Inhaltsverzeichnis
von

Atari Power
Das Superbuch

Sehr geehrte Atarianer/innen

Wir beglückwünschen Sie zum Kauf von ‚Ataripower'. Wir wollen uns nicht lange mit einem Vorwort o.ä. aufhalten, da dies nur Platz kostet, der sinnvoller genutzt werden kann. Sie werden sich sicher über die Aufmachung des Buches wundern. Aber aus wirtschaftlichen Gründen sahen wir uns zu diesem Schritt gezwungen. Wir sahen uns zu dieser Entscheidung gezwungen, nachdem klar war, dass das Buch nicht mehr als 30,- DM kosten durfte. Ein richtiges Buch, in Kleinserie gedruckt und gebunden, hätte einen Preis von 8.-DM. Diese 8.- DM wollten wir nicht an Sie weitergeben. Wir nehmen an, dass dies mit Ihrem Einverständnis geschehen ist. Zugegeben, ‚Ataripower' als echtes Buch sähe zwar besser aus, mehr drinstehen würde dort allerdings nicht. Na ja, wie dem auch sei, wir wünschen Ihnen mit diesem Buch viele interessante Stunden. Und ganz sicher werden Ihre Programme nach dem Studium von ‚Ataripower' professioneller sein. ‚Wir',dass sind übrigens die Betreiber des Atari Club Düsseldorf.

So, jetzt aber los. Aber zuvor noch einige wichtige Informationen. Sämtliche hier aufgeführten Programme, haben Democharakter. Sie werden hier keine fertigen Spiele o.ä. finden. Die müssen Sie schon selbst schreiben. Jedes dieser Programme ist dafür gedacht, in andere bereits bestehende Programme eingefügt oder in ein noch zu schreibendes Programm eingeplant zu werden. Sie sollten hierbei vorsichtig vorgehen. Meistens werden Maschinenunterroutinen verwendet. Die notwendigen Daten hierzu stehen immer in DATA

Zeilen. Sie dürfen sich dort nicht vertippen, sonst ist das Programm verloren. Deshalb die Regel, die Sie immer beherzigen müssen : erst speichern dann starten. Wenn sich der Atari jetzt aufhängt, kann das Programm neu geladen und der Fehler gesucht werden.

Wie Sie als fortgeschrittener sicher bald sehen werden, liegen viele Maschinenroutinen in der Page 6 (1536-1796 dez.). Wenn Sie diese Unterroutinen miteinander kombinieren möchten, müssen Sie diese an eine andere Stelle verlegen. Mit einem Einbau in bereits bestehende Programme ist dort zu prüfen, ob die Page 6 noch frei ist. Die Page 6 ist ein Speicherbereich, den der Computer bei normalem Betrieb nicht anrührt. Dies ist so ziemlich der einzige Platz, der gegen Überschreibung sicher ist. Da der Atari keine feste Speicherbelegung hat, sondern einen Bildspeicher usw. je nach Bedarf hin- und herschiebt, sollte man den Atari überlisten und somit festen Speicherplatz schaffen. Um dies zu machen, gehen Sie so vor:

Im Computer gibt es eine Speicherzelle, die dem OS mitteilt, wievil Seiten (eine Seite sind 256 Byte) ihm zur Verfügung stehen. Zieht man von dieser Zahl 4 ab und schreibt diese wieder in die Zelle hinein, so steht nun 1 K RAM zur Verfügung, der nicht gelöscht wird. Das Ganze sieht als Programm so aus:

10 X=PEEK(106): POKE,X-4: ANFANG=PEEK(106)*256

In der Variable ANFANG steht nun die Anfangsadresse des geschaffenen Freiraumes. Allerdings steht nach einer solchen Prozedur natürlich 1K RAM weniger zur Verfügung.

Kleine Routinen lassen sich auch sehr gut in einem String unterbringen.

Dies geht aber nur, wenn das Maschinenprogramm mittels USR aufgerufen wird. Diese Methode ist sehr elegant und spart eine Menge DATA's. Man geht dabei so vor:

```
10 DIM MASCH$(X): FOR A=1 TO X: READ DATEN: MASCH$(A,A)=CHR$(DATEN)
   : NEXT A
```

X ist hierbei die Anzahl der vorhandenen Daten. Nach Aufruf dieser Zeile stehen die benötigten Daten in einem String MASCH$. Das Maschinenprogramm wird nun mit dem Kommandeo X=USR(ADR(MASCH$)) aufgerufen. Mit dieser Methode spart man gegenüber der DATA Methode bei 1000 Daten etwa 3000 Byte ein.

Auch hierbei gilt, wie auch sonst überall : Durch Probieren versteht man am schnellsten.

Nun noch ein Hinweis:

Sämtliche Druckfehler sind beabsichtigt, da dieses Buch auch etwas für Leute bieten soll, die nur nach Fehlern suchen.

Bei fehlerhaften Programmen oder sonstigen Fehlern bitten wir um Nachricht an:

A.Müller

Karlstr.11

4000 Düsseldorf

Aber bitte nur schriftlich !

Ein neuer, blinkender Cursor

Wer wollte nicht schon immer mal einen neuen Cursor haben. Hier ist nun ein Generieru7ngsprogramm für einen blinkenden Cursor, der jede gewünschte Farbe annehmen kann. Ab Zeile 409 stehen 4 DATA Zeilen für 4 verschiedene Cursorfiguren. Sie können nur eine Zeile einsetzen. Die anderen müssen weggelassen werden. Der Cursor wird so gemacht, wie sie es im Player Missile Kurs gelernt haben. Achten Sie nur darauf, dass die ersten 4 Spalten nicht genutzt werden

(1,2,4,8). Der Cursor wird aus dem Player 2 hergestellt. Der Cursor wird mit POKE 752,1 eingeschaltet. Bei POKE 752,0 wird der normale Atari Cursor eingeschaltet. Das Programm generiert eine sichere Zone, indem es 33 Pages vom RAMTOP subtrahiert. Dies kostet natürlich Speicherplatz. Wenn nur Graphics 0 benutzt wird, kann die 33 in der Zeile 170 zueiner 6 geändert werden. Nun steht mehr RAM zur Verfügung.

```
10 GOSUB 160
20 POKE 82,0
30 FOR Z=0 TO 180:READ A:POKE X+Z,A
40 POSITION 26,5:? X+Z;:NEXT Z:POKE 752,0:?
50 ? "POKE 752,1 schaltet unseren Cursor ein”
60 ? "POKE 752,0 schaltet den Atari Cursor ein”
70 ? "Gib Blinkgeschwindigkeit ein (1-64) ";:TRAP 70:INPUT I
80 IF I<1 OR I>64 THEN 70
90 POKE X+55,I
100 ? :? "Welche Cursorfarbe (0-255) ";:TRAP 100:INPUT FARBE
110 IF FARBE<1 OR FARBE>255 THEN 100
120 POKE X+74,FARBE
130 ? :? "Achtung! Diese Routine benuetzt PAGE 6 als P/M RAM
(Player 2)”
140 Y=USR(X)
150 POKE 752,1:END
160 GRAPHICS 0:POKE 752,1:POKE 82,1:POKE 83,39
170 X=(PEEK(106)-33)*256
180 RETURN
190 DATA 104,165,212,24,105,36,133,212
200 DATA 165,212,105,0,133,213,162,7
210 DATA 160,144,177,212,157,0,6,136
220 DATA 202,16,247,164,212,166,213,169
230 DATA 7,76,92,228,173,240,2,240
240 DATA 124,173,43,2,240,4,169,0
250 DATA 240,6,173,8,6,24,105,8
260 DATA 141,8,6,48,104,160,255,169
270 DATA 0,153,0,6,136,192,8,208
280 DATA 248,169,10,141,194,2,141,20
290 DATA 208,169,1,141,111,2,141,27
300 DATA 208,169,0,141,10,208,169,0
310 DATA 141,7,212,169,2,141,29,208
320 DATA 169,58,141,47,2,141,0,212
330 DATA 166,85,164,84,165,87,13,147
340 DATA 2,240,12,173,191,2,201,4
350 DATA 208,35,152,24,105,20,168,138
360 DATA 10,10,105,48,141,2,208,152
370 DATA 10,10,10,105,39,168,162,7
380 DATA 189,0,6,153,0,6,136,202
390 DATA 16,246,76,98,228,169,0,141
400 DATA 2,208,76,98,228
409 REM Normaler Cursor
410 DATA 240,240,240,240,240,240,240,240
419 REM Fenster
420 DATA 240,144,144,144,144,144,144,240
429 REM Punkt
430 DATA 0,0,0,0,0,0,0,96
439 REM Dreieck
440 DATA 0,64,32,16,16,32,64,0
```

Neustart bei RESET

Um ein Programm zu schützen, muss es erstmal gegen die Unterbrechung von BREAK und RESET Tasten gesichert werden. Wie die BRAK Taste ausgeschaltet werden kann, steht an anderer Stelle in diesem Buch. Um die RESET Taste auszuschalten, müssen die folgenden Programmzeilen am Anfang des Programms gesetzt werden. Nach Drücken von RESET startet das Basicprogramm, als wäre RUN eingegeben worden.

```
10 FOR A=1536 TO 1583: READ B: POKE A,B: NEXT A
20 POKE 12,0: POKE 13,6, POKE 9,2
30 DATA 160,240,185,47,5,145,88,200,208,248,169,13,141,74,3,169,0
40 DATA 133,2,169,6,133,3,169,2,133,9,76,0,160,0,0,0,48,47,43,37
50 DATA 24,20,18,12,17,18,26,50,53,46,0
```

Graphics 8 mit Graphics 0 mischen

Wer wollte nicht schon immer mal ein Graphics 8 Fenster mit Graphics 0 Zeichen füllen. Aus Basic ist dies ohne Unterstützung von Maschinensprache nur langsam möglich. Man liest die Daten des darzustellenden Zeichens aus dem ROM und schreibt sie dann in den Bildspeicher. Bei einer größeren Anzahl von Zeichen nimmt dies beträchtliche Zeit in Anspruch. Deshalb hier ein Maschinenprogramm. Dieses Programm hat eine Besonderheit : Das folgende lange Programm fabriziert eine einzige Zeile 0. In dieser Zeile ist das benötigte Naschinenprogramm enthalten. Diese Zeile wird dann mit LIST"D:MISCH.LST",0 bzw. mit LIST"C:",0 abgespeichert. Das Generierungsprogramm wird nun nicht mehr benötigt. Diese abgelistete Zeile 0 können Sie nun mit ENTER"D:MISCH.LST" bzw. ENTER"C:" an jedes andere Programm anfügen. Achten Sie darauf, dass die Zeile 0 nicht editiert werden darf.

Man könnte in einem Graphics 0 Fenster so schreiben :

POSITION X,Y

PRINT TEXT$

Wenn Sie im Graphics 8 Fenster schreiben wollen, müssen Sie folgendermassen vorgehen:

ADR=(PEEK(136)+256*PEEK(137)+5)

A=USR(ADR,X,Y,ADR(TEXT$),LEN(TEXT$))

Das ist alles. Natürlich muss dann die Zeile 0 vorhanden sein. Das Programm kann man hervorragend einsetzen, um Integrale, Wurzeln usw. darzustellen.

```
0 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
1 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
2 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
3 REM XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
5 REM Genau 45 mal X in einer Zeile
10 AD=PEEK(136)+256*PEEK(137)
20 POKE AD+2,204:POKE AD+3,204
30 FOR I=0 TO 2: FOR J=0 TO 5:POKE AD+51*I+50+J,88:NEXT J:NEXT I
40 LIST 0
50 FOR I=1 TO 198:READ X:POKE AD+I+4,X:NEXT I
60 LIST 0
210 DATA 104,201,4,240,9,170,240,5,104,104,202,208,251,96,104,133
,215,104,133,214
220 DATA 104,104,168,104,133,217,104,133,216,104,104,240,236,133,
212,24,165,214,101,88
230 DATA
133,214,165,89,101,215,133,215,152,240,15,165,214,105,64,133,214,16
5,215,105
240 DATA 1,133,215,136,208,241,132,221,160,0,132,220,177,216,160,
0,170,16,1,136
250 DATA 132,213,138,41,96,208,4,169,64,16,14,201,32,208,4,169,0,
16,6,201
260 DATA 64,208,2,169,32,133,218,138,41,31,5,218,133,218,169,0,
161,3,6,218
270 DATA 42,202,208,250,109,24,2,133,219,164,221,177,218,69,213,
164,220,145,214,200
280 DATA 132,220,196,212,208,182,24,165,214,105,40,133,214,144,2,
230,215,230,221,169
290 DATA 8,197,221,208,159,96
320 DATA 155,40,67,41,32,98,121,32,65,67,68,155,155,246,239,238,
160,200,174
330 DATA 208,174,203,245,229,243,244,229,242,243,155,155,155
```

Basic Abschaltung für Atari 400/800 Besitzer

Jeder, der einen Atari 400 oder einen Atari 800 sein eigen nennt, hat bestimmt schon auf Abhilfe gesonnen, wie man sein Basic ein-

bzw. ausschaltet, ohne es jedes Mal aus dem Schacht zu ziehen. Dabei wird nämlich jedes Mal der Computer ein- und ausgeschaltet. Das dies dem Computer nicht gut tut, (Spannungsspitzen können ihn zerstören) lässt sich denken.

Wir haben nun auf Abhilfe gesonnen und folgende kleine Schaltung entwickelt, die es Ihnen ermöglicht, mittels eines Schalters das Modul je nach Bedarf zuzuschalten. Na ja, Schaltung ist vielleicht etwas übertrieben. Eigentlich besteht das ganze nur aus einem Schalter. Diesen bauen Sie irgendwo neben der Modulklappe ein. Am besten setzen sie über den Schalter eine Leuchtdiode (LED) ein, die Ihnen anzeigt, ob das Modul eingeschaltet ist.

Für den Einbau nehmen sie am besten dünnen Draht. Der geeignetste Draht ist 0.1 mm Kupferlackdraht. Achten Sie darauf, dass dieser beim Einbau nicht irgendwo scheuert. Die Metallkanten der Deckplatte des Moduls müssen mit tesafilm o.ä. entschärft werden.

Der Einbau geht folgendermassen vonstatten :

1. Teil

- Entnehmen Sie das Basicmodul aus dem Schacht und schrauben Sie die Schraube auf der Rückseite heraus.
- Sie können nun die Metallplatte des Moduls durch leichtes Drücken nach oben entfernen.
- Sie sehen nun das Innenleben des Moduls. Holen Sie die Platine aus dem Gehäuse, indem Sie den Halteclip vorsichtig wegbiegen und die Platine herausnehmen. Leider klemmt die Platine gelegentlich, sodas man vorsichtig etwas Gewalt anwenden muss.
- Legen Sie die Platine jetzt mit der Bestückungsseite nach oben vor sich auf den Tisch.
- Wenn Sie sich jetzt Bild Nr.1 anschauen, sehen Sie, dass Sie eine Leiterbahn unterbrechen müssen. Diese Leiterbahn ist wichtig, denn sie ist die Kennung für den Computer, ob ein Modul in ihm steckt oder nicht. Wenn Sie diese Leiterbahn nun mit einem Schalter überbrücken, können Sie das Modul jederzeit ein- oder ausschalten.

