auch in Basic effektvoll einsetzen.

In Listing 1 finden alle vier Player auf diese Art Verwendung. Sie werden auf vierfache Breite geschaltet und mittels des Prioritätsregisters "vor" den Bildschirm gelegt. Nun folgt der Ausdruck eines Textes, der wegen der Überdeckung noch nicht sichtbar ist. Fährt man jetzt die Player nach links und rechts weg, erhält man den Effekt eines sich öffnenden Vorhangs. Das könnte sich bestimmt gut für den Titelvorspann Ihres nächsten Programms eignen.

Gehen wir noch einen Schritt weiter. Wenn man die ANTIC-DMA benutzt, kommt die Form der Player zustande, indem pro Bildschirmzeile und pro Player ein Byte aus dem PM-Speicher gelesen und (natürlich auf internem Weg) in das Formregister von GTIA geschrieben wird. Diese Aufgabe kann aber auch der 6502 übernehmen!

Wenn man die Player durch "Handarbeit" des 6502 erzeugt, ist es zusätzlich möglich, ihre Farbe oder Breite für jede Zeile neu festzulegen. Im Klartext heißt das, daß man mit dieser Technik vier mehrfarbige Player erstellen kann. Im Programm nach Listing 2 werden vier siebenfarbige erzeugt und in Bewegung gehalten. Unmöglich? Schauen Sie es sich selbst an.

Timing

Bevor Sie jedoch in zu großen Jubel ausbrechen, sollten Sie sich vor Augen halten, mit welchen Nachteilen diese Farbenpracht erkauft ist. Der 6502 wurde zum Sklaven der Bilderzeugung; er lässt sich nur noch in ca. 5000 bis 6000 Maschinenzyklen zum Rechnen benutzen. Die Player können nicht mehr frei über den Bildschirm bewegt werden, da z.B. die Farb- und Forminformationen für Player 4 erst ab etwa der horizontalen Mitte des Bildschirms zur Verfügung stehen.

Das ist einfach eine Begrenzung durch die maximale Geschwindigkeit des 6502. In der Zeit, die er braucht, um vier Form- und vier Farbregister zu ändern, ist der Elektronenstrahl schon in der Mitte des Bildschirms angelangt. Außerdem - und das ist das Schlimmste - darf die Hintergrund-DMA nicht mehr verwendet werden, da sie das ganze Timing durcheinanderbringt.

Trotz dieser enormen Beschränkungen ist es interessant, sich einmal mit diesen Techniken, in der Fachsprache Kernel genannt, auseinanderzusetzen. Der Trick dabei ist, dass der 6502 mit Hilfe des Registers VCOUNT feststellen kann, welche Bildzeile gerade erzeugt wird. Er kann diese Zahl mit den gewünschten vertikalen Positionen (VPOS in Listing 2) der Player vergleichen und daraus ermitteln, zu welchen Zeitpunkten er Form und Farbe in die Hardware-Register schreiben muss. Die Werte für Form und Farbe werden zweckmäßigerweise für jeden Player in kurzen Tabellen festgehalten (STAB und CTAB in Listing 1).

Wie schon erwähnt, ist die ganze Sache ein Zeitproblem. Der Vergleich der Vertikalpositionen mit VCOUNT und das Laden der Form- und Farbwerte lässt sich schon nicht mehr in einer Bildschirmzeile bewältigen. Daher arbeitet das Programm auch nur mit zweizeiliger Auflösung. Jeweils in der ersten Zeile werden die Vergleiche von VCOUNT mit den Vertikalpositionen ausgeführt und die entsprechenden Werte aus den Tabellen geladen.

Nach einer horizontalen Synchronisation durch einen Schreibbefehl auf WSYNC folgt die zweite Phase, in der die zuvor ermittelten Werte möglichst schnell in die HardwareRegister übertragen werden. Im Beispiel geschieht dies, indem die erste Phase die festgestellten Werte in den Operanden der LDA -Befehle der zweiten Phase einträgt (selbstverändernder Code). Der ganze Vorgang läuft in einer Schleife ab, die im Vertical-Blank-Interrupt gestartet und beim Erreichen eines bestimmten VCOUNT-Werts (im Beispiel 120) beendet wird.

Sie sehen, die Kernel-Programmiertechnik hat ihre engen Grenzen und ist auch nicht ganz einfach zu beherrschen. Auf der anderen Seite lässt sich damit aber Erstaunliches aus dem Atari herausholen. Es wäre sogar denkbar, das Horizontalregister eines Players innerhalb einer Bildzeile zu verändern und diesen somit zweimal in einer Zeile (!) zu verwenden. Solche sogenannten horizontalen Kernels bedeuten allerdings ein enormes Timing-Problem und sind nur in den seltensten Fällen ratsam.

Sinnvoll dagegen wäre z.B. eine Kombination von ANTIC-DMA mit der Kernel-Idee, indem man die DMA zur Erzeugung der Form, den Kernel dagegen nur zum Verändern der Farben einsetzt. Warum versuchen Sie es nicht einmal?

