

PM-Grafik, Scrolling und Page-Flipping sind alles Techniken, mit denen Atari-Computer vor einigen Jahren Meilensteine gesetzt haben. Doch das I-Punktchen des grafischen Potentials sind die Display-List Interrupts (kurz DLI). Mit deren Hilfe können noch mehr Farben, noch mehr bewegte Objekte oder auch mehrere Zeichensätze gleichzeitig dargestellt werden.

Diese DLIs haben nur einen Nachteil: Sie sind ungemein kritisch in der Ausführungszeit und daher ausschließlich den Assemblerprogrammierern vorbehalten. Das ist natürlich ein Grund für die Assemblerecke, sich dieses Themas anzunehmen. Den Anfang machen zwei Beispiele. Später werden wir uns mit dem Timing der DLIs befassen, ein Thema, das besonders Leute mit etwas Assembler-Erfahrung interessieren wird. Zunächst aber ein paar Grundlagen für alle, die sich mit Interrupts und Rasterzeilen noch nicht so auskennen.

Sie wissen ja sicherlich, dass von Ihrem Computer erzeugte Bild besteht aus 50 Einzelbildern pro Sekunde. Aber auch jedes dieser Einzelbilder (Frames) wird nicht im Ganzen aus gegeben, sondern von einem Elektronenstrahl in einzelnen Rasterzeilen auf den Bildschirm gezeichnet. Nur durch die Trägheit des Auges und der Nachleuchtdauer des Bildschirms sehen wir ein stehen des, meist flimmerfreies Bild.

Für die Grafikprogrammierung braucht man eine Möglichkeit, den Prozessor an die Erzeugung des Bildes zu koppeln. Damit vermeidet man Bildstörungen (Flackern etc.), und Bewegungen werden gleichmäßig und flüssig. Beim Atari wurden zu diesem Zweck zwei Interrupts verwendet. Der eine tritt auf, wenn die Ausgabe eines Frames gerade beendet ist und der Elektronenstrahl zu seinem Ausgangspunkt zurückkehrt. Wir kennen ihn bereits aus früheren Assembler-ecken; es ist der VBI. Der zweite Interrupt, der DLI, lässt sich bereits während des Bildaufbaues auslösen.

Es dürfte inzwischen bekannt sein, dass die Grafik des Atari durch einen eigenen Videoprozessor, den Antic, aufgebaut wird. Auch dieser Prozessor hat ein eigenes Programm (die Display-List), in dem festgelegt wird, welcher Speicherbereich in welchem Modus darzustellen ist. So besteht die Display-List eines GRAPHICS-0 Bildschirms u.a. aus 24 Anweisungen zur Darstellung einer Textzeile. Man kann nun bei jeder Antic-Anweisung ein Bit setzen (das höchstwertige), so dass der Videoprozessor einen Display-List-Interrupt auslöst.

Die Möglichkeiten einer solchen Einrichtung sind enorm. Da der Interrupt auftritt, während der Elektronenstrahl noch unterwegs ist, kann man einige für die Bilderzeugung verantwortliche Hardware-Register ändern. So ist es möglich, oberhalb und unterhalb der Interruptzeile verschiedene Zeichensätze zu benutzen, wenn man CHBASE (\$D409) mit einem DLI ändert. Auch kann man Farbregister neu besetzen und somit mehr Farben darstellen oder die Positionsregister der Player verändern etc. Eine der eindrucksvollsten Möglichkeiten für DLI habe ich bei meinem Programm "Schreckenstein" verwendet. Mit Hilfe eines DLIs werden die Register für Fein-Scrolling (HSCROL und VSCROL) in der Mitte des Bildschirms geändert. Dadurch sind zwei Fenster möglich, die sich vollkommen unabhängig voneinander scrollen lassen.

Technik

Bleibt eigentlich nur noch zu klären, wie man mit DLIs um geht. Drei Schritte sind zu tun:

1. Das DLI-Bit in der gewünschten Zeile der Display-List setzen.
2. Den DLI-Vektor VDSLST (\$0200, \$0201) auf die Interrupt-Routine richten.
3. Den Interrupt mit Bit 7 von NMIEN (\$D40E) freigeben.