Unterbrechen Sie mit einem scharfen Messer die Leiterbahn. Achten Sie darauf, dass Sie die anderen Leiterbahnen nicht beschädigen. (Eine Leiterbahn ist ca.0,035 mm dick)

- Sie löten jetzt an die beiden gekennzeichneten Stellen je einen Draht.
- Wenn Sie sich auch eine Leuchtdiode einbauen wollen, müssen sie für die Spannungsversorgung noch einen Draht anschließen. Dieser wird auf der Rückseite der Platine angelötet. (Siehe Bild Nr.2)
- Kennzeichnen Sie jetzt die Enden der Drähte, damit Sie sie bei geschlossenem Modul auch auseinanderhalten können.
- WICHTIG !! Nun kleben sie um die oberen Kanten der abdeckplatte etwa zwei Lagen Tesafilm, damit diese scharfe Kante Ihr Kabel nicht beschädigt.
- Setzen Sie jetzt die Platine wieder einund bauen Sie das Modul wieder zusammen. Dabei müssen die 3 Kabel zwischen dem Gehäuse und dem Tesafilm liegen. Sie können natürlich auch ein kleines Loch in das Gehäuse bohren, aus dem Sie dann Ihr Kabel herausführen.

2. Teil

- Jetzt legen Sie das Modul erst mal zur Seite und widmen sich dem Computer. Von diesem entfernen Sie sämtliche Anschlüsse und vorhandenen Module.
- Jetzt drehen Sie ihn auf den Kopf und schrauben ihn auf.
- Nun drehen Sie den Computer wieder richtig herum und heben vorsichtig die Deckplatte ab. Sie können sich jetzt einen geeig-

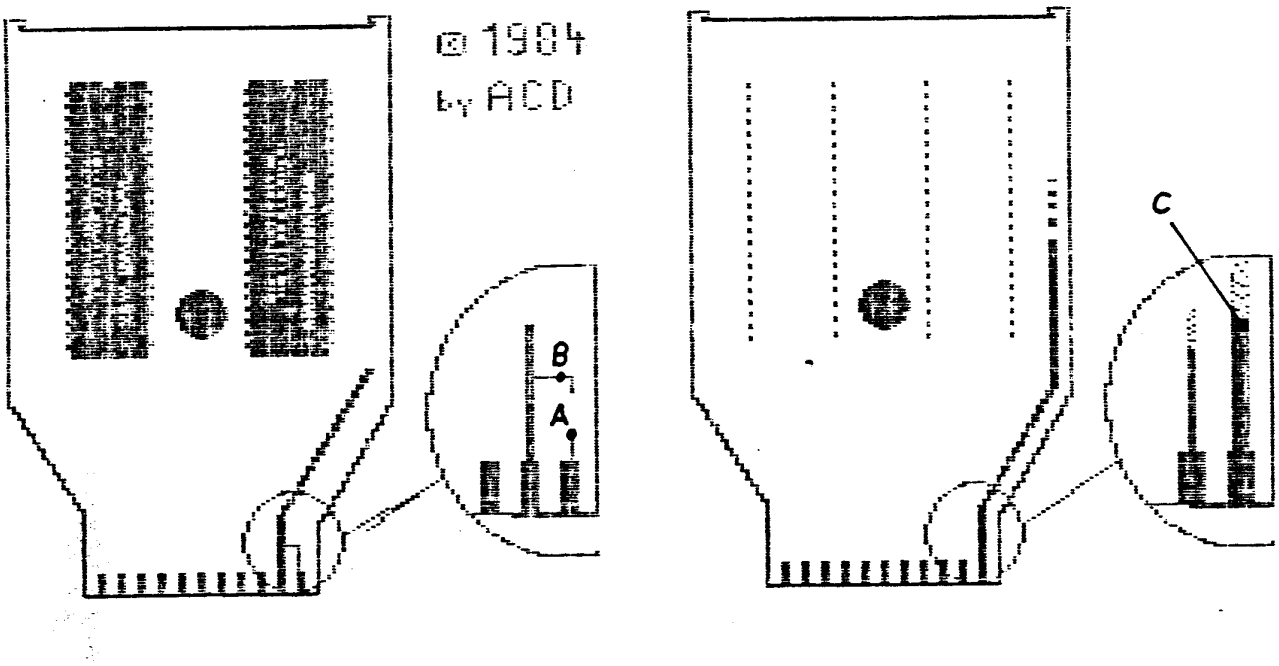
neten Platz für den Schalter suchen. Beim Atari 800 ist links neben der Klappe (neben dem Atarizeichen) noch genügend Platz.

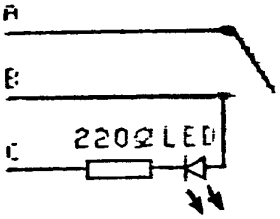
- Dort bohren Sie zwei passende Löcher in das Plastik. Achten Sie aber darauf, dass Sie das Loch nicht zu tief ansetzen, sonst können Sie die Mutter nicht mehr auf das Schaltergewinde aufdrehen.

- Nachdem Sie nun die beiden Köcher gebohrt haben, setzen Sie jetzt den Schalter und die LED ein. Nun löten wir dort die Kabel des Moduls fest. (Bild Nr. 3) Vergessen Sie nicht den Widerstand für die Leuchtdiode.

- Nun wird das Modul in den Schacht gesetzt, und der Computer zugeschraubt. Fertig.

Achten Sie bitte auf folgendes : Auf den Leiterbahnen des Moduls befindet sich ein Schutzlack. Damit Sie Ihre Drähte festlöten können, müssen Sie diesen Lack erst mit einem Messer wegkratzen. Achten Sie darauf, dass Sie keine Kurzschlüsse verursachen.





Einführung in die Player Missile Graphik

Der Atari Computer hat neben hervorragendem Sound und Graphik die Möglichkeit, auch 8 frei bewegliche Objekte auf dem Bildschirm darzustellen. Diese 8 Objekte lassen sich in zwei Gruppen einteilen : 4 Missiles und 4 Players. Wie der Name Missile schon dsagt, werden diese vornehmlich als Geschosse in Spielen eingesetzt. Die Players werden meist eingesetzt, um Raumschiffe, Panzer usw. darzustellen. Der Vorteil dieser Graphik besteht darin, dass eine ähnliche Darstellung von Objekten bei herkömmlichen Computern nur in Maschinensprache möglich ist. Der Atari ermöglicht es aber, mit einfachen Basicbefehlen ein Programm zu schreiben, was Besitzern von anderen Computern nur nach umfangreichem Studium der Maschinenspreache möglich ist. Wenn Sie diesen Artikel nun aufmerksam durchlesen, dann werden Sie nach einiger Übung in der Lage sein, Programme zu schreiben, di auch den Vergleich mit Automatenspielen nicht zu scheuen brauchen.

Allerdings etwas Übung gehört schon dazu. Dies gilt auch für die anderen Artikel in diesem Buch. Es ist unmöglich, auf sämtliche Möglichkeiten einzugehen, denn würde jeder Artikel (Scrolling, Displaylist, Player Missile Graphik) den Umfang dieses Buches haben. Deswegen ist es unerlässlich, dass Sie selbst mit den Programmen herumprobieren. Löschen Sie einzelne Zeilen, oder ändern Sie die Werte in den vorgegebenen Programmen, und schauen Sie was passiert. Dadurch lernt man besser und schneller, als beim Durchlesen eines dicken Artikels.

Doch zurück zu Player Missile Graphik : Wie gesagt, stehen uns also 4 Geschosse (Missiles) und 4 Objekt (Players) zu Verfügung. Diese können unabhängig voneinander auf dem Bildschirm bewegt werden. Leider gibt es beim Atari einige Einschränkungen:

- Jeder Player und jedes Missile kann jeweils nur eine Farbe annehmen. Wenn man also mehrfarbige Objekte haben will, muss man mehrere verschiedenfarbige Player oder Missiles überlagern.
- Ein Player kann nur 8 Pixel breit sein. Für grössere Objekte, oder wenn eine grössere Auflösung gewünscht ist, müssen wir auch hier kombinieren.

Doch diese Einschränkungen sind zumindest am Anfang nicht weiter störend. Später werden Sie selbst auf Abhilfe sinnen können.

Die nun immer benützte Abkürzung ‚PM' steht für Player - Missile Graphik'. Da die PM ein Hardware Register ist, brauchen Sie sich nicht zu wundern, wenn im Folgenden eine Speicherzelle verschiedene Bedeutung hat.

Die nun folgende Zeile wird am Anfang eines Programms mit PM ausgeführt. Vorher muss allerdings die gewünschte Graphikstufe gewählt werden.

```
10 A=PEEK(106)-8:PM=256*A:POKE 54279,A:POKE 559,46:POKE 53277,3:
     POKE 623,1
```

Erklärung der obigen Programmzeile :

A=PEEK(106)-8 Bereitstellen des benötigten Speicherplatzes für PM

PM=256*A In der Variablen PM steht nun der Anfang des bereitgestellten Speicherplatzes

POKE 54279,A Dies sagt dem Atari, ab welchem Speicherbereich die PM anfängt.

POKE 559,46 Double Line Resolution (d.h. eine Playerzeile ist 2 Graphics 8 Zeilen hoch) bei 62 : Single Line Resolution (eine Playerzeile ist 1 bei Graphics 8 Zeile hoch : also doppelte Auflösung in der Höhe)

POKE 53277,3 PM einschalten

POKE 623,X X ist zu ändern je nach gewünschter Priorität :

X=1 : alle Player haben Priorität über alle Spielfeldregister

X=2 : Player 0, Player 1, alle Spielfelder, Player2, Player 3

X=4 : alle Spielfelder haben Vorrang vor allen Playern

X=8 : Spielf.0, Spielf.1, alle Player, Spielf.2, Spielf.3

Sie bestimmen hiermit, ob ein Player vor oder hinter einem gezeichneten Objekt zu sehen sein soll. Ein Spielfeld ist gleichbedeutend mit den Farben eines gezeichneten Bildes. Am besten probieren Sie dies nun aus, indem Sie alle 4 Player auf einen Graphics 7 Bildschirm bringen. Dort sollten dann einige Objekte mit verschiedenen COLOR Befehlen gezeichnet werden. Bei der Bewegung der Player werden Sie feststellen, dass je nach Priorität eine dreidimensionale Darstellung möglich ist.

Wie bereits gesagt, dient der erste Befehl in Zeile 10 dazu, den benötigten Speicherplatz freizuhalten. Damit Sie bei anderen Graphikstufen keine bunten Streifen auf den Bildschirm bekommen, muss bei höheren Grafikstufen mehr Speicherplatz freigehalten werden. Die benötigten Werte entnehmen Sie bitte aus der folgenden Tabelle.

Wenn Sie also die PM in der Graphicsstufe 7 anwenden wollen (in Double Line Res.), muss der erste Befehl wie folgt lauten :

```
10 A=PEEK(106)-24: .....
```

Dies gilt nur, wenn die PM voll ausgenutzt wird. Falls weniger als vier Player benötigt werden, kann diese Zahl auch entsprechend verkleinert werden.

Die folgende Grafik verdeutlicht die RAM Aufteilung in der PM

Um den am Anfang noch mit eventuel unerwünschten Daten gefüllten Speicher zu löschen, muss als zweite Zeile folgendes stehen :

20 FOR I=PM+348 TO PM+2048:POKE I,0:NEXT I

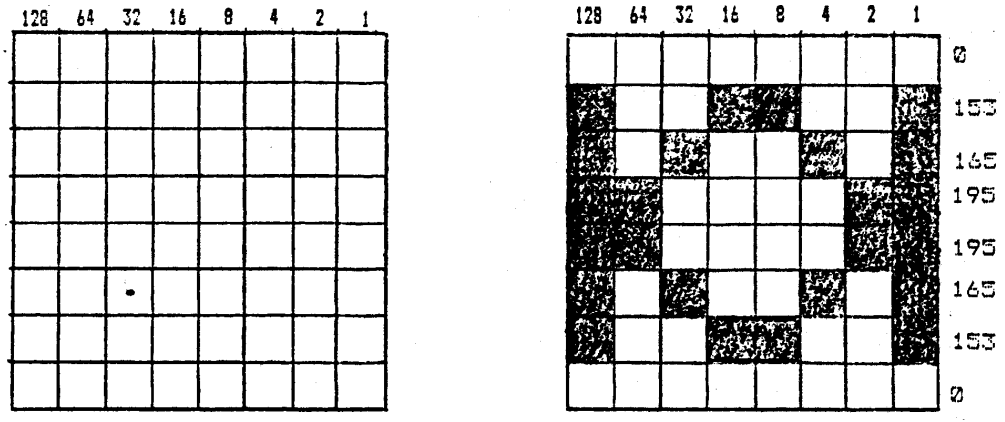
Bei double Line Res. Muss der Speicher selbstverständlich nur bis PM+1024 gelöscht werden.

Wie bereits erwähnt, bietet die PM die Möglichkeit, vier verschiedene Figuren (Olayer 0 bis Player 3) und vier verschiedene Geschosse (Missile 0 bis Missile 3) zu bewegen und eventuelle Zusammenstöße untereinander zu registrieren.

Nun wollen wir aber anfangen, einen Player darzustellen. Ein Player ist 8 Zeichen breit und 128 Zeichen hoch (bzw. 255 bei single Line Res.)

Wollen wir einen Player als Ufo darstellen, machen wir folgendes:

1. Wir zeichnen uns eine 8*8 Matrix und
2. Zeichnen dort das Bild des Ufo’s ein:



3. Nun addieren wir einfach die Zahlen in einer Reihe und Poken diese Zahl in die dazugehörige Speicherstelle. Wir schauen nun in der Tabelle nach un entnehmen aus dieser, dass Player 0 bei PM+512 anfängt. Wir setzen nun die gesamten Horizontalwerte des Player’s in eine DATA Zeile z.B.

50 DATA 0,153,165,195,195,165,153,0

Jetzt können wir diese Werte in die Speicherstellen des Player 0 einlesen. Da sich PM+512 aber ganz oben am Rand des Bildschirmes befindet, addieren wir dort noch eine entsprechende Y-Position zu. Sagen wir mal, Y soll 58 betragen. Dann sieht die Zeile 60 wie folgt aus:

60 FOR I=1 TO 8:READ A:POKE PM+512+58+I,A:NEXT I

Wenn ein Player höher sein soll, dann können natürlich auch mehr Datas eingelesen werden.

Wenn wir das Programm bis hier laufen lassen, tut sich nichts. Das liegt daran, dass die X-Position ja noch 0 ist. Um auf dem Bildschirm zu erscheinen, muss die X-Position zwischen ca. 50 und ca. 200 liegen. Die X-Register für die Player sind wie folgt:

Player	X-Register
0	53248
1	53249
2	53250
3	53251

Um Player 0 jetzt sehen zu können, geben wir folgende Zeile ein :

70 POKE 53248,150: REM Die X Position beträgt hier 150

Jetzt sehen wir das Ufo ganz deutlich auf dem Bildschirm. Leider ist es aber schwarz. Die Player können aber auch farbig dargestellt werden. Dies geschieht so:

Gewünschte Farbe = Colornummer (0-15) * 16 + Helligkeit (0-14)

Wobei Colornummer und Helligkeit genauso gehandhabt werden wie beim SETCOLOR Befehl. Diese Zahl kann nun in das Farbregister von Player 0 gepoked werden. Das Farbregister von Player 0 ist die Speicherzelle 704. Also :

80 POKE 704,X : REM X ist die gewünschte Farbe

Wenn Sie für X 186 einsetzen, wird das Ufo hellgrün :

(11(grün)*16+10(Helligkeit))=186

Wenn Sie das Programm bis hier haben laufen lassen muss sich ein grünes UFO auf dem Bildschirm zeigen. Wenn nicht, können Sie jetzt auf Fehlersuche gehen.