Peter Finzel

```
100 REM * EFFEKT MIT
110 REM * DIREKTPROGRAMMIERTEN
120 REM * PM-GRAPHIK
130 REM * P.FINZEL '87
200 HPOSP0=53248:SIZEP0=53256
210 GRAFP0=53261:PCOLR0=704
220 GRACTL=53277:GPRIOR=623
250 GRAPHICS 2+16:SETCOLOR 4,0,0
260 SETCOLOR 0,0,14
280 POKE GPRIOR,1
300 REM * PLAYERS VORBEREITEN
310 FOR I=0 TO 3
320 POKE SIZEP0+I,3
330 POKE GRAFP0+I,255
340 POKE PCOLR0+I,0
350 NEXT I
400 POKE HPOSP0,64
410 POKE HPOSP0+1,96
420 POKE HPOSP0+2,128
430 POKE HPOSP0+3,160
440 POSITION 5,4:PRINT #6;"COMPUTER-"
450 POSITION 5,5:PRINT #6;"KONTAKT"
460 POSITION 6,7
470 PRINT #6;"IST SPITZE"
500 FOR I=0 TO 64
510 POKE HPOSP0+1,96-I
520 POKE HPOSP0,64-I
530 POKE HPOSP0+2,128+I
540 POKE HPOSP0+3,160+I
550 FOR J=0 TO 50:NEXT J
560 NEXT I
600 FOR I=64 TO 0 STEP -1
610 POKE HPOSP0,64-I
620 POKE HPOSP0+1,96-I
630 POKE HPOSP0+3,160+I
640 POKE HPOSP0+2,128+I
650 FOR J=0 TO 10:NEXT J
660 NEXT I
700 END
```

```
*****************************************
*
* 7-FARBIGE PLAYERS OHNE DMA
*
* ASSEMBLER; ATMAS-II
*
* P. FINZEL 1987
*****************************************
*
* ZEROPAGE-ADRESSEN
*

VPOS0     EQU $F0             VERTIKALPOS. PLAYER 0
VPOS1     EQU $F1             V-POS. PLAYER 1
VPOS2     EQU $F2             V-POS. PLAYER 2
VPOS3     EQU $F3             V-POS. PLAYER 3

*
* DARSTELLUNGS-FLAGS:
*         $FF = NOCH NICHT DARGESTELLT
*         0-7 = DARSTELLUNG LAEUFT
*         8   = ENDE
*

FLAG0     EQU $F4
FLAG1     EQU $F5
FLAG2     EQU $F6
FLAG3     EQU $F7

*

VC        EQU $F8             ZWISCHENSPEICHER F. VCOUNT

*
* OPERATING SYSTEM & HARDWARE
*
SDMCTL    EQU $022F           DMA-KONTROLLREG.
HPOSP0    EQU $D000           HOR.-POSITION
SIZEP0    EQU $D008           BREITE DER PLAYER
GRAFP0    EQU $D00D           FORMREGISTER
COLPM0    EQU $D012           FARBE PLAYER
GRACTL    EQU $D01D           GRAPHIK-KONTROLLREG.
WSYNC     EQU $D40A           WAIT FOR HSYNC
VCOUNT    EQU $D40B           NUMMER DER BILDZEILE
SETVBV    EQU $E45C           ROUTINE INTERRUPTVEKTOR
XITVBV    EQU $E462           ABSCHLUSS DES VBI
```

```
          ORG $A800

          JMP START

*
* HORIZONTAL-POSITIONEN
*

HPOS0     DFB 60
HPOS1     DFB 100
HPOS2     DFB 140
HPOS3     DFB 180

*
* GESCHWINDIGKEITEN
*

VX        DFB $FE,2,$FE,2

*

ZAEHL     DFB 0               HILFSZAEHLER

*
* ------------------------------------------------
* HAUPTPROGRAMM
* ------------------------------------------------
*

START     LDA #30             VERTIKALE START-
          STA VPOS0           POSITIONEN FEST-
          LDA #50             LEGEN
          STA VPOS1
          LDA #70
          STA VPOS2
          LDA #90
          STA VPOS3

          LDA #1              MITTLERE BREITE
          STA SIZEP0          FUER ALLE PLAYERS
          STA SIZEP0+1        AUSWAEHLEN
          STA SIZEP0+2
          STA SIZEP0+3
          LDA #0              GESAMTE DMA
          STA SDMCTL          ABSCHALTEN
          LDA #0              DATENWEG VON ANTIC
          STA GRACTL          ZU GTIA SPERREN

          LDY #VBIPGM:L       VBI-ROUTINE
          LDX #VBIPGM:H       STARTEN
          LDA #7              DEFERRED VBI
          JSR SETVBV
          RTS
```

```
*
* ---------------------------------------------
* VBI-ROUTINE
* ---------------------------------------------
*

VBIPGM   CLD                 ZUR SICHERHEIT
         LDX #3
LOESCH   LDA #0
         STA GRAFP0,X        FORMREGISTER LOESCHEN
         LDA HPOS0,X         HORIZONTALE
         STA HPOSP0,X        POSITIONEN FESTLEGEN
         LDA #$FF            FLAGS AUF NOCH
         STA FLAG0,X         NICHT DARGESTELLT
         DEX
         BPL LOESCH

*
* SORGT FUER BEWEGUNG: VPOSS WIRD GEMAESS
*                      VX VERAENDERT