Nach diesen Schritten würde der Vektor VDSLST durchsprungen, nachdem Antic die Anweisung mit dem gesetzten DLI-Bit verarbeitet hätte (genaueres dazu noch später). Was jetzt noch fehlt, ist die vom DLI angesprungene Routine. Da dieser Programmteil im Interrupt läuft, muss man ein paar Regeln sehr sorgfältig beachten:

Alle Register müssen vor der Verwendung auf den Stack gerettet werden. Braucht man nur den Akku, so genügt ein PHA-Befehl. Vor Veränderungen der Hardware-Register sollte ein Schreibbefehl auf die Adresse WSYNC (\$D40A) ausgeführt werden. Damit lassen sich Störungen vermeiden; der genaue Mechanismus wird uns später noch beschäftigen. Wichtig ist auch, dass alle Veränderungen direkt an den Hardware-Registern ausgeführt werden müssen, denn die Schattenregister werden ja erst im VBI übertragen. Vor dem Verlassen der DLI Routine müssen die ursprünglichen Inhalte der Register wiederhergestellt werden. Ein ganz einfaches Beispiel für eine DLI-Routine und deren Verwendung in Basic finden Sie in Listing 1. Hier wird nur die Hintergrundfarbe des Textes etwa in der Mitte des Bildschirms geändert.

Mehr DLIs

Natürlich kann der DLI noch viel, viel mehr. In den Listings 2 und 3 bekommt jede Zeile eines GRAPHICS-0 Bildschirms mit Hilfe eines DLIs eine eigene Farbe zugewiesen. Die Farben sind in der Tabelle FARBTAB (ab \$603) zusammengefasst und können z.B. von Basic geändert werden - eine praktische Methode, wenn man einzelne Zeilen hervorheben will. Im Basic Listing 3 ist ein kleines Demo enthalten, an dem Sie sehen, was man alles damit anstellen kann.

Listing 2 zeigt Ihnen die inneren Vorgänge dieses schon relativ komplizierten Interrupt Beispiels, in dem VBI und DLI Zusammenarbeiten. Im ersten Teil werden VBI- und DLI Routinen wie oben beschrieben eingerichtet. Der VBI, der ja immer nach der Ausgabe eines kompletten Bildes stattfindet, hat nur die Aufgabe, einen Zeiger in die Farbtabelle auf Null zu setzen. Ist der erste DLI aus gelöst, wird das vom Zeiger markierte Tabellenelement (nach einem STA WSYNC) in das Farbregister 2 übertragen. Danach wird der Zeiger weiter geschaltet, so dass der nächste DLI das zweite Element überträgt. Sind alle 24 DLIs vorbei, so tritt wieder ein VBI in Erscheinung, setzt den Zeiger zurück, und das Spiel beginnt von vorne. Sie werden sich vielleicht fragen, warum im vorigen Beispiel kein VBI zum Rücksetzen des Farbregisters nötig war. Ganz einfach, weil die VBI-Routine im ROM diese Aufgabe beim Kopieren der Schattenregister in die Hardware-Register automatisch erledigt.

Besonderheit

Wenn Sie sich die Routine zum Setzen der DLI-Bits in der Display-List genauer ansehen, werden Sie feststellen, dass der erste DLI nicht, wie anzunehmen, in der ersten Zeile des GRAPHICS-0 Bildschirms ausgelöst wird, sondern schon davor. Jede Display-List beginnt mit drei Anweisungen für je 8 Leerzeilen und genau bei der letzten dieser Anweisungen wird das DLI-Bit schon gesetzt. Das hat seinen Grund. Wenn nämlich eine Anweisung in der Display-List mehrere Rasterzeilen erfordert (z.B. der Textmodus Antic 2), dann wird der DLI erst in deren letzter Rasterzeile ausgelöst. Würden wir daher die Anweisung für die erste Textzeile mit einem DLI-Bit versehen, so hätte der Farbwechsel erst Auswirkungen auf die zweite Zeile. Wir lernen daraus, das DLI-Bit immer in der Zeile vorher zu setzen.