Versuchen Sie nun, alle 4 Player auf verschiedenen X und Y Positionen darzustellen (unterschiedliches Aussehen der Player's) und diese mit dem Steuerknüppel horizontal zu bewegen. Falls Sie es nicht schaffen sollten : Die Lösung ist in Programm 1 zu finden. Aber versuchen Sie es erst selbst !!!

Leider ist das Bewegen auf der Y-Achse nicht so einfach wie die X-Bewegung. Um z.B. das UFO um eine Zeile nach oben zu bewegen, muss es gänzlich gelöscht werden. Das dies relativ lange dauert, braucht wohl nicht weiter erwähnt zu werden.

Um dieses Problem zu umgehen, haben wir in das Programm 2 einen Programmrumpf aufgebaut, der es Ihnen ermöglichen wird, durch einen Poke die Player und die Missiles auf eine bestimmte Y-Pos. zu bringen. Da dies in Maschinensprache geschieht, darf dort nichts verändert werden. Sie brauchen bei Ihren Programmen nur die Zeilen 32000 bis 32500 übernehmen. Dort springen Sie dann nach dem Programmstart einmal hin, um Player Missile zu initialisieren. Dort wird alles für Sie erledigt, So unter anderem auch das Einschalten der Player Missile usw. Änderungen müssen nur dort erfolgen, wo Sie erwünscht werden.

In Zeile 32030 muss, je nach Graphikstufe, ein anderer Speicherbedarf kalkuliert werden.

In Zeile 32080 werden die Farben festgelegt.

Ab Zeile 32250 stehen die Daten für die einzelnen Players.

Die anderen Zeilen müssen mehr oder weniger so übernommen werden.

Nach der Initialisierung werden die Player mit dem Kommando POKE PLX oder POKE PLY horizontal bzw. vertikal bewegt. Wobei der Player 0 mit dem Kommando POKE PLX/PLY bewegt wird, Player 1 mit POKE PLX+1/PLY+1, Player 2 mit POKE PLX+2/PLY+2 usw.

In die jeweilige Speicherzelle wird der gewünschte X und Y Wert geschrieben. Diehe als Beispiel hierzu Programm 3.

Eine weitere Besonderheit von diesem Programm ist, dass es Ihnen ermöglicht wird, das Aussehen der einzelnen Player rasch und unkompliziert zu ändern. Dazu müssen vorher die benötigten Shapes abgespeichert werden. Später im Programm werden sie nur noch abgerufen. Doch schauen wir uns an, wie das Ganze funktioniert : Wie wir bereits gesehen haben, liegt der Speicherbereich der Player Missile Graphik von PM bis PM+2048. Wie aus der Tabelle zu entnehmen ist, bleibt der Bereich von PM bis PM+1023 unbenutzt, d.h. genau 1000 Byte verschenkter Speicherplatz. In diesem Bereich speichern wir jetzt die Bilder für die verschiedenen Player ab. Wo dies genau geschieht, zeigt uns die folgende Tabelle (Abb. 1).

PMBASE	Player 0 - Image
+ 256	Player 1 - Image
+ 512	Player 2 - Image
+ 768	Player 3 - Image
+ 1024 bis 2048	oder Missiles Player 0 bis 3

Abb. 1

PMBASE	- Bild 1	- POKE PDR,1
+ 8	- Bild 2	- POKE PDR,9
+16	- Bild 3	- POKE PDR,17
+24	- Bild 4	- POKE PDR,25
+32	- Bild 5	- POKE PDR,33

Abb. 2

Aus dieser Tabelle ist zu ersehen, dass für einen Player max. 255 Byte (d.h. 255 Zeilen) zur Verfügung stehen. Dabei ist es unerheblich, wieviele verschiedene Bilder sich in 255 Bytes befinden. Man kann einen Player 120 Zeilen hoch machen und hat dann Platz für zwei verschiedene Bilder. Ebenso kann man einen Player nur 10 Zeilen hoch machen, dann ist Platz für 25 verschiedene Bilder, die je nach Bedarf ausgewählt werden können. Bei Player 0 werden diese Bilder mit POKE PDR,X ausgewählt. X ist hierbei das jeweilige Bild, multipliziert mit der Bildhöhe; z.B. : Es sind 5 Bilder mit jeweils einer Höhe von 8 Zeilen gespeichert. Dann erfolgt der Abruf wie in Abb.2 zu sehen.

Zu beachten ist noch folgendes : die Bilder eines Players müssen immer gleich hoch sein. Die Höhe hierfür wird beim Programmstart , in PLL abgespeichert. Auch hier verweise ich auf das Programm #3.
Mit den POKE’s 53256 (Pl.0) bis 53259 (Pl.3) kann die Grösse der einzelnen Player geändert werden. Dies geschieht wie folgt :

Gepokte Zahl	Grösse
0	Normal
1	Doppelt
3	Dreifach

Nun möchten wir aber nicht nur die Player bewegen, sondern auch Schüsse o.ä. abgeben. Diese Schüsse heissen Missiles. Es gibt 4 Stück davon. Sie haben die gleiche Farbe wie der dazugehörige Player. Missile 2 hat also die gleiche Farbe wie die, die für Player 2 ausgewählt wurde. Die Bewegung erfolgt ähnlich wie bei den Player’s. Die X Bewegung wird mit den POKE’s 53252 (Miss. 0) bis 53255 (Miss.3) ausgeführt.
Wie aus der folgenden Tabelle zu ersehen ist, liegen die Missiles im RAM Bereich von PM+768 bis PM+1023. In diesen Speicherpositionen liegen 4 Missiles. Ja nachdem welche Zahl man in diese Zeilen schreibt, wird das jeweilige Geschoss bewegt.

Missile 0 – 1

Missile 1 - 4

Missile 2 - 16

Missile 3 - 64

Die Bewegung von Missile 0 wird wie folgt gehandhabt.
mit POKE 53252,X die X-Position (ca. 50 bis ca. 200)
und die Y-Position mit z.B. **FOR I=0 TO 255:POKE PM+768+I,0:POKE PM+768+I+1,1:NEXT I**
Anstatt der gepokten 1 können die anderen Missiles stehen (also 4, 16, 64). Um diese horizontal zu bewegen, muss natürlich deren X- Register geändert werden.

Nun müssen wir natürlich auch noch feststellen können, ob ein Schuss getroffen hat, ob ein Player oder ein Missile ein gezeichnetes Objekt (Spielfeld) berührt hat, ob zwei Player zusammengestossen sind .....

Dafür gibt es die Kollisionsregister. In den nachfolgenden Tabellen können Sie sehen, mit welchen PEEK welche Zusammenstösse registriert werden können. Normalerweise sind alle Kollisionsregister 0. Bei Berührung der ihnen zugeordneten Objekte ändern sich die Werte in den K.-Registern. Damit eine kurzzeitige Berührung nicht aus Versehen übersehen wird, erfolgt keine automatische Löschung der Register. Dies muss man nach einer Berührung selbst machen. Dies geschieht mit POKE 53278,255.

So, nachdem dieses Thema nun ausführlich behandelt wurde, hoffe Ich, dass Sie mit dem Programm #3 zurecht kommen. Wo Sie etwas nicht verstehen, ändern Sie die entsprechenden Programmzeilen und probieren Sie herum.

	Player				
Player	0	1	2	3	Peek
0	X	2	4	8	53260
1	1	X	4	8	53261
2	1	2	X	8	53262
3	1	2	4	X	53262

Kollisionsregister :
Player - Player

	Player				
Missile	0	1	2	3	Peek
0	1	2	4	8	53256
1	1	2	4	8	53257
2	1	2	4	8	53258
3	1	2	4	8	53259

Kollisionsregister :
Missile - Player

	Color			
Player	1	2	3	Peek
0	1	2	4	53252
1	1	2	4	53253
2	1	2	4	53254
3	1	2	4	53255

Kollisionsregister :
Spielfeld (Color) - Player

#1

```
10 GRAPHICS 0:SETCOLOR 2,0,0
11 POKE 752,1:? „Ü“:REM CURSOR AUS
20 A=PEEK(106)-8:PM=A*256:POKE 53277,3:POKE 559,46:POKE 54279,A:R
EM PM EINSCHALTEN
30 FOR I=PM+384 TO PM+1024:POKE I,0:NEXT I:REM SPEICHER CLEAREN
40 Y0=20:Y1=50:Y2=80:Y3=110:REM Y POSITION DER 4 PLAYER
50 FOR I=1 TO 8:READ A:POKE PM+512+Y0+I,A:NEXT I:REM PLAYER 0
60 FOR I=1 TO 8:READ A:POKE PM+640+Y1+I,A:NEXT I:REM PLAYER 1
70 FOR I=1 TO 8:READ A:POKE PM+768+Y2+I,A:NEXT I:REM PLAYER 2
80 FOR I=1 TO 8:READ A:POKE PM+896+Y3+I,A:NEXT I:REM PLAYER 3
89 REM DATAS DER PLAYER 0-3
90 DATA 0,153,165,195,195,165,153,0
100 DATA 170,85,170,85,170,85,170,85
110 DATA 170,170,170,170,170,170,170,170
120 DATA 255,129,129,129,129,129,129,255
150 POKE 704,31:POKE 705,62:POKE 706,93:POKE 707,124:REM FARBEN D
ERPLAYER 0-3
160 X0=60:X1=80:X2=100:X3=120:REM X POSITIONEN DER 4 SPIELER
170 S=STICK(0):REM STEUERKNUEPPEL
171 TRAP 300
180 IF S=11 THEN X0=X0-2.5:X1=X1-2:X2=X2-1.5:X3=X3-1:REM LINKS
190 IF S=7 THEN X0=X0+2.5:X1=X1+2:X2=X2+1.5:X3=X3+1:REM RECHTS
200 POKE 53248,X0:POKE 53249,X1:POKE 53250,X2:POKE 53251,X3:REM X
-POSITIONEN DER 4 PLAYER POKEN
210 GOTO 170
300 FOR I=1 TO 300:NEXT I:? „ü“:GOTO 160
```

#2

```
32000 REM
32010 FOR I=1536 TO 1706:READ A:POKE I,A:NEXT I
32020 FOR I=1774 TO 1787:POKE I,0:NEXT I
32030 PM=PEEK(106)-16:PMBASE=256*PM
32040 FOR I=PMBASE+1023 TO PMBASE+2047:POKE I,0:NEXT I
32050 FOR I=0 TO 3:DRW=PMBASE+I*256+1
32060 RESTORE 32250:FOR J=0 TO 3
32070 FOR K=DRW+J*24 TO DRW+J*24+23:READ X:POKE K,X:NEXT K:NEXT J
:NEXT I
32080 POKE 704,12:POKE 705,128:POKE 706,48:POKE 707,192
32090 PLX=53248:PLY=1780:PLL=1784
32100 POKE 559,62:POKE 623,1:POKE 1788,PM+4:POKE 53277,3:POKE 542
79,PM
32110 PDR=1772:POKE 1771,PM
32120 X=USR(1696)
32140 DATA 162,3,189,244,6,240,89,56,221,240,6,240,83,141,254,6,1
06,141
32150 DATA 255,6,142,253,6,24,169,0,109,253,6,24,109,252,6,133,20
4,133
32160 DATA 206,189,240,6,133,203,173,254,6,133,205,189,248,6,170,
232,46,255
32170 DATA 6,144,16,168,177,203,145,205,169,0,145,203,136,202,228
,244,76,87
32180 DATA 6,160,0,177,203,145,205,169,0,145,203,200,202,208,244,
174,253,6
```

```
32190  DATA  173,254,6,257,240,6,189,236,6,240,48,133,203,24,138,14
1,253,6
32200  DATA  109,235,6,133,204,24,173,253,6,109,252,6,133,206,189,2
40,6,133
32210  DATA  205,189,248,6,170,160,0,177,203,145,205,200,202,208,24
8,174,253,6
32220 DATA 169,0,157,136,6,202,48,3,76,2,6,76,98,229,0,0,104,169
32230 DATA 7,162,6,160,0,32,92,228,96
32240 REM
32250 DATA 0,12,12,30,0,12,12,0,12,14,30,45,13,13,12,28,28,20,52,
34,34,34,102,0
32260  DATA  0,12,12,30,0,12,12,0,12,14,14,13,26,4,8,12,12,28,24,28
,12,8,24,0
32270  DATA  0,12,12,30,0,12,12,0,12,14,10,14,30,12,8,12,28,28,8,12
,12,8,24,0
32280  DATA  0,12,12,30,0,12,12,0,12,12,12,10,6,30,12,12,12,12,20,2
0,20,18,50,6,0
```

