

PETER'S ASSEMBLERECKE

Player-Missile Grafik ganz anders

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 GRCTL 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 lan-

ge 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 effektiv 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. Führt 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 Hardware-Register 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-Programmietechnik 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

Listing 1

```
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 GRCTL=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 HPOS0,64
410 POKE HPOS0+1,96
420 POKE HPOS0+2,128
430 POKE HPOS0+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 HPOS0+1,96-I
520 POKE HPOS0,64-I
530 POKE HPOS0+2,128+I
540 POKE HPOS0+3,160+I
550 FOR J=0 TO 50:NEXT J
560 NEXT I
600 FOR I=64 TO 0 STEP -1
610 POKE HPOS0,64-I
620 POKE HPOS0+1,96-I
630 POKE HPOS0+3,160+I
640 POKE HPOS0+2,128+I
650 FOR J=0 TO 10:NEXT J
660 NEXT I
700 END

```

Listing 2

```

*****
*
*       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.
HPOS0 EQU $D000 Hor.-Position
SIZEP0 EQU $D008 Breite der Player
GRAFP0 EQU $D00D Formregister
COLPM0 EQU $D012 Farbe Players
GRCTL EQU $D01D Graphik-Kontrollreg.
WSYNC EQU $D40A Wait for HSYNC
VCOUNT EQU $D40B Nummer der Bildzeile
SETVBV EQU $E45C Routine f. Interruptvektoren
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 GRCTL 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 HPOS0,X Positionen festlegen

```

```

LDA #$FF      Flags auf noch
STA FLAG0,X   nicht dargestellt
DEX
BPL LOESCH

```

```

*
* sorgt fuer Bewegung: VPOS 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

```

```

*
* und schliesslich 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 u.s.w.
CC1 LDA #0
STA COLPM0+1
CS2 LDA #0
STA GRAFP0+2
CC2 LDA #0
STA COLPM0+2
CS3 LDA #0
STA GRAFP0+3
CC3 LDA #0
STA COLPM0+3
LDA VC              Bildschirmende
CMP #120            erreicht?
BEQ VBIENDE         ja -->
JMP NXTZEIL         weiter ==>

```

```

*
* jetzt erst ist der VBI zu Ende!
*

```

```

VBIENDE JMP X1TVBV   VBI-Ende

```

```

*-----
* Form- und Farbtabellen
*-----
* PLAYER 0

```

```
*
STAB0    DFB $18,$3C,$7E,$FF
          DFB $7E,$3C,$18,$00
CTAB0    DFB 4,8,12,14,12,8,4,0
*
* PLAYER 1
*
STAB1    DFB $7E,$3C,$18,$FF
          DFB $18,$3C,$7E,$00
CTAB1    DFB $4E,$4A,$3B,$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 $A4,$AB,$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
```