Timing

Obwohl der 6502 in Ihrem Atari mit seinen 1.79 MHz schon beachtlich schnell läuft, muss man bei einer DLI-Routine mit der Laufzeit aufpassen. Keine Angst, es ist noch lange nicht kritisch, wenn wie bisher nur eine einziges Farbregister geändert werden soll. Aber wenn man mehr als 3 Hardware-Register überschreiben will, muss schon mit jedem Maschinenzklus gerechnet werden.

Wie viel Zeit bei einem DLI zur Verfügung steht, wird an Bild 1 klar. Es zeigt zwei Rasterzeilen eines Bildes, wobei in der ersten Zeile ein DLI stattfinden soll. Man muss noch wissen, dass die Zeit, die der Elektronenstrahl für eine Rasterzeile benötigt, beim 6502 für 114 Zyklen ausreicht. Es vergeht nun schon einige Zeit von der Auslösung des DLIs bis zum Sprung durch den Vektor VDSLST. Das führt dazu, dass die DLL-Routine erst ab Zyklus 33 der ersten Zeile zum Zuge kommt.

Das Register WSYNC

Man könnte jetzt einfach ein Farbregister mit einer LDA/STA-Kombination neu besetzen. Die Folge davon wäre aber, dass der Farbwechsel mitten in der Rasterzeile stattfinden würde. Durch einige zusätzliche Effekte (DMA!) zittert die Farbgrenze auch noch hin und her und führt zu einer Verunstaltung der Grafik. Aus dieser Klemme hilft uns das Register WSYNC (\$D40A, Wait for SYNC). Ein Schreibbefehl auf diese Adresse hält den Prozessor solange an, bis der Elektronenstrahl das Ende der Rasterzeile erreicht hat. Genauer gesagt nimmt der 6502 sieben Zyklen vor Anfang der nächsten Rasterzeile seine Arbeit wieder auf. Zu diesem Zeitpunkt befindet sich der Elektronenstrahl außerhalb des sichtbaren Bereiches und jegliche Änderungen der Grafik bleiben unsichtbar.

Da der Haltezustand des Prozessors im Zyklus 107 aufgehoben wird, muß der Schreibbefehl, der selbst vier Zyklen benötigt, spätestens bei 103 beginnen. Man kann sich leicht ausrechnen, dass für den ersten Teil des DLIs maximal 70 Zyklen verbleiben. Da der Speicher des Ataris aber mit dynamischen Speicher-ICs aufgebaut ist, werden einige Zyklen für den sog. "Refresh" abgezweigt (9 pro Zeile). Findet der DLI in einer Zeile statt, in der der Antic Informationen für die Bildschirmgrafik per DMA aus dem Speicher holen muss, so verringert sich die Zeit für den DLI ganz enorm (20 bis 40 Zyklen beansprucht der Antic je nach Grafikmodus für sich). Man kann also davon ausgehen, dass vom Sprung durch den DLI Vektor bis zu WSYNC ca. 21 bis 61 Zyklen zur Verfügung stehen. Was ist in dieser Zeit zu tun? Erstens müssen die Register gerettet werden, zweitens bietet es sich an, die neuen Werte für Farben etc. gleich in die CPU-Register zu schreiben.

Hier ein kleines Beispiel, in dem die Änderung von drei Farbregistern vorbereitet wird:

```
PHA
TXA
PHA
TYA
PHA
LDA +FARBE1
LDX +FARBE2
LDY +FARBE3
STA WSYNC
```

Damit hat man die zweite Phase des DLIs schon gut vor bereitet. Man kann die Werte nach WSYNC nun mit drei Store-Befehlen schnellstmöglich in die Hardware-Register übertragen. Die Fortsetzung des obigen Beispiels wäre z.B. :

```
STA COLPFO
STX COLPF1
STY COLPF2
```

Das ist auch bitter nötig, denn die Zeit ist knapp. Von WSYNC bis zum Zeilenende sind 7 Zyklen, bis zum Auftauchen des Strahles am Anfang der nächsten Rasterzeile sind etwa 17 Zyklen Zeit. Etliche davon beansprucht die DMA des Antic (1 bis 3), mit eingeschalteter PM-Grafik sind es 5 mehr. Das bedeutet, dass in dieser "kritischen" Phase des DLIs ca. 16 bis 23 Zyklen genutzt werden dürfen. Bedenkt man, dass die drei Store-Befehle schon 12 Zyklen dauern, dann wird klar, dass hier keine großen Sprünge möglich sind.