3

```
10 GRAPHICS 7+16:FOR I=0 TO 30:COLOR RND((0)*3+1:PLOT RND(0)*150,R
ND(0)*90:NEXT I
30 GOSUB 32000
31 X=100:J=100:S=1
32 POKE 53256,1
33 POKE PLY+1,222:POKE PDR+1,1
34 POKE 53278,255
40 FOR I=215 TO 30 STEP -3:REM UFO BEWEGUNG AUF DER X-ACHSE
41 IF PEEK(53256)=1 THEN 1000:REM TREFFERABFRAGE
42 IF PEEK(53257)=2 THEN 1100:REM TREFFERABFRAGE
44 POKE 53278,255:REM KOLLISIONSREGISTER LOESCHEN
50 POKE PLX,I
51 A=RND(0):IF A0.5 THEN J=J+1:GOTO 55
52 J=J-1:IF J<30 THEN J=30
55 IF J>120 THEN J=120
56 POKE PLY,J:REM UFO BEWEGUNG AUF DER Y-ACHSE
60 S=S+8:IF S47 THEN S=1
70  POKE  PDR,S:SOUND  0,S*2,10,S/4:REM  AUSWAHL  DES  JEWEILIGEN  UFO
BILDES
80 IF SCH=1 THEN 95:REM EIN SCHUSS DES UFO'S IST BEREITS UNTERWEG
S
82 IF RND(0)<0.9 THEN 100:REM KEIN NEUERUFO SCHUSS
84 POKE 53253,I+3:MMY=PMBASE+768+J
86 SCH=1:REM NEUER UFO SCHUSS
95 POKE MMY,0:REM UFO SCHUSS BEWEGUNG
100 ST=STICK(0):TR =STRIG(0):REM ABFRAGE JOYSTICK UND TRIGGER
110 IF ST=7 THEN X=X+3:IF X>200 THEN X=200:REM BASIS NACH RECHTS
120 IF ST=11 THEN X=X-3:IF X<50 THEN X=50:REM BASIS NACH LINKS
130 POKE PLX+1,X:REM NEUE BASISSTATION
140 IF SC=1 THEN 161:REM BASIS SCHUSS BEREITS UNTERWEGS
150  IF  TR=1  THEN  MY=PMBASE+768+220:GOTO  200:REM  KEIN  NEUER  BASIS
SCHUSS
```

```
155 SC=1:REM NEUER BASIS SCHUSS
160 MX=X+4:POKE 53252,MX:REM X-POSITION DES MISSILES (BASIS)
161 MX=X+4:POKE MY,0:MY=MY-6:POKE MY,1:POKE MY-6,1:POKE MY-12,1:S
OUND 1,MY,8,10:REM BEWEGUNG DES BASIS SCHUSS
162 IF MY-(PMBASE+768)33 THEN SC=0:POKE MY,0:POKE MY-6,0:POKE MY
-12,0:SOUND 1,0,0,0:REM SCHUSS AUSSERHALB SCREEN?
200 NEXT I
300 GOTO 40
1000 REM UFO GETROFFEN
1010 FOR I=60 TO 0 STEP -1:SOUND 0,I*4,8,I/5:POKE 704,RND(0)*255:
NEXT I:POKE 704,12:POKE 53278,255
1020 POKE PLX,0:GOTO 40
1100 REM BASIS GETROFFEN
1110 FOR I=60 TO 0 STEP -1:SOUND 0,I*4,8,I/5:POKE 705,RND(0)*255:
NEXT I:POKE 705,122:POKE 53278,255
1120 POKE PLX+1,50:GOTO 40
32000 REM
32010 FOR I=1536 TO 1706:READ A:POKE I,A:NEXT I
32020 FOR I=1774 TO 1787: POKE I,0:NEXT I
32030 PM=PEEK(106)-32:PMBASE+2047:POKE I,0:NEXT I
32050 DRW=PMBASE+1
32060 RESTORE 32250:REM UFO DATAS
32070 FOR K=DRW TO DRW+6*8:READ X:POKE K,X:NEXT K:REM EINLESEN DE
R UFO DATAS
32072 DRW=PMBASE+255+1:RESTORE 32400:REM BASIS DATAS
32074 FOR K=DRW TO DRW+6:READ X:POKE K,X:NEXT K:REM EINLESEN DER
BASIS DATAS
32080 POKE 704,12:POKE 705,122:REM FARBEN FESTLEGEN
32090 PLX=53248:PLY=1780:PLL=1784
32100 POKE 559,62:POKE 623,1:POKE PLL,8:REM DA JEWEILS 8 UFO DATEN
32111 POKE PLL+1,6:REM DA 6 BASIS DATEN
32120 X=USR(1696)
32130 RETURN
32140 DATA 162,3,189,244,6,240,89,56,221,240,6,240,83,141,254,6,1
06,141
32150 DATA 255,6,142,253,6,24,169,0,109,253,6,24,109,252,6,133,20
4,133
32160 DATA 206,189,240,6,133,203,173,254,6,133,205,189,248,6,170,
232,46,255
32170 DATA 6,144,16,168,177,203,145,205,169,0,145,203,136,202,208
,244,76,87
32180 DATA 6,160,0,177,203,145,205,169,0,145,203,200,202,208,244,
174,253,6
32190 DATA 173,254,6,157,240,6,189,236,6,240,48,133,203,24,138,14
1,253,6
32200 DATA 109,235,6,133,204,24,173,253,6,109,252,6,133,206,189,2
40,6,133
32210 DATA 205,189,248,6,170,160,0,177,203,145,205,200,202,208,24
8,174,253,6
32220 DATA 169,0,157,236,6,202,48,3,76,2,6,76,98,228,0,0,104,169
32240 REM 6 UFO BILDER
32250 DATA 0,8,28,34,65,34,28,8
32250 DATA 0,8,28,34,67,34,28,8
32250 DATA 0,8,28,34,69,34,28,8
32250 DATA 0,8,28,34,73,34,28,8
32250 DATA 0,8,28,34,81,34,28,8
32250 DATA 0,8,28,34,97,34,28,8
32390 REM 1 BASIS BILD
32400 DATA 8,8,8,28,62,73,65
```

Antic und die Displaylist

Eine Einführung in die Graphikmöglichkeiten des Atari

Teil 1

„Antic“ ist die Bezeichnung für den Graphikprozessor des Atari.
Er steuert alles, was zum Aufbau des Fernsehbildes benötigt wird.
Antic ist ein richtiggehender, programmierbarer Microprozessor und verfügt über DMA (Direkt Memory Access, direkter Speicherzugriff), d.h., dass Antic jederzeit, wenn er Daten für den Bildaufbau benötigt, auf den gesamten Speicher des Ataris zugreifen kann. Da die 6502 CPU während dieser Zeit ihrerseits keinen Speicherzugriff durchführen kann, setzt die Arbeit des Graphikprozessors die Verarbeitungsgeschwindigkeit von Programmen herab, was sich mit dem folgenden Programm leicht zeigen lässt :

```
10 GRAPHICS 0
20 POKE 559,34
30 POKE 19,0:POKE 20,0
40 FOR I=0 TO 5000:NEXT I
50 TIME=PEEK(20)+256*PEEK(19)
60 GRAPHICS 0
70 PRINT „Zeit=“;TIME
```

Lassen Sie das Programm durchlaufen und merken Sie sich die Zeit, die angezeigt wird. Ändern Sie nun die Zeile 20 in:

```
20 POKE 559,0
```

Jetzt lassen Sie das Programm erneut ablaufen. Der Bildschirm wird jetzt zwar dunkel, jedoch sollte die Laufzeit um ca. 30 % kürzer sein.
Anstatt dessen, können Sie aber auch Zeile 10 in jeden anderen Graphikmode abändern und einen Vergleich des Zeitbedarfs der einzelnen Graphikmodis anstellen.

Probieren Sie auch mal den Unterschied zwischen einem leeren und mit beliebigen Zeichen gefüllten Bildschirm aus. Der Unterschied ist beträchtlich !

Auch Player Missile Graphik kostet Zeit ! Im ungünstigsten Fall kann sich die Ausführungszeit von Programmen um über 50 % erhöhen. (Grosses Bildfenster + hochauflösende Graphik + gefüllter Bildschirm + PM Graphik)

Da man aber nicht immer alle diese Dinge auf einmal benötigt, kann man sie je nach Bedarf ein- und ausschalten.

Hierzu dient nun die bereits erwähnte Speicherzelle 559 ($022F – SDMCTL). Sie ist das Shadow-(Schatten)register für die Speicherstelle 54272 ($D400 – DMACTL). Als Shadow Register bezeichnet man Speicherstellen im RAM Bereich, deren Inhalt vom Atari automatisch jede 50-tel Sekunde in das entsprechende Register eines der Peripheriebausteine des Ataris übertragen wird.

Für uns bedeutet das, dass wir anstatt in559 auch in 54272 poken können, allerdings mit dem Erfolg, dass das von uns gewünschte Ergebnis nur max. eine 50-tel Sekunde anhält.

Was kann man nun alles mit dieser Speicherstelle anfangen ?
Darüber gibt uns die Tabelle auf der nächsten Seite Auskunft :

Dezimalwert	Bit	Funktion	
0-3	0-1	0- kein Bildfenster	Zeichen / Zeile 0
		1- kleines Bildfenster	32
		2- normales Bildfenster	40
		3- grosses Bildfenster	48
4	2	Missiles ein- und ausschalten	
8	3	Player ein- und ausschalten	
16	4	Einzeilige Auflösung für PM Graphik Ein- ausschalten (ansonsten 2-teilig)	
32	5	Antic ein- ausschalten (norm. Graphik)	
64	6	Nicht benutzt	
128	7	Nicht benutzt	

Beim Einschalten erhält die Speicherstelle den Wert 34(32+2), also normales Bildfenster und eingeschalteter Antic.

Die beiden anderen Bilfdormate werden vom Basic leider nicht unterstützt, richtig verwenden kann man sie nur in Maschinensprache. Wie Sie aus der Tabelle entnehmen können, ist es z.B. auch möglich, nur PM-Graphik zu verwenden (ohne normale Graphik), wenn man es für nötig hält.

Antic und die Displaylist

Eine Einführung in die Graphikmöglichkeiten des Atari

Teil 2 – Die Displaylist

In diesem Teil wollen wir uns damit beschäftigen, wie wir Antic dazu bringen können, Text oder Graphik auf dem Bildschirm darzustellen.

Hierzu wird ein kurzes Programm verwendet : die sogenannte Displaylist.

Diese Methode ist zwar für Computer in dieser Preisplasse nicht üblich, jedoch ist der Atari gerade dadurch vergleichbaren Rechnern graphisch überlegen.

Die Displaylist ist zwar ein Maschinenprogramm, jedoch sind die verwendeten Instruktionen natürlich anders als die 6502 Codes; es sind Spezialbefehle, die speziell auf den Antic-Graphikprozessor zugeschnitten sind.

Die Informationen darüber, wo sich die Displaylist im Speicher befindet, bezieht Antic aus den Speicherstellen 560 ($0230,SDLSTL) und 561 ($0231,SDLSTH). Dies sind die Shadowregister für die Anticregister 54274 ($D402,DLISTL) und 54275 ($D403,DLISTH).

Wenn Sie wissen wollen, wo sich die Displaylist befindet, schreiben Sie : DL=PEEK(560)+256*PEEK(561)

Und um die Displaylist zu aktiv9ieren :

```
POKE 560,DL/256-INT(DL/256)
POKE 561,INT(DL/256)
```

Wobei DL die Adresse Ihrer Displaylist ist.

Im Folgenden wollen wir uns eine Displaylist einmal näher ansehen. Geben Sie hierzu das folgende Programm ein.

```
10 DIM D(32)
20 GRAPHICS 3+16
30 DL=PEEK(560)+256*PEEK(561)
40 FOR I=0 TO 31
50 D(I)=PEEK(DL+I) : NEXT I
60 GRAPHICS 0
70 FOR I=0 to 31
80 PRINT D(I);“ „;: NEXT I
```

Die ersten drei Befehle lauten 112 ($70) und bedeuten jeweils die Ausgabe von 8 leeren Fernsehzeilen. Da das vom Atari dargestellte Fernsehbild über insgesamt 238 Fernsehzeilen verfügt (eine Graphics 8 Zeile entspricht einer Fernsehzeile), waren dieseBefehle nötig, um das Graphics 3+16 Bildfenster, das über 192 Fernsehzeilen verfügt, in die Mitte des Bildschirms zu rücken. Wenn wir diese Befehle weglassen, dann würde unser Fernsehbild am oberen Rand des Fernsehers beginnen. Allen durch Graphics-Befehle generierten Displaylists ist es gemeinsam, dass sie über diese drei Leeranweisungen verfügen und das Bild 192 Fernsehzeilen hoch ist. Doch schauen wir weiter.

Die nächsten drei Werte gehören zusammen und sind eigentlich ein Doppelbefehl. Der erst Wert 72 setzt sich aus 64+8 zusammen. Die 8 bedeutet, dass die erste Bildzeile (bzw. die ersten acht Fernsehzeilen, da eine Graphics 3 Zeile 8 Fernsehzeilen hoch ist) eine Graphics 3 Zeile ist. Sehen Sie sich hierzu auch die Tabelle am Schluss an. Bei einer Graphics 8 Displaylist stände dort 79. Diese setzt sich dann aus 64+15 zusammen, wobei 15 eine Zeile in Graphics 8 bedeutet.

Wie Sie sehen, unterscheiden sich die Graphikcodes der Displaylist von den Graphikbefehlen in Basic. Hier ist Vorsicht geboten.

Die 64 ist eine sogenannte LOAD MEMORY SCAN (LMS) – Anweisung und teilt Antic mit, an welcher Adresse sich die Daten befinden (das Bild), die zur Anzeige gebracht werden sollen. Darauf folgt eben diese Adresse, aufgespalten in nieder- und höherwertiges Byte.

Da Antic nun weiss, wo sich im Speicher die Daten für die Anzeige befinden, wird die LMS Anweisung für die weiteren Zeilen nicht mehr benötigt, es folgen 23 achten. Somit haben wir 24 Graphics 3 Zeilen (Die erste Zeile wird aus der 64+8 gemacht, es folgen 23 weitere).

Die letzten drei Bytes bilden wiederum einen Befehl, nämlich eine Sprungbefehl zum Beginn der Displaylist, da Antic das Bild ja nicht nur einmal, sondern immer wieder aufbauen soll. Die erste Zahl (die 65) sagt der Antic, dass sie auf den VERTICAL INTERRUPT (VBI), der alle 1/50 Sekunde ausgelöst wird, warten soll. Dies dient zur Synchronisation von Antic mit dem Schreibstrahl in der Bildröhre. Ohne dieses Warten würde man kein stehendes Bild erhalten. Bei amerikanischen Geräten tritt der VBI übrigens jede 1/60 Sekunde ein. Dadurch sind diese Computer um ca 20 % schneller. Wer dich also über ein zu schnelles Spiel ärgert, der kann sich damit trösten, dass dieses Spiel in Amerika noch schneller ist.

Die beiden letzten Bytes beinhalten somit die Startadresse der Displaylist, sind also mit den Inhalten der Speicherstelle 560 und 561 identisch.

Hier noch einmal eine schematische Übersicht über den Aufbau der Displaylist, die über Graphics-Befehle aufzurufen ist. (ohne Textfenster)

1. 3 x 8 leere Fernsehzeilen
2. LMS- und Displayanweisung
3. Restliche Displayanweisungen (2+3. Ergeben zusammen 192 Fernsehzeilen.
4. Rücksprunganweisung

Mit Textfenster sieht das Ganze folgendermassen aus :

1. 3 x 8 leere Fernsehzeilen
2. LMS- und Displayanweisung
3. Restliche Displayanweisungen (2.+3. Ergeben zusammen 160 F.z.)
4. LMS-Anweisung und Textdisplayanweisung für Graphics 0 (66)
5. 3 restliche Graphics 0 Textdisplayanw.(4.+5. Zus 32 F.z.)
6. Rücksprungbefehl

Bei der Erstellung eigener Displaylist müssen Sie folgendes beachten :

1. Die Summe der Fernsehzeilen darf 238 nicht überschreiten
2. Bei längeren Displaylists müssten Sie darauf achten, dass die Displaylist eine 1 K Grenze nicht überschreiten kann. Sie können also keine Displaylist von der Adresse 2000 – 2100 verwenden, da 2*1024=2048 eine 1K Grenze ist. Sie können also bei diesem Beispiel erst bei 2048 anfangen und haben Platz bis 3072. Sie können diese Grenze allerdings mit einer Sprunganweisung (ohne VBI ! siehe Tabelle) überqueren.
3. Die Displaydaten können eine 4 K Grenze nicht überschreiten. Wenn die Datenmenge größer als 4 K ist (z.B. für ein Graphics 8 Display) müssen Sie mit LMS Anweisungen arbeiten, um die 4 K Grenze zu überschreiten. Schauen Sie sich einfach eine Graphics 8 Displaylist an. Dann sehen Sie was gemeint ist.
4. In der folgenden Übersicht sind alle weiteren Informationen enthalten, die Sie benötigen, um eigene Displaylists zu erstellen. Ausser den vorher erwähnten Beschränkungen sind Ihrer Phantasie keine Grenzen gesetzt (durch eine Displaylist von 238 Graphics 8 Zeilen und POKE 559,35 können Sie z.B. die Auflösung des Ataris auf den Maximalwert von 384 x 238 = 91392 Pixel steigern !)