*

         LDX ZAEHL           WER IST DRAN?
         LDA VPOS0,X         VPOS=VPOS+VX
         CLC
         ADC VX,X
         CMP #16  OBERE GRENZE?
         BCC INVERS
         CMP #112 UNTERE GRENZE?
         BCS INVERS
         STA VPOS0,X
         JMP ZAEHLER

INVERS   LDA VX,X GESCHWINDIGKEIT
         EOR #$FF INVERTIEREN
         STA VX,X
         INC VX,X

ZAEHLER  INC ZAEHL           ZAEHLER AUF
         LDA ZAEHL           NAECHSTEN PLAYER
         CMP #4
         BNE NXTZEIL
         LDA #0
         STA ZAEHL
```

```
*
* PHASE 1 DES KERNELS
*

NXTZEIL  LDA VCOUNT         VCOUNT ZWISCHEN-
         STA VC             SPEICHERN
         LDX FLAG0          DARSTELLUNG?
         BPL PEND0          JA -->
         LDA VPOS0          NEIN, ANFANGS-
         CMP VC             POSITION ERREICHT?
         BNE PL1            NEIN -->
         INX                JA, FLAG ERHOEHEN
PEND0    CPX #8             ENDE ERREICHT?
         BCS PL1            JA -->
         LDA STAB0,X        NEIN, DANN FORM
         STA CS0+1          AUS SHAPE-TAB.
         LDA CTAB0,X        UND FARBE AUS
         STA CC0+1          COLOR-TAB.
         INX                FLAG WEITER
         STX FLAG0          UND MERKEN

*
* JETZT DAS GLEICHE SPIEL FUER PLAYER 1
*

PL1      LDX FLAG1
         BPL PEND1
         LDA VPOS1
         CMP VC
         BNE PL2
         INX
PEND1    CPX #8
         BCS PL2
         LDA STAB1,X
         STA CS1+1
         LDA CTAB1,X
         STA CC1+1
         INX
         STX FLAG1

*
* EBENSO FUER PLAYER 2
*

PL2      LDX FLAG2
         BPL PEND2
         LDA VPOS2
         CMP VC
         BNE PL3
         INX
PEND2    CPX #8
         BCS PL3
         LDA STAB2,X
         STA CS2+1
         LDA CTAB2,X
         STA CC2+1
         INX
         STX FLAG2
```

```
*
* EBENSO FUER PLAYER 3
*

PL3       LDX FLAG3
          BPL PEND3
          LDA VPOS3
          CMP VC
          BNE SEG2
          INX
PEND3     CPX #8
          BCS SEG2
          LDA STAB3,X
          STA CS3+1
          LDA CTAB3,X
          STA CC3+1
          INX
          STX FLAG3

*--------------------------------------
* PHASE 2: GEFUNDENE WERTE IN HARDWARE
*           REGISTER UEBERTRAGEN
* ACHTUNG: DIE NULLEN WERDEN PER
*           PROGRAMM UEBERSCHRIEBEN
*--------------------------------------

SEG2      STA WSYNC           HOR.SYNCHRONISATION
CS0       LDA #0              WERT NACH FORM-
          STA GRAFP0          REGISTER 0
CC0       LDA #0              WERT NACH FARB-
          STA COLPM0          REGISTER 0
CS1       LDA #0
          STA GRAFP0+1        USW
CC1       LDA #0
          STA COLPM0+1
CS2       LDA #0
          STA GRAFP0+2        USW
CC2       LDA #0
          STA COLPM0+2
CS3       LDA #0
          STA GRAFP0+3        USW
CC3       LDA #0
          STA COLPM0+3
          LDA VC              BILDSCHIRMENDE
          CMP #120            ERREICHT?
          BEQ VBIENDE         JA -->
          JMP NXTZEIL         WEITER ==>

*
* JETZT IST DER VBI ZU ENDE!
*

VBIENDE   JMP XITVBV          VBI ENDE
```

```
*
*--------------------------------------
* FORM UND FARBTABELLEN
*--------------------------------------
* PLAYER 0
*

STAB0     DFB $18,$3C,$7E,$FF
          DFB $7E,$3C,$18,$00
CTAB0     DFB 4,8,12,14,12,5,4,0

*
* PLAYER 1
*

STAB1     DFB $7E,$3C,$18,$FF
          DFB $18,$3C,$7E,$00
CTAB1     DFB $4E,$4A,$38,$34
          DFB $32,$44,$46,0

*
* PLAYER 2
*

STAB2     DFB $03,$07,$0F,$1F
          DFB $3F,$7F,$FF,$00
CTAB2     DFB $84,$88,$94,$98
          DFB $A4,$A8,$A4,0

*
* PLAYER 3
*

STAB3     DFB $3C,$7E,$FF,$7E
          DFB $FF,$7E,$3C,$00
CTAB3     DFB $14,$28,$3C,$44
          DFB $58,$6C,$74,0
```