Viel Zeit

Für die letzte Phase des DLIs gibt es dann keine zeitliche Einschränkung mehr. Sie sollte nur zu Ende sein, bevor der DLI beim nächsten Frame ausgelöst wird, also in etwa 24000 Zyklen. Man benutzt diese Phase, um die Register in umgekehrter Reihenfolge wieder vom Stack zu holen und den Prozessor per RTI vom Interrupt zu erlösen. Zu obigem Beispiel wäre dies:

```
PLA
TAY
PLA
TAX
PLA
RTI
```

Natürlich kann man auch noch andere Aufgaben an dieser Stelle erledigen. Etwa einen Zeiger weiterführen (s. Beispiel) oder den DLI-Vektor auf eine andere Routine richten. Das würde dazu führen, dass ein weiterer DLI ganz andere Aufgaben ausführen könnte. Denkbar wären auch DLIs, die mit mehreren STA WSYNC arbeiten und sich somit über mehrere Rasterzeilen erstrecken. Auf diese Weise erzeugt man diese Farbbalken, die so ungeheuer plastisch aus sehen. An diesen Ausführungen haben Sie sicherlich bemerkt, dass DLIs ganz schön verzwickt sein können. Das trifft nicht zu, wenn man nur eine Farbe ändern will, aber es gibt auch Fälle, in denen man mit jedem Maschinenzklus rechnen muss. Ich erinnere mich noch bestens an einen DLI in meinem letzten Spiel, der vier Player versetzen, zwei Farben ändern und die Bildschirmbreite modifizieren sollte. Im wahrsten Sinne des Wortes - das kostete Schweiß!