Wenn Sie eine eigene Displaylist verwenden wollen, müssen Sie darauf achten, dass der Datenkanal (#6) für den Bildschirm geöffnet ist sonst sind Befehle wie DRAWTO, PLOT, PRINT#6 usw. nicht verwendbar. Dies erreichen Sie durch einen beliebigen Graphik Befehl, der ein Graphikfenster hat. (z.B. GRAPHICS 3). Wenn Sie nun bestimmte Zeilen beschreiben wollen , so poken Sie am besten vorher in die Speicherstelle 87 den entsprechenden Graphicswert.
Wenn Sie beispielsweise mitten in einem Block von GR.7 Zeilen eine Textzeile haben (z.B. GR.1), so erstellen Sie die Graphik durch POKE 87,7 und verwenden anschließend PLOT und DRAWTO Befehle . Den Text schreiben Sie nach POKE 87,1 mit PRINT #6 – Befehlen.
Schwierigkeiten kann es auch unter Umständen mit dem POSITION Befehl geben. Durch POKE 87,8 können Sie zwar die ärgerlichen ERROR 141 (Cursor out of Range) vermeiden, sofern Ihre Displaylist nicht mehr als 192 Zeilen umfasst. Allerdings können Verschiebungen auftreten, da mehrere Graphik Modi 20 oder nur 10 Bytes pro Zeile statt der üblichen 40 anzeigen. Hier hilft nur Nachrechnen oder Ausprobieren.
Wenn Sie eigene Displaylists erstellen wollen, gehen Sie am besten so vor :
1. Legen Sie einen Speicherbereich fest, wo Ihre neue DL liegen soll. Die PAGE 6 ist ein guter Platz hierfür (dez. 1536-1791)
2. Über legen Sie, wieviel Zeile Sie in welchem Graphik-Modi darstellen wollen, und notieren Sie sich die genaue Abfolge.(Auf die Anzahl der Fernsehzeilen achten !!)
3. Berechnen Sie aus den Ausgaben in der dritten Spalte der übersicht den Speicherplatz, den Sie für den Bildspeicher benötigen. Wenn Sie diesen Wert von der höchsten verwendbaren RAM-Adresse (bei einem 48 K Rechner mit Basic Cartridge ist dies 40960) ab. Ziehen, erhalten Sie die Adresse, wo später Ihr Bildspeicher be-

ginnt.Jede andere, niedrigere Adresse wäre natürlich auch möglich. Diese Adresse verwenden Sie später bei der LMS Anweisung, ausserdem müssen Sie sie in die Speicherstellen 88 un 89 poken, damit das Betriebssystem weiss, wo die Bilddaten liegen. Achten Sie darauf, dass 4 K-Grenzen durch eine zusätzliche LMS – Anweisung umgangen werden müssen.

Wenn Sie ausserdem noch die normalen PRINT Befehle verwenden wollen, müssen Sie die Adresse auch noch in die Speicherstellen 660 und 661 einpoken. Dies funktioniert allerdings nur dann, wenn Sie vorher eine Displaylist mit Textfenster hatten.

In den meisten Fällen ist es jedoch einfacher, bestehende Displaylisten zu verändern. Man spart sich hierbei eine Menge Arbeit.

	Antic Mode	Basic Mode	Bytes pro Zeile	Pixel pro Zeile	Anzahl der Farben	Anzahl TV-Zeilen
Text	2	0	40	40	1.5	8
	3	-	40	40	1.5	10
	4	(12)	40	40	4	8
	5	(13)	40	40	4	16
	6	1	20	20	5	8
	7	2	20	20	5	16
Grafik	8	3	10	40	4	8
	9	4	10	80	2	4
	10	5	20	80	4	4
	11	6	20	160	2	2
	12	(14)	20	160	2	1
	13	7	40	160	4	1
	14	(15)	40	160	4	1
	15	8	40	320	1.5	1

Leerzeilen - Befehle

0 – 1 Leerzeile
16 – 2 Leerzeilen
32 – 3 Leerzeilen
48 – 4 Leerzeilen
64 – 5 Leerzeilen
80 – 6 Leerzeilen
96 – 7 Leerzeilen
128 – 8 Leerzeilen

Sprung – Befehle

1 – Sprung zur folgenden Adresse

65 – wie 1 aber mit Warten auf VBI

Optionen für Display – Befehle

\+ 16 – Horizontales Scrollen
\+ 32 – Vertikales Scrollen
\+ 64 – Load Memory Scan
\+ 128 – Displaylist Interrupt

Programmerklärung

Das untenstehende Programm ist eine Demonstration dessen, was man mit Displaylisténderung alles anfangen kann. In diesem Demo wurde diesmal nicht eine neue Displaylist erstellt, sondern einfach die schon bestehende geändert. Diese geschieht in den Zeilen 30 und 60 bis 63. Man beachte, dass jeweils immer die Speicherzelle 87 geändert wird. Zeile 70 + 80 verdient besondere Aufmerksamkeit : Wenn Sie versuchen, nach POKE 87,0, auf den unteren Teil des Graphics 7 Screens zu schreiben, gibt der Atari ERROR 141 aus. Wann der Computer denkt, er wäre im Mode 0, erlaubt er auch nur 24 Zeilen zu schreiben. Mehr geht nicht. Deshalb muss der Computer überlistet werden. Man teilt dem OS einfach eine neue Bildanfangsadresse mit. Nun kann man weiter unten anfangen. In diesem Fall haben wir das Bild um 31 Zeilen mit je 40 Byte = 1240 Byte nach unten verschoben. Bei einigen Anwendungen muss hier natürlich ein anderer Wert hinein.

```
10 GRAPHICS 7+16:COLOR 1
11 SETCOLOR 2,0,0
20 DL=PEEK(560)+256*PEEK(561)
30 POKE DL+3,66:REM 1. Zeile graphics 0
40 POKE 87,0:? #6;“        Display List Demoprogramm      “
45 POKE 87,7
50 FOR I=0 TO 159 STEP 31.8:FOR J=0 TO 159 STEP 31.8:PLOT I,1:DRA
WTO J,30:NEXT J:NEXT I
60 POKE DL+36,2:REM Graphics 0
61 POKE DL+37,6:REM Graphics 1
62 POKE DL+38,7:REM Graphics 2
63 POKE DL+39,2:REM Graphics 0
69 REM neuen Bildanfang Simulieren
70 A1=PEEK(88)+256*PEEK(89):A2=A1+1240:H=INT(A2/256):L=(A2/256-H)
*256
80 POKE 88,L:POKE 89,H:POKE 87,0
81 POSITION 0,0
82 POKE 752,1
90 ? #6;“   wir mischen jetzt GRAPHICS 7 mit     „;
100 ? #6;“ GRAPHICS 1 und        graphics 2“
110 ? #6;           und Graphics 0“
120 POKE 87,7:FOR I=0 TO 100:COLOR RND(0)*4+1:PLOT 40,20:DRAWTO R
ND(0)*40+20,RND(0)*40+5
130 PLOT 120,20:DRAWTO RND(0)*40*100,RND(0)*42+5
140 NEXT I
200 FOR I=6 TO 255:SOUND 0,I,10,10:SOUND 1,I-2,10,10:SOUND 2,I-4,
10,10:SOUND 3,I-6,10,10:POKE 708+Q,I:NEXT I
210 Q=Q+1:IF Q=4 THEN Q=0
220 POKE 709,216:POKE 710,0:POKE 711,120:POKE 713,0:GOTO 200:REM
Farben
1000 GOTO 1000
```

Antic und die Displaylist

Eine Einführung in die Graphikmöglichkeiten des Ataris

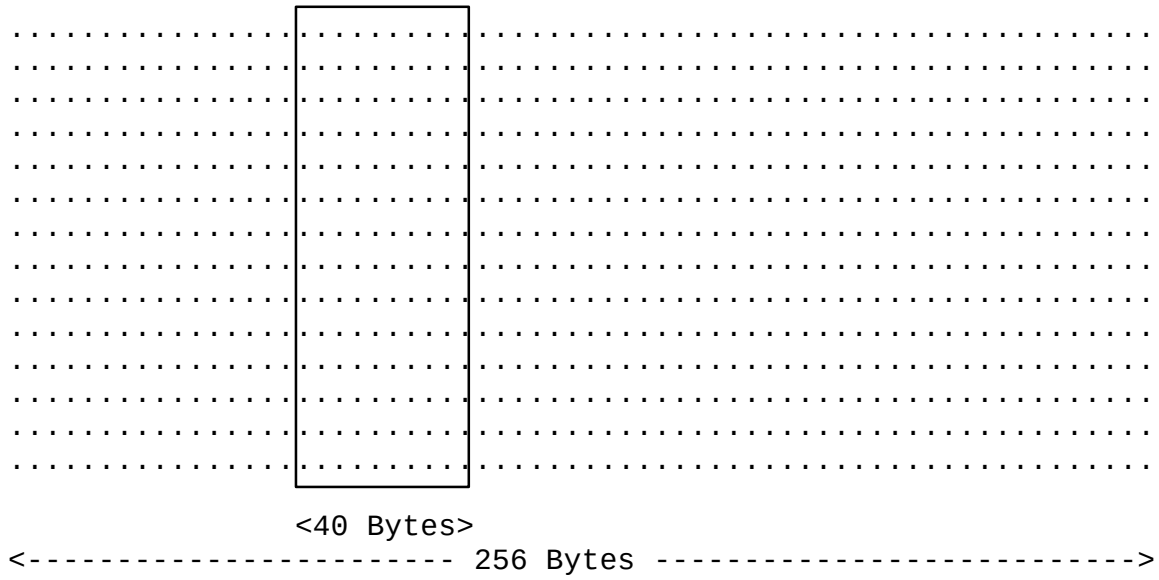
Teil 3 – Scrolling

Grundsätzlich unterscheidet man zwischen zwei verschiedenen Arten von Scrolling : Course Scrolling und Fine Scrolling. Course (Grob – Scrolling) bedeutet, dass der Inhalt des Bildspeichers byteweise verschoben wird, um beim Benutzer den Eindruck kontinuierlicher Bewegung zu erreichen.

Wer schon einmal bei einem herkömmlichen Computer versucht hat einen Defender o.ä. zu schreiben, wird sicherlich zustimmen, dass dies selbst in Maschinensprache eine ziemlich Zeitaufwendige Angelegenheit ist und die Qualität des Bildes, aufgrund der Sprünge in der Bewegung, ziemlich schlecht ist, Das Demoprogramm 1 vermittelt einen ungefähren Eindruck von der Qualität dieser Bilder. Aber es gibt natürlich bessere Möglichkeiten.

Wie wir schon in Teil 2 gesehen haben, bietet der Atari durch Load Memory Scan – Anweisungen (LMS) in der Displaylist die Möglichkeit, die Startadresse der Bilddaten für jede Zeile einzeln festzulegen.

Wenn man nun die Bild-Daten folgendermassen anordnet, ist es möglich, nur durch Verändern der Adresse in der LMS – Anweisung den Bildschirm zu scrollen.



Man schiebt den Bildschirm also praktisch wie ein Fenster über den Speicher.

Der Wert von 256 Bytes ist zwar willkürlich gewählt, jedoch zu empfehlern, da der Programmieraufwand hierbei minimal ist. Das Demoprogramm 2 verwendet diese Methode und zeigt, dass es selbst im ach so langsamen Atari Basic möglich ist, den Bildschirm mit annehmbarer Geschwindigkeit zu scrollen.

Normalerweise würde man natürlich die Scroll – Routine als Maschinenprogramm schreiben, aber hier wurde erst einmal bewusst darauf verzichtet.

Ihnen wird sicherlich aufgefallen sein, dass sich ein Sprung in der Bewegung befindet, wenn die Schleife von vorne beginnt. Dies kann man nur durch eine geschickte Anordnung der Bilddaten vermeiden. Doch dazu später mehr.

Nun zum Fine – Scrolling.

Im obigen Beispiel haben wir den Bildausschnitt immer um ein Zeichen verschoben, welches bekanntlich acht Bildpunkte breit ist. Beim Fine – Scrolling wird nun um jeweils zwei Bildpunkte verschoben, um eine weichere, nicht so ruckartige Bewegung zu erreichen.

Auf gewöhnlichen Computern wäre dies, wenn überhaupt, nur mit komplizierten Umdefinitionen des Zeichensatzes und ähnlich fiesen Tricks möglich.

Eine bitweise Verschiebung einer hochauflösenden Grafikseite, wäre allein durch ihren hohen Rechenaufwand nicht mehr zu vertreten. Glücklicherweise können wir diese Aufgaben völlig Antic überlassen, der intern dazu in der Lage ist, Fine – Scrolling sowohl horizontal, als auch vertikal durchzuführen. Hierzu müssen wir für jede Zeile, die später gescrollt werden soll, das entsprechende Bit in der Displaylistanweisung setzen.

Man kann also selbst bestimmen, welche Zeilen gescrollt werden sollen und welche nicht (z.B. für Score Anzeigen usw.) Nun kann man durch poken in das Antic – Register HSCROL ($D404 – 54276) oder VSCROL ($D405 – 54277) angeben, um wieviele Bildpunkte gescrollt werden soll. Selbstverständlich ist es auch möglich, gleichzeitig horizontal und vertikal zu scrollen (Spiele wie Zaxxon und Blue Max sind gute Beispiele hierfür).

Demoprogramm Nr. 3 zeigt ein Bespiel für die Verwendung der Scrollregister.

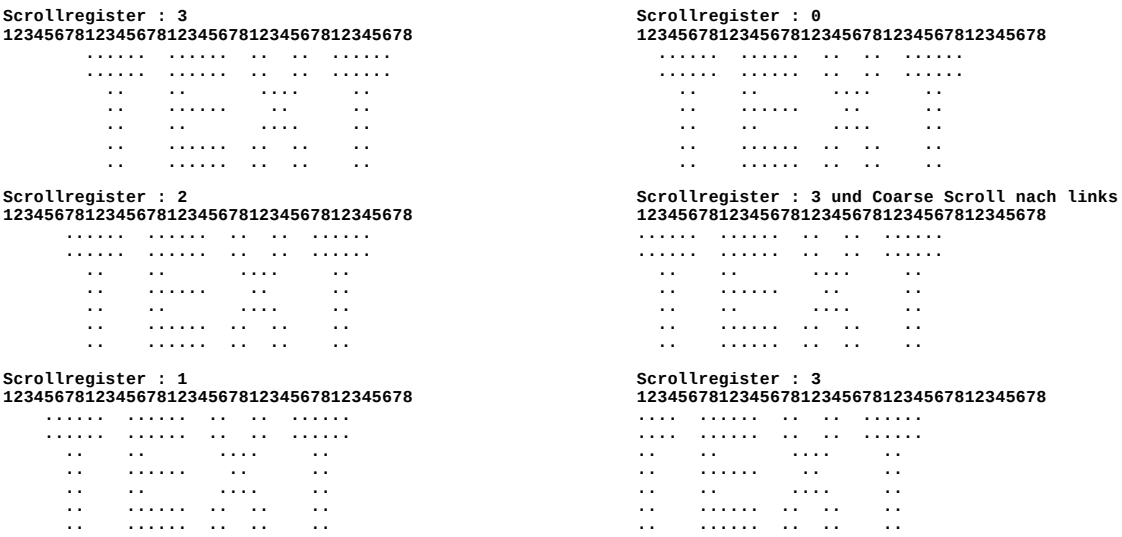
Wichtig sind hierbei nur die unteren vier Bit der Werte, die in diese Register geschrieben werden; die oberen vier werden nicht beachtet.

Somit ergeben sich die Werte von 0 – 15 als mögliche Verschiebungen. 0 ist der normale Wert, jede weitere Zahl verschiebt die Zeile um jeweils 2 Bildpunkte (d.h. 2 Graphik 8 Punkte) nach rechts bzw. nach oben.

Was macht man aber nun, wenn man mehr als 15 Einheiten weit scrollen will? Hier hilft nur eine Kombination von Coarse- und Fine – Scrolling.

Wenn wir also beispielsweise eine Graphics 0 Zeile scrollen wollen, durchlaufen wir erst die Scrollregister 3 bis 0 und führen dann einen Coarse – Scroll aus, der den Beginn der Zeile um ein Byte verschiebt.

Die folgende Graphik soll das verdeutlichen.



Für ein Scrolling nach rechts müsste das Ganze natürlich in umgekehrter Reihenfolge ablaufen.

Hier ist nun auch der Zeitpunkt gekommen, wo man auf den weiteren Gebrauch von Basic – Routinen verzichten sollte. Eine Maschinenroutine ist nicht nur wegen der schnelleren Ausführungszeit zu empfehlen, sie bietet auch die Möglichkeit, das Scrolling im Interrupt ablaufen zu lassen. Das bringt folgende Vorteile :

1. Da das Scrolling in der Zeit geschieht, in der kein Bild dargestellt wird, können keinerlei Bildstörungen auftreten.
2. Da ein Interruptprogramm völlig unabhängig vom Hauptprogramm laufen kann, brauchen wir uns nach dem Start nicht mehr weiter darum zu kümmern.

Für diejenigen, die noch nie etwas von Interrupts o.ä. gehört haben, hier eine kleine Erklärung :

Als Interrupt bezeichnet man die Unterbrechung der Arbeit der CPU, durch ein externes Signal (in unserem Fall die Bildaustastlücke (Vertikal Blank), die alle 1/50 Sekunde ausgelöst wird), woraufhin die CPU ein ganz bestimmtes Maschinenprogramm (eben die Interruptroutine) abarbeitet.

Es gibt beim Atari zwei verschiedene Vertical – Blank Interrupts : immediate (direkt, sofort) und deferred (das heisst soviel wie aufgeschoben).

Immediate VBI’s werden immer ausgeführt, deferred werden nur ausgeführt, wenn gerade keine zeitkritische Operation durchgeführt wird (z.B. Ansprechen von Disk, Drucker usw.). Für die Interruptroutinen, die etwas mit Graphik zu tun haben, sollte man immer den immediate VBI benutzen.

Hier noch ein paar wichtige Adressen für den Gebrauch von VBI’s :

54286 - $D40E NMIEN Non Maskable Interrupt Enable.,
Hier muss Bit 6 gesetzt sein,
Damit VBI’s ausgeführt werden.

546 - $0222 VVBLKI VBLANK immediate Register

547 - $0223 Dies ist der Vektor für den immediate VBI.

548 - $0224 VVBLKD VBLANK deferred register

549 - $0225 Dies ist der Vektor für den deferred VBI.

Immediate VBI’s müssen mit dem Befehl JMP $E45F abgeschlossen werden, deferred VBI’s mit JMP $E462.

Für die Leute, die noch keine Ahnung von Maschinensprache haben, ist im Demoprogramm Nr.4 eine Interruptroutine enthalkten, die auch so verwendet werden kann; sie läuft mit allen Graphicmodi, die 40 Bytes pro Zeile verwenden, allerdings müssen Sie sich dann eine entsprechende Displaylist selbst erstellen.

Nun noch ein paar Worte zum Erstellen von Landschaften : Grundsätzlich sollte man möglichst auf den Gebrauch hochauflösender Graphik verzichten, da der Speicherbedarf einer bildschirmfüllenden Graphik immens ist (bei 192 Zeilen und einer Breite von 256 Bytes wären das rein theoretisch 48 KB).

Besser sind da die Antic – Textmodes (2,4 oder 5). Besondere Beachtung sollte man dem Übergang vom Ende zum Anfang der Landschaftsdaten widmen. Es ist nicht möglich, einfach über den ganzen Speicherbedarf zu scrollen. Was dann passiert, zeigt uns Demo #2 : Der erste Teil der Landschaft scrollt am Ende um eine Zeile versetzt ins Bild, da Antic nun die Pagegrenze überschreitet und sich Daten holt, die eigentlich nicht für diese Zeile bestimmt sind. So geht das also nicht.

Im Demoprogramm #4 wurde eine andere Technik angewandt : Es wird nicht mehr der gesamte Speicherbereich durchscrollt, sondern es wird bereits auf den Anfang des Bildes umgeschaltet, bevor das Ende

Erreicht wird. Hierzu müssen sich natürlich am Ende der Landschaft die gleichen Bilddaten befinden wie am Anfang, damit man den Sprung nicht bemerkt.

Beispiel:

Nehmen wir einmal an, die Breite des Bildschirms würde 4 Bytes entsprechen und für den Bildbereich 16 Byte reserviert sein.

Dann müsste man das Bild in einer Zeile folgendermassen aufbauen:

```
ABCDEFGHIJABCDEF          A – J : Beliebige Datenbytes

0123456789111111          Byte
          012345
```

Beim Coarse – Scroll würde der Zähler der LMS – Anweisung dann immer die Werte von 0 – 9 durchlaufen.

Es ist wichtig, dass man am Ende 6 (4+2) Bytes und nicht, wie man vielleicht vermuten könnte, 4 Bytes mit den Anfangsdaten füllt. Das liegt daran, dass beim Finescroll Bytes ja auch nur teilweise im Bildfenster erscheinen können.

Im Demoprogramm 3a wird gezeigt, wie Fine- und Course Scrolling kombiniert werden und wie unbemerkt vom Ende des Bildes auf den Anfang umgeschaltet wird. Hier sieht man schon, dass das eigentlich nur noch in Maschinensprache möglich ist. Wenn Sie den POKE 559,0 weglassen, sehen Sie, wie das Bild wackelt. (Mit POKE 559,0 wird während des Umschaltens der Bildschirm ausgeschaltet). In Maschinensprache könnte man diese Operationen während des VBI unbemerkt ausführen.

Im Demoprogramm #4 würden am Ande 42 (40+2) Bytes mit den Anfangsdaten gefüllt (Zeile 210-240).

Der Zähler in der VBI – Routine läuft dementsprechend die Werte von 0 bis 213 (Zeile 10080m Byte #3) durch.

Diesen Wert müssen Sie verändern, wenn Sie die Breite Ihrer Landschaft verändern wollen.

Auf vertikales Scrollen wurde hier nicht weiter eingegangen, da dies zum einen nicht so häufig verwendet wird, zum anderen müssten Sie sich das auch selbst beibringen können, da es prinzipiell genauso abläuft und die Speicherorganisation wesentlich einfacher ist.

```
10 REM   SCROLLING - DEMO #1
20 REM
50 GRAPHICS 0
60 POKE 752,1:REM   CURSOR AUS
97 REM
100 DIM TEXT$(255)
110 TEXT$(1,1)=" ":REM    STRING
120 TEXT$(255)=" ":REM  MIT SPACES
130 TEXT$(2)=TEXT$:REM    FUELLEN
140 REM
150 TEXT$(40,132)="Horizontales Scrollen mit Stringfunktionen in
Basic bietet eine sehr schlechte Bildqualitaet,"
160 TEXT$(133,255)=" da die Bildschirmausgabe nicht synchronisier
t ist und immer acht Pixel weit verschoben wird."
197 REM
198 REM  TEXTAUSGABE
199 REM
200 FOR  I=1 TO 215
210 POSITION 0,0
220 PRINT TEXT$(I,I+39)
230 NEXT I
240 GOTO 200
```

```
10 REM   SCROLLING - DEMO #2
20 REM
100 GRAPHICS 0:SETCOLOR 4,1,0:SETCOLOR 2,1,0:SETCOLOR 1,0,9
107 REM
108 REM  NEUE DISPLAYLIST ERSTELLEN
109 REM
110 DL=1536
120 FOR I=0 TO 2:POKE DL,112:DL=DL+1:NEXT I
130 FOR I=0 TO 23:POKE DL,66:POKE DL+1,0:POKE DL+2,120+I:DL=DL+3:
NEXT I
140 FOR I=0 TO 2:READ A:POKE DL,A:DL=DL+1:NEXT I
145 DATA 65,0,6
147 REM
148 REM  LANDSCHAFT ERSTELLEN
149 REM
150 Y=12
160 FOR X=0 TO 255
170 Y=Y+SGN(RND(0)-0.5):ST=120*256+X
180 FOR A=Y*256 TO 23*256 STEP 259
190 POKE ST+A,128:NEXT A
```

```
200 NEXT X
247 REM
248 REM  DISPLAYLIST AKTIVIEREN
249 REM
250 POKE 560,0:POKE 561,6
297 REM
298 REM  SCROLLING - ROUTINE
299 REM
300 FOR X=0 TO 215
310 DL=1540
320 FOR Y=0 TO 72 STEP 3
330 POKE DL+Y,X:NEXT Y:NEXT X
340 GOTO 300
```

```
1 REM   SCROLLING - DEMO #3
2 REM
10 GRAPHICS 0:DL=PEEK(560)+256*PEEK(561)
17 REM
18 REM  DISPLAYLIST VERAENDERN
19 REM
20 POKE DL,64+16+7:POKE DL+1,0:POKE DL+2,6
27 REM
28 REM  TEXT EINPOKEN
29 REM
30 DIM A$(48)
40 A$=" FINE-SCROLLING TEXT"
50 FOR I=1 TO LEN(A$)
60 POKE 1536+I,ASC(A$(I,I))-32:NEXT I
97 REM
98 REM  SCROLLING - ROUTINE
99 REM
100 FOR I=0 TO 15
105 FOR J=1 TO 20:NEXT J
110 POKE 54276,I:NEXT I
120 FOR I=14 TO 1 STEP -1
125 FOR J=1 TO 20:NEXT J
130 POKE 54276,I:NEXT I
140 GOTO 100
```

```
1 REM   SCROLLING - DEMO #3 A
2 REM
10 GRAPHICS 0:DL=PEEK(560)+256*PEEK(561)
11 SETCOLOR 0,13,12:SETCOLOR 2,3,4
17 REM
18 REM  DISPLAYLIST VERAENDERN
19 REM
20 POKE DL,64+16+7:POKE DL+1,0:POKE DL+2,6
27 REM
28 REM  TEXT EINPOKEN
29 REM
30 DIM A$(99)
40 A$=" FINE-SCROLLING TEXT PRAESENTIERT VOM ATARI CLUB DUESSELDO
RF   FINE-SCROLLING TEXT "
50 FOR I=1 TO LEN(A$)
60 POKE 1536+I,ASC(A$(I,I))-32:NEXT I
```

```
97 REM
98 REM  SCROLLING - ROUTINE
99 REM
100 FOR I=15 TO 0 STEP -1
105 FOR J=1 TO 10:NEXT J
110 POKE 54276,I:NEXT I:POKE 559,0
111 BI=BI+2:POKE 54276,15:POKE DL+1,BI:POKE 559,34
112 IF BI=62 THEN BI=0:POKE DL+1,0:POKE 54276,15:? “Umschaltung“:
GOTO 100:REM UMSCHALTUNG VOM ENDE ZUM ANFANG
113 POKE 559,34:GOTO 100
120 FOR I=14 TO 1 STEP –1
125 FOR J=1 TO 20:NEXT J
130 POKE 54276,I:NEXT I
140 GOTO 100
```

```
10 REM  SCROLLING – DEMO #4
20 REM
100 GRAPHICS 0:SETCOLOR 4,1,0:SETCOLOR 2,1,4:SETCOLOR 1,0,8
107 REM
108 REM  NEUE DISPLAYLIST ERSTELLEN
109 REM
110 DL=1536:RESTORE
120 FOR I=0 TO 2:POKE DL,112:DL=DL+1:NEXT I
125 CHR=15:REM 15=GRAPHICS 8
130 FOR I=0 TO 23:POKE DL,64+16+CHR:POKE DL+1,0:POKE DL+2,120+I:D
L=DL+3:NEXT I
140 FOR I=0 TO 2:READ A:POKE DL,A:DL=DL+1:NEXT I
145 DATA 65,0,6
147 REM
148 REM  LANDSCHAFT ERSTELLEN
149 REM
150 Y=12
160 FOR X=0 TO 255
170 Y=Y+SGN(RND(0)-0.5):ST=120*256+X
180 POKE ST+A,255:NEXT A
200 NEXT X
207 REM
208 REM  LANDSCHAFT KORRIGIEREN
209 REM
210 FOR I=0 TO 41
220 FOR Y=120*256 TO 143*256 STEP 256
230 POKE Y+214+I,PEEK(Y+I)
240 NEXT Y:NEXT I
247 REM
248 REM  DISPLAYLIST AKTIVIEREN
249 REM
250 POKE 560,0:POKE 561,6
297 REM
298 REM  SCROLLING - ROUTINE
299 REM
300 A=USR(256): REM  VSI AN
6970 REM
6980 REM  VBI-ROUTINE ERSTELLEN
6990 REM
7000 RESTORE 10000:SUM=0
```

```
7010 FOR I=256 TO 359
7020 READ A:POKE I,A
7030 SUM=SUM+1
7040 NEXT I
7050 READ CHECKSUM:IF CHECKSUM<>SUM THEN ? „DATA-FEHLER“:END
7060 RETURN
9970 REM
9980 REM  VBI-DATA
9990 REM
10000 DATA 104,169,12,141,34,2,169,1
10010 DATA 141,35,2,96,72,152,72,173
10020 DATA 48,2,133,203,173,49,2,133
10030 DATA 204,160,0,198,205,177,203,201
10040 DATA 65,240,61,41,240,201,80,240
10050 DATA 6,32,90,1,76,29,1,32
10060 DATA 90,1,165,205,41,3,141,4
10070 DATA 212,201,3,240,2,208,18,177
10080 DATA 203,201,213,240,8,24,105,1
10090 DATA 145,203,76,81,1,169,0,145
10100 DATA 203,32,90,1,32,90,1,76
10110 DATA 29,1,200,208,2,230,204,96
10120 DATA 104,168,104,76,95,228,0,0
10130 DATA 10697
```