Peter Finzel

```
10 REM *****
20 REM * ZWEIFARBIGESGR.0
30 REM *
40 REM * EINFACHES DLI-BEISPIEL
50 REM *****
60 REM
100 REM * DLI-ROUTINE IN PAGE 6
110 FOR I=1536 TO 1546:READ A:POKE I,A:NEXT I
115 REM *          PLA
120 DATA 72
125 REM *          LDA #$C2
130 DATA 169,194
135 REM *          STA WSYNC
140 DATA 141,10,212
145 REM *          STA COLPF2
150 DATA 141,24,208
155 REM *          PLA
160 DATA 104
165 REM *          RTI
170 DATA 64
200 REM * DLI-BIT IN DL SETZEN
210 DLIST=PEEK(561)*256+PEEK(560)
220 POKE DLIST+16,128+2
300 REM *          DLI FREIGEBEN
310 VDSLST=512:NMIEN=54286
320 POKE NMIEN,64:REM *    DLI AUS
330 POKE VDSLST,0:REM *    VEKTOR
340 POKE VDSLST+1,6:REM *  EINTRAGEN
350 POKE NMIEN,192:REM *   DLI EIN
```

```
100 REM *****
110 REM * Vielfarbeffekt in GR.0
120 REM *      BASIC-Loader
130 REM * P. FINZEL      1986
140 REM *****
200 MPGM=1536:FARBTAB=1540
220 GOSUB 500
230 X=USR(MPGM)
300 FOR I=0 TO 23
310 X=PEEK(FARBTAB+I)
320 POKE FARBTAB+I,14
330 FOR T=0 TO 50:NEXT T
340 POKE FARBTAB+I,X
350 NEXT I
360 GOTO 300
500 REM * Vielfarbeffekt installieren
510 S=0:RESTORE 600
520 FOR A=1536 TO 1664:READ D:POKE A,D:S=S+D:NEXT A
530 IF S<>14129 THEN ? "DATEN-FEHLER !":STOP
590 RETURN
600 DATA
104,76,29,6,20,34,52,66,84,98,116,130,148,162,180,194,212
610 DATA
226,244,2,20,34,52,66,84,98,116,130,0,173,48,2,172,49,2,133
620 DATA
212,132,213,160,2,177,212,9,128,145,212,200,177,212,9,128
630 DATA
145,212,200,200,200,177,212,9,128,145,212,192,27,208,245
640 DATA
169,0,141,14,212,169,98,141,0,2,169,6,141,1,2,169,121,141
650 DATA
34,2,169,6,141,35,2,169,192,141,14,212,96,72,138,72,174,28
660 DATA
6,189,4,6,141,10,212,141,24,208,232,142,28,6,104,170,104
670 DATA 64,169,0,141,28,6,76,95,228
```

```

*****
* DISPLAY-LIST-INTERRUPTS
*
* VIELFARBIGES GRAPHICS 0 DISPLAY
*****

COLPF2    EQU $D018          HARDWARE-REG . FUER FARBE
WSYNC     EQU $D40A          WARTEN AUF ZEILENENDE
NMIEN     EQU $D40E          NMI-FREIGABE

VDSLST    EQU $0200          VEKTOR FUER DLIS
VVBLKI    EQU $0222          VBI-VEKTOR (IMMEDIATE)
SDLSTL    EQU $0230          ZEIGER AUF D-LIST
SYSUBV    EQU $E45F          VBL-ROUTINE IM ROM

HILFZP    EQU $D4            HILFSREGISTER IN ZERO-P.

*
* IM ATMAS MONITOR MIT G 0600 STARTEN
*
        ORG $0600

        NOP                  FUER BASIC HIER PLA
        JMP START            TABELLE UEBER SPRINGEN

*
* TABELLE DER ZEILENFARBEN:
*

FARBTAB   DFB $14,$22,$34,$42,$54,$62
          DFB $74,$82,$94,$A2,$B4,$C2
          DFB $D4,$E2,$F4,$02,$14,$22
          DFB $34,$42,$54,$62,$74,$82

ZEIGER    DFB 0              ZEIGER IN TABELLE

START     LDA SDLSTL          D-LIST-ZEIGER IN
          LDY SDLSTL+1        ZERO-PAGE KOPIEREN
          STA HILFZP          F. INDIREKTE
          STY HILFZP+1        ADR.-ART
          LDY #2              DLI-BIT IN
          LDA (HILFZP),Y      DRITTER LEERZEILE
          ORA #$80            SETZEN
          STA (HILFZP),Y
          INY                  DLI-BIT IN DER
          LDA (HILFZP),Y      LMS-ANWEISUNG
          ORA #$80            DER D-LIST SETZEN
          STA (HILFZP),Y
          INY                  LMS-ADRESSE
          INY                  UEBERSPRINGEN
INTSET     INY                SCHLEIFE F.
          LDA (HILFZP),Y      DLI-BIT IN RESTL.
          ORA #$80            DISPLAY-LIST
          STA (HILFZP),Y
          CPY #27             SCHON ALLE ZEILEN?
          BNE INTSET          NEIN->

```

```

*
* DLI UND VBI AKTIVIEREN
*
        LDA #0                INTERRUPTS AUS
        STA NMEN              (VBI & DLI)
        LDA #DLIPGM:L         DEN VEKTOR AUF

        STA VDSLST            DIE INT. -ROUTINE
        LDA #DLIPGM:H         SETZEN
        STA VDSLST+1
        LDA #VBIPGM:L         VBI-PROGRAMM
        STA VVBLKI            EINBINDEN
        LDA #VBIPGM:H         (IMMEDIATE)
        STA VVBLKI+1
        LDA #SC0 UND VBI & DLI
        STA NMEN              FREIGEBEN
        RTS

*
* DLI-ROUTINE VIEL ARBEN-EFFEKT IN GR.0
*
DLIPGM   PHA                AKKU RETTEN
        TXA
        PHA                X-REG. RETTEN
        LDX ZEIGER          INDEX LADEN
        LDA FARBTAB,X        FARBE AUS TABELLE
        STA WSYNC            ZEILENENDE ABWARTEN
        STA COLPF2           FARBE EINTRAGEN
        INX                  ZEIGER WEITER-
        STX ZEIGER           SCHALTEN
        PLA                  X-REG. VOM STACK
        TAX
        PLA                  AKKU BESORGEN
        RTI                  SCHLUSS !

*
* VBI-ROUTINE VIELFARBENEFFEKT
*
VBIPGM   LDA #0                ZEIGER AUF ANFANG
        STA ZEIGER          DER FARBTABELLE
        JMP SYSVBU          VBI WEITER

```