Source Code der VBI Routine

```
80       .OPT NO LIST
90       .TAB 15,18,30
0100 ; ===========================
0110 ; HORIZONTAL SCROLL
0120 ; ==                       ==
0130 ; ==     IMMEDIATE VBI     ==
0140 ; ==                       ==
0150 ; ==   BY THOR - 4/11/84   ==
0160 ; ==                       ==
0170 ; ===========================
0180 ;
0190 ; === PROGRAMMSTART ===
0200 ;
0210     *=  #0100
0200 ;
0220 ;
0230 ; === ADRESSEN ===
0240 ;
0250 DLIST = $CB
0260 COUNT = $CD
0270 VVBLKI = $0222
0280 DL  =   $0230
0290 HSCROL = $D404
0300 ;
0310 ; === VBI INSTALLIEREN ===
```

```
0320 ;
0330     PLA           ;ANZAHL BYTES
0340     LDA # <VBLANK ;
0350     STA VVBLKI   ;INSTALLIERE
0360     LDA # >VBLANK ;BVI-VEKTOR
0370     STA VVBLKI+1 ;
0380     RTS          ;DAS WARS...
0390 ;
0400 ; === VBI - ROUTINE ===
0410 ;
0420 VBLANK
0430     PHA          ;AKKU RETTEN
0440     TYA          ;Y-REGISTER
0450     PHA          ;RETTEN
0460     LDA DL       ;AKTUELLE
0470     STA DLIST    ;DISPLAYLIST
0480     LDA DL+1     ;ERMITTELN
0490     STA DLIST+1 ;LOESCHE
0500     LDY #$00     ;DL-ZEIGER LOW
0510     DEC COUNT    ;DEC ZAEHLER
0520 SEARCH
0530     LDA (DLIST),Y ;HOLE DL-BYTE
0540     CMP #$41     ;IST ES JMP(65)?
0550     BEQ ENDVBI   ;JA, DAS WARS..
0560     AND ##$F0    ;IST ES LMS MIT
0570     CMP #$50     ;HSCROLL?
0580     BEQ FOUND    ;JA!
0590     JSR INCREASE ;NEIN, ERHOEHE
0600     JMP SEARCH   ;ZEIGER, TRY AGAIN
0610 FOUND
0620     JSR INCREASE ;ERHOEHE ZEIGER
0630     LDA COUNT    ;HOLE ZAEHLERWERT
0640     AND #$03     ;MASKIERE 2 BITS
0650     STA HSCROL   ;SPEICHERE BITS
0660     CMP #$03     ;COARSE SCROLL,
0670     BEQ COARSE   ;WENN 3 SONST
0680     BNE EXIT     ;WEITER
0690 COARSE
0700     LDA (DLIST),Y ;HOLE LMS LOW
0710     CMP #213     ;VON VORNE,WENN
0720     BEQ NEWSTART ;213 ERREICHT
0730     CLC
0740     ADC #$01     ;ERHOEHE
0750     STA (DLIST),Y ;LMS-LOW
0770 NEWSTART
0780     LDA #$00     ;SETZE LMS-LOW
0790     STA (DLIST),Y ;AUF 0
0800 EXIT
0810     JSR INCREASE ;ERHOEHE 2-MAL
0820     JSR INCREASE ;DL-ZEIGER
0830     JMP SEARCH   ;UND SUCHE WEITER
0840 INCREASE
0850     INY          ;ERHOEHE ZEIGER
0860     BNE NOCORR   ;PRUEFE AUF UEBERTRAG
0870     INC DLIST+1 ;INS HIGHBYTE
0880 NOCORR
0890     RTS
0900 ENDVBI
0910     PLA          ;Y-REGISTER
0920     TAY          ;UND AKKU
0930     PLA          ;WIEDERHERSTELLEN
0940     JMP $E45F    ;VBI-EXIT
0950     .END
```

Page Flipping

Wenn Sie den Artikel über die Displaylist durchgearbeitet und verstanden haben, müsste dieser Artikel eigentlich sehr einfach zu verstehen sein. Ohne Kenntnisse der Displaylist ist es schwieriger, da wir ja nicht nochmal alles wiederholen wollen.

Page Flipping ermöglicht Ihnen, mit relativ wenig Aufwand, schnell zwischen mehreren, im Arbeitsspeicher befindlichen Bildern hin- und herzuschalten.

Dies deshalb, weil im Atari kein fester Speicherbereich für den Bildspeicher existiert. Der Bildschirm ist eigentlich nur ein Fenster, das über einen bestimmten Speicherbereich gelegt wird. Somit ist es auch möglich dieses Fenster nacheinander über verschiedene Speicherbereiche zu legen. Der Speicherinhalt wird nun auf dem Bildschirm angezeigt. Genau das ist Page Flipping. Wie wir wissen, existiert für das Bild im Speicher eine Displaylist. In dieser steht drin, wo die Bilddaten stehen, wie das Bild aufgebaut werden soll usw. Diese Displaylist befindet sich meist vor dem Anfang der Bilddaten. Um nun verschiedene Bilder abspeichern zu können, müssen wir den Computer überlisten. Normalerweise wird der Atari bei Aufruf eines Graphicmodes die Bilddaten an das Ende des verfügbaren Speicherbereichs setzten. Wenn wir dem Computer nun weis machen, er verfüge über weniger Speicherplatz, wird er die Bilddaten an eine niedrigere Stelle setzen. Genau dies können wir tun. In der Speicherzelle 106 befindet sich die zur Verfügung stehende Seitenzahl. (Eine Seite entspricht 256 Byte). Angenommen wir wollen zwei GRAPHICS 8 Bilder im Speicher haben. Wir gehen dann wie folgt vor:

1. Merken der ursprünglich vorhandenen Seitenzahl.

2. Merken des Anfangs der Displaylist von Bild 1

3. Zeichnen von Bild 1

4. Von dem in 106 gespeichertem Wert wird nun 32 (32*256=8192 Byte) abgezogen, da ein GRAPHICS 8 Bild 8 K braucht. Dieser Wert wird nun wiedeer in 106 hineingepoked. Der Computer denkt nun, ihm ständen 8K weniger zur Verfügung.

5. Beim erneuten Aufruf eines GRAPHICS 8 Fensters, werden die Displaylist und die Bilddaten um 8192 Byte niedriger angesiedelt.

6. Zeichnen eines neuen Bildes.

7. Merken des jetzigen Displaylistanfangs.

8. Wenn jetzt abwechselnd auf die beiden Displaylists zugegriffen wird, ändert sich auch das jeweilige Bild. Die Mitteilung wo sich die neue Displaylist befindet, wird in den Zellen 560 und 561 abgelegt.

Probieren Sie ein bischen herum. Sie können natürlich nicht nur zwei, sondern je nach vorhandenem Speicherplatz und benötigtem Speicherbedarf viele viele Bilder auch verschiedener Graphicmodi abspeichern.

Anzuwenden ist Pageflipping bei Spielen (verschiedene Szenen), Geschäftsprogrammen (Umschaltung zwischen Balkendiagramm, Kurvendiagramm und Statistik) usw.

Schauen Sie sich jetzt das Beispielprogramm an. Es schaltet ständig zwischen zwei Graphics 7 Fenstern und einem Graphics 0 Fenster um. Als Besonderheit ist das Graphics 0 Fenster noch mit einer Displaylisténderung ausgestattet.

```
0 REM Page Flipping Demo
10 GRAPHICS 7
30 A=PEEK(106)
40 DLISTL1=PEEK(560):DLISTM1=PEEK(561)
45 FOR I=0 TO 50:COLOR RND(0)*4+1:PLOT 80,40:DRAWTO RND(0)*159,RN
D(0)*90:NEXT I
46 ? "üSeite 1"
50 POKE 106,A-32
60 GRAPHICS 7
70 DLISTL2=PEEK(560):DLISTH2=PEEK(561)
72 FOR I=0 TO 50:COLOR RND(0)*4+1:PLOT 80,40:DRAWTO RND(0)*159,RN
D(0)*90:NEXT I
73 ? "üSeite 2"
100 A=PEEK(106):POKE 106,A-32
110 GRAPHICS 0:DL=PEEK(560)+256*PEEK(561)
120 POKE DL+8,6:POKE DL+9,7:POKE 82,0:DLISTH3=PEEK(561):DLISTL3=P
EEK(560)
130 POSITION 0,3:? "         DAS            page flipping"
140 POKE DL+12,0:POKE DL+14,0:POSITION 0,6
150 ? "        Ein Superbonus beim Atari"
160 POSITION 2,18:? "Seite 3"
190 POKE 561,DLISTH1:POKE 560,DLISTL1:SOUND 0,200,10,8:FOR T=1 TO
 200:NEXT T:POKE 561,DLISTH2:POKE 560,DLISTL2
195 SOUND 0,180,10,8:FOR T=1 TO 200:NEXT T:POKE 561,DLISTH3:POKE
560,DLISTL3:SOUND 0,160,10,0
196 FOR T=1 TO 200:NEXT T
200 GOTO 190
```

Schneller Eingabemodus

Wenn beim Atari eine Taste länger als 1 Sek. Gedrückt wird, wiederholt sich diese Funktion. Manchmal ist diese Verweildauer von 1 Sek. Doch ziemlich lang. XL Besitzer können die Geschwindigkeit in der Speicherzelle 730 ändern. Besitzer von älteren Modellen müssen das nächste Programm eintippen um den gleichen Effekt zu bekommen.

```
10 REM schnellerer Eingabemodus
20 REM durch ändern der Zahl 10 in Zeile 90 geht’s noch schneller
30 FOR R=256 TO 292
40 READ B:POKE R,B:NEXT R
60 POKE 255,104
70 X=USR(255)
80 DATA 160,22,162,1,169,6,32,92,228,169,0,133,2,169,1,133,3,169
90 DATA 3,133,9,96,173,43,2,201,11,144,2,169,10,141,43,2,76,95,22
8
```

Kopieren von geschützten Diskprogrammen

Anleitung für die Diskstationen 810 und 1050

Jeder Diskettenbenutzer hat sich garantiert schon darüber geärgert, dass sich viele Diskprogramme nicht einfach so kopieren lassen. Die Disketten sind meist mit zerstörten Sektoren und/oder geänderten Formaten 'ausgerüstet'. Nur mit Software lässt sich dies nicht simulieren. Dazu ist schon eine Portion Hardware nötig. Dass wir hier nicht eine Bauanleitung für eine Happy oder einen Archiver veröffentlichen, ist w2ohl klar. Aber viele Programme lassen sich mit folgenden kleinen Tricks trotzdem kopieren und dann auch starten. Dazu werden nur zwei Teile benötigt. Ein Schalter und eine normale Diode (1N 4148 o.ä.)

Als Anzeige, in welcher Stellung sich der Schalter befindet, lässt sich ohne weiteres noch eine LED einfügen. Sie wird dann mit einem 220 Ohm Widerstand an den Ausgang des 5 Volt Spannungsreglers angeschlossen.

Da sich per Software ein defekter Sektor nicht erzeugen lässt, machen wir folgendes. Wir schalten zwischen Schreib- und Lesekopf beim Schreiben eine Diode. Beim Schreiben geht dann von der zum Schreibkopf gesandten Wechselspannung eine Halbwelle verloren. Dadurch ist dieser Sektor beim späteren Lesen nicht mehr lesbar und wird als defekt erkannt. Wie gesagt, lässt sich damit nicht jede Diskette erfolgreich kopieren, aber bei der Mehrheit funktioniert dies einwandfrei.

Dir folgende Software nimmt die meiste Arbeit ab. Sie arbeiten damit wie folgt:

Nehmen Sie sich ein Kopierprogramm, das die ganze Diskette kopieren kann. Nun kopieren Sie die Disk wie sonst auch. Eventuell

p.S. Es ist besser statt der Siliziumdiode eine Seraniumdiode zu verwenden,
Ausserdem sollte man abgeschirmtes Kabel verwenden

Auftretende Fehlermeldungen ignorieren Sie einfach. Nachdem Sie die ganze Diskette kopiert haben, laden Sie das folgende Programm ein. Nach Einlegen der Originaldisk drücken Sie L und RETURN. Das Programm liest jetzt nochmal alle Sektoren und registriert die defekten. Al nächstes wird die Destination Diskette eingelegt, der Schalter betätigt und wiederum RETURN gedrückt. Nun zerstört das Programm sämtliche registrierte Sektoren.

Damit Sie bei öfterem Kopieren nicht jedesmal die Originaldisk durchlesen müssen, besteht die Möglichkeit, die einmal registrierten Sektoren auszudrucken oder abzuschreiben. Sie geben dann beim nächsten Mal nur noch die zu zerstörenden Sektoren ein. Doch nun zur Bauanleitung :

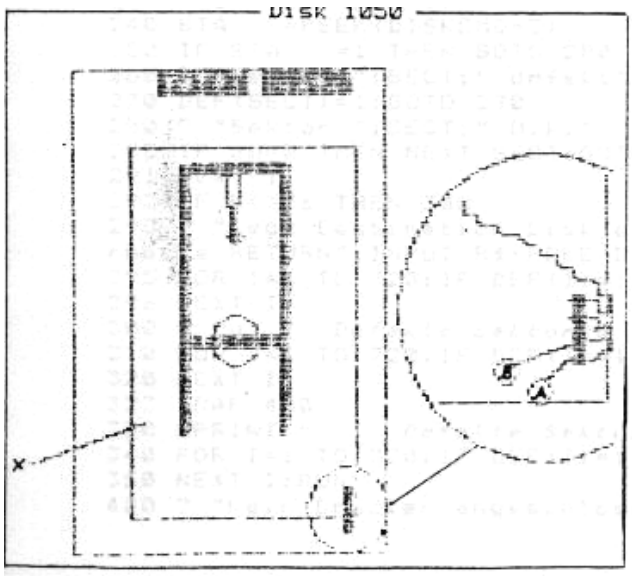
Diskstation 810

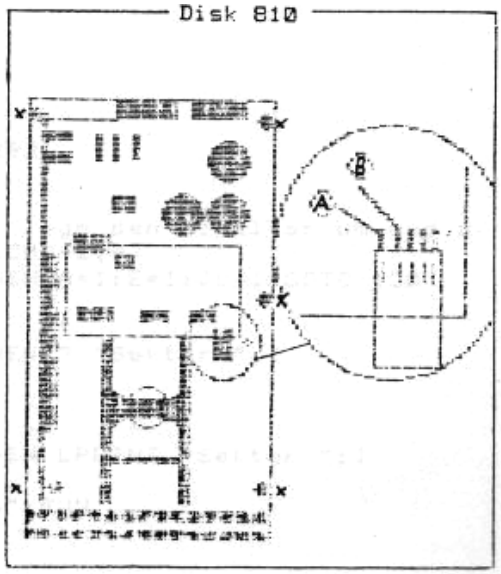
1. Lösen sämtliche Kabel von der Station.
2. Abnehmen der 4 Plastikplättchen auf der Oberseite der Station.
3. Lösen der darunterliegenden 4 Schrauben.
4. Abnehmen des Deckels.
5. Lösen der 5 gekennzeichneten Schrauben **(X)**.
6. Abnehmen der Frontblende.
7. Nach Abnahme des Atarizeichens (festgeklebt) von der Front, wird ein passendes Loch für den Schalter und eventuell für die LED gebohrt.
8. Anlöten zweier Drähte an die gekennzeichneten Stellen. (0)
9. Anlöten der Diode und der beiden Drähte an den Schalter.
10. Von Punkt 6 rückwärts ausgehend die Station wieder schliessen.
11. Fertig.

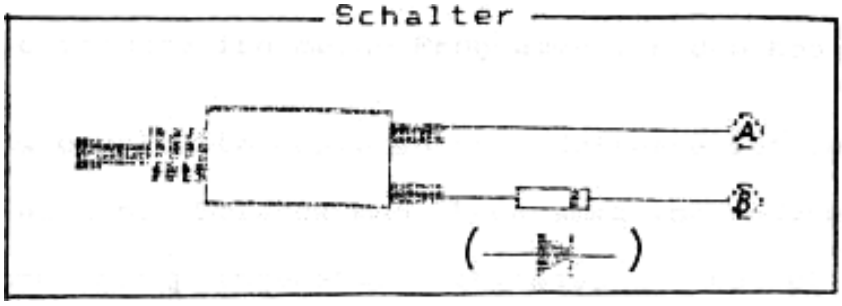
Diskstation 1010

1. Lösen sämtliche Kabel von der Station.
2. Lösen der 6 Schrauben auf der Bodenplatte.
3. Abnehmen der Frontblende.
4. Abnehmen des Deckels.
5. In die Frontblende wird links ein passendes Loch für den Schalter und eventuell für die LED gebohrt.
6. Hochklappen des Laufwerks. (X)
7. Jetzt gibt es zwei Möglichkeiten. Entweder man löst die Schrauben auf der Platine, nimmt die Platine heraus und lötet dort auf der Unterseite die beiden Drähte fest. Oder man steckt zwei Drähte in den vorhandenen Stecker mit hinein. (Man braucht die Platine nicht zu lösen)
8. Anlöten der Diode und der beiden Drähte an den Schalter.
9. Die Station wieder schliessen.
10. Fertig.

Achten Sie bei beiden Anleitungen auf die richtige Polung der Diode und das Sie keine Kurzschlüsse o.ä. veranstalten.








```
0 REM               (c) 1984 by Atari Club Düsseldorf
1 DISKCMD=768:TRAP 40000:? :? :? :POKE 710,144:POKE 709,255:
POKE 712,144:OPEN #1,4,0,"K:"
2 ? "Lesen oder Schreiben ?";:GET #1,K:? CHR$(K)
3 IF K=76 THEN POKE DISKCMD+2,82:A=1:E=720:GOTO 20
4 IF K=83 THEN 6
5 GOTO 2
6 ? "Von,Bis":INPUT A,E:IF A<1 OR A>720 OR E<1 OR E>720 THEN
 ? "FEHLER":GOTO 6
7 ? "Lege den Schalter um"
8 POKE DISKCMD+2,87
20 POKE DISKCMD+1,1
40 DIM BUFFER$(128)
50 DIM JUMP$(10),DEF(720),R$(1)
55 FOR I=0 TO 720:DEF(I)=0:NEXT I
60 BUFFER$(1,1)="X":BUFFER$(128)="X":BUFFER$(2)=BUFFER$
70 ADDR=ADR(BUFFER$)
80 AHI=INT(ADDR/256)
90 ALOW=ADDR-(AHI*256)
100 POKE DISKCMD+4,ALOW
110 POKE DISKCMD+5,AHI
111 JUMP$(1)=CHR$(104)
112 JUMP$(2)=CHR$(32)
113 JUMP$(3)=CHR$(83)
114 JUMP$(4)=CHR$(228)
115 JUMP$(5)=CHR$(96)
116 POKE 776,128:POKE 777,0
120 FOR SECT=A TO E
140 SHI=INT(SECT/256)
150 SLOW=SECT-(SHI*256)
160 POKE 778,SLOW
170 POKE 779,SHI
230 X=USR(ADR(JUMP$))
240 STA   =PEEK(DISKCMD+3)
250 IF STA   =1 THEN GOTO 280
260 ? "Sektor ";SECT;" defekt"
270 DEF(SECT)==1:GOTO 290
280 ? "Sector ";SECT;" O.k."
290 IF JU=0 THEN NEXT SECT:GOTO 292
291 NEXT I
292 IF K<>76 THEN 300
293 ? "Lege Destination Disk ein, lege den Schalter um und d
ruecke RETURN":INPUT R$:POKE DISKCMD+2,87
295 FOR I=1 TO 720:IF DEF(I)=1 THEN A=I:E=I:JU=1:GOTO 120
296 NEXT I
300 ? "ü     Defekte Sektoren :     "
310 FOR I=1 TO 720:IF DEF(I)=1 THEN ? "Sektor ";I
320 NEXT I
323 TRAP 400
330 LPRINT "     Defekte Sektoren :     "
340 FOR I=1 TO 720:IF DEF(I)=1 THEN LPRINT "Sektor ";I
350 NEXT I:RUN
400 ? "Kein Drucker angeschlossen":RUN
```

Wie schütze ich meine Programme vor dem Kopieren ?

Das unerlaubte Kopieren von Software ist zu einer Modeerscheinung geworden. Solange man Programme von anderen kopiert, ist das ja noch recht angenehm. Schliesslich kostet einem das so erworbene Programm nichts. Anders ist es allerdings, wenn man selbst der Programmierer eines Programms ist und dieses verkauft.
Wenn das Programm nicht geschützt wurde, kann man damit rechnen, dass es spätestens 1 Monat nach erscheinen, im Besitz von ca. 50 % aller Computerbesitzer ist. Dies ist natürlich äussertst ärgerlich, da so die potentiellen Käufer bereits beliefert sind. Deshalb werden Programme geschützt. Die professionellen Softwarehäuser versuchen ständig, neue Schutzroutinen zu entwickeln. Die Softwarevertreiber rechnen nun damit, dass der Schutz ca. 2 Monate hält. In dieser Zeit wollen Sie den Grossteil der kalkulierten Einnahmen bereits eingenommen haben. Deshalb die hohen Softwarepreise. Wie kann nun Otto Normalatarianer sein Programm gegen unerlaubtes Kopieren schützen?
Es gibt mehrere Möglichkeiten :

1. Listschutz

Damit nicht jeder die folgenden Massnahmen ausser Kraft setzt, benutzen wir nun einen Listschutz. Vor dem Abspeichern lassen wir folgende Programmzeile durchlaufen :

```
32000 A=PEEK(130)+256*PEEK(131):E=PEEK(132)+256*PEEK(133):
      FOR I=A TO E:POKE I,155:NEXT I
```

Das nun abgespeicherte Programm ist nicht mehr lesbar, und auch nicht mehr editierbar.
Weiterhin können wir das Programm folgendermassen schützen:

```
32010 POKE PEEK(138)+256*PEEK(139)+2,0:SAVE"D:NAME.BAS":NEW
```

bzw.

```
32010 POKE PEEK(138)+256*PEEK(139)+2,0:SAVE"C:":NEW
```

Das Programm lässt sich nicht mehr mit LOAD"D:NAME" oder CLOAD laden, sondern nur noch mit RUN"D:NAME" oder RUN"C:". Danach darf es nicht mehr durch Break oder Reset unterbrochen werden. Falls doch, hängt sich der Computer auf. Es kann also keiner mehr in das Programm eingreifen bzw. Änderungen vornehmen.

2. Ladeschutz

Damit nicht trotzdem jemand diese Schutzroutinen rückgängig macht (es gibt bereits Programme hierfür), kann man folgenden Trick anwenden:

Als erstes wir eine leere Diskette mit dem DOS 2.0 von Atari beschrieben. Dann gehen wir zurück ins Basic und laden das zu schützende Basicprogramm ein. Nun geben wir POKE 4226,X ein. X ist hierbei eine von Ihnen zu wählende Geheimzahl. Das Programm wird nun wieder mit SAVE"D:NAME" abgespeichert.

Es ist nun aus dem Directory nicht mehr zu erkennen. Wenn ein Käufer des Programm mittels einem DOS das Directory aufruft, sieht er NICHTS. Und ohne Directory kann man auch kein Programm laden. Dies geht nur, wenn man vorher die Geheimzahl in 4226 gepoked hat. Dies macht man zweckmässigerweise mit dem AUTORUN.SYS o.ä.

Der normale Wert von 4226 ist übrigens 105.

3. Kopierschutz

Als letzte Möglichkeit gibt es den Kopierschutz. Mit den oben genannten Schutzroutinen ist es zwar möglich Programmeingriffe zu verhindern (um das Copyright zu entfernen oder die folgende Routine usw.), aber das Kopieren ist immer noch möglich. Wie wir in dem Artikel über das Kopieren von Software gesehen haben, ist es möglich einen Sektor zu zerstören. Diesen Sektor kann man per Software abfragen. Falls er nicht defekt ist (nach unerlaubtem Kopieren ohne Hardwarezusatz), kann z.B. die Diskette formatiert werden. Zweckmässigerweise zerstört man auf der zu schützenden

Disk sämtliche nicht benutzten Sektoren. Bei einem normalen Basicprogramm von 150 Sektoren und dem DOS mit 80 Sektoren sind immerhin noch ca. 480 Sektoren frei. Diese werden einfach zerstört. Zweckmässigerweise lässt man mitten in den zerstörten Sektoren noch einen ganzen, den man auch abfragt. Dann ist es für jeden Schwarzkopierer eine Heidenarbeit, sämtliche defekten Sektoren zu lesen.

Achten Sie aber darauf, dass Sie das Directory nicht zerstören. Dieses steht in den Sektoren 360 bis 367 (dez.)

Die Abfrage eines defekten Sektors erfolgt so:

```
10 D=768:POKE D+2,82
20 SEKT=610: REM Wert des defekten Sektors
30 DIM BUFFER$(128),JUMP$(10)
40 ADDR=ADR(BUFFER$)
50 AMI=INT(ADDR/256)
60 ALOW=ADDR-(AHI/256)
70 POKE D+4,ALOW:POKE D+5,AHI
80 JUMP$(1)=CHR$(104)
90 JUMP$(2)=CHR$(32)
100 JUMP$(3)=CHR$(83)
110 JUMP$(4)=CHR$(228)
120 JUMP$(5)=CHR$(96)
130 POKE 776,128:POKE 777,0
140 SHI=INT(SEKT/256)
150 SLOW=SEKT-(SHI*256)
160 POKE 778,SLOW:POKE 779,SHI
170 X=USR(JUMP$)
180 ST=PEEK(D+3)
190 IF ST<>1 THEN 500
200 X=USR(58487) : REM Sektor nicht defekt – Raubkopie
500 REM Hier beginnt das eigentliche Programm
```

Es sind natürlich auch mehrere Sektoren abfragbar.

Natürlich sollte man für eigene Zwecke immer das ungeschützte Originalprogramm aufbewahren, denn es können immer mal Verbesserungen notwendig werden.

Trotz dieser Schutzmassnahmen ist es allerdings möglich Ihr Programm zu kopieren. Und zwar mit dem angegebenen Hardwarezusatz. Aber den hat ja nicht jeder.

Zu empfehlen ist ausserdem die versteckte Unterbringung einer (verschlüsselten) Seriennummer, aus der Sie dann erkennen können, welcher Kunde Ihr Programm kopiert und weitergegeben hat.

Graphics 0 Text in vier Farben

Die Graphikstufe 4 (Antic) ist eine wenig bekannte Graphikstufe des Ataris. Genau wie Stufe 0, ist sie in der Lage, 40x24 Zeichen darzustellen. Sie hat allerdings dazu noch die Möglichkeit, diese in 4 Farben darzustellen.
Dies ist dadurch möglich, weil der Anticmode 4 die Graphikmuster anders liest als im Mode 0.
Graphics Mode 0 liest für die Zeichen ein Bit auf einmal ein. Jedes Bit stellt ein Pixel auf dem Bildschirm dar. Eine 1 wird mit einer anderen Helligkeit dargestellt als eine 0. Antic Mode 4 dagegen liest zwei Bit auf einmal ein. Jedes Bitpaar stellt hier ein Pixel dar. Ein Pixel im Mode 4 ist doppelt so breit wie im Mode 0. Da der Atari aber im Mode 0 bei fast allen Zeichen zwei Pixel breite Punkte verwendet (ein Punkt (.) ist z.B. zwei Pixel breit), fällt der Unterschied fast nicht auf. Es gibt vier verschiedene Kombinationen der zwei Bits : 00, 01, 10, 11. Jede Kombination stellt eine andere Farbe dar. Die Farbe des Bitpaares 00 ist in der Stelle 712 gespeichert; die des Bitpaares 01 in 708; 10 in 709 und die Farbe des Paares 11 steht in 710.
Programm 1 ändert die Displaylist in den Anticmode 4 um. Wenn Sie dieses Programm starten, werden Sie feststellen, dass dieser Graphicsmode nicht zu benutzen ist. Warum ? Da die Zeichen nicht dafür gedacht sind um als Bitpaar eingelesen zu werden, sehen Sie nur ein Durcheinander auf dem Screen. Um eine praktische Nutzung aus dem Mode 4 zu ziehen, muss der Zeichensatz geändert werden. Dies geschieht mit dem Programm 2. Die vier Farben werden dann durch das Eingeben von verschiedenen Typen erreicht (Grosschrift, Kleinschrift, CTRL Zeichen und Invers CTRL Zeichen) . Programm 3 demonstriert diese Möglichkeiten.

Achten Sie darauf, dass Sie erst Programm 2 laufen lassen, bevor Sie Programm 3 einladen. Um nun nicht immer die ganzen DATA's im Programm benutzen zu müssen, können Sie den Zeichensatz auch abspeichern. Programm 4 saved den Zeichensatz auf die Diskette (erst Progr.2 laufen lassen) . Programm 5 holt ihn wieder zurück. Sie können also Ihr Programm mit vierfarbigem Text ausstatten, indem Sie mit Teil 5 den Zeichensatz einladen und dann mit Teil 1 nach Antic Mode 4 umschalten. Mehrfarbige Schrift sieht allemal besser aus als Weiss auf Blau.

```
0 REM PROGRAMM 1
10 DL=PEEK(560)+256*PEEK(561)
20 POKE DL+3,PEEK(DL+3)+2
30 FOR I=DL+6 TO DL+28
40 POKE I,4
50 NEXT I
```

```
30015 REM  PROGRAMM 2
30110 TOP=PEEK(106)
30120 LOWT=TOP-5:POKE 106,LOWT
30130 GRAPHICS 0:? "BiTtE wArTeN":SETCOLOR 4,0,0:SETC
OLOR 1,13,6:SETCOLOR 0,8,8
30140 Z=30800:CS=256*(TOP-4)
30150 H=CS+264:J=H+207:L=30300:K=1:GOSUB Z
30160 H=CS+520:J=H+207:K=1.5:GOSUB Z
30170 H=CS+776:J=H+207:K=0.51:L=30560:GOSUB Z
30180 H=CS+128:J=H+79:K=1:L=30570:GOSUB Z
30190 H=CS+8:J=H+55:K=1.5:L=30650:GOSUB Z:H=CS+64:J=H
+7:L=30560:GOSUB Z:H=CS+72:J=H+7:L=30640:GOSUB Z
30200 H=CS+256:J=H+7:L=30560:GOSUB Z
30210 H=CS+80:J=H+15:K=0.5:L=0:GOSUB Z:H=CS+104:J=H+7
:GOSUB Z:H=CS+120:J=H+7:GOSUB Z:H=CS+224:J=H+23:GOSUB
 Z
30220 H=CS+472:J=H+23:GOSUB Z
30240 H=CS:J=H+7:GOSUB Z:H=CS+496:J=H+23:GOSUB Z
30300 DATA 4,1,2,3,3,2,3,4
30310 DATA 4,2,3,2,3,3,2,4
30320 DATA 4,2,3,5,5,3,2,4
30330 DATA 4,6,3,3,3,3,6,4
30340 DATA 4,2,5,2,5,5,2,4
30350 DATA 4,2,5,2,5,5,5,4
30360 DATA 4,2,5,5,3,3,2,4
30370 DATA 4,3,3,2,3,3,3,4
30380 DATA 4,2,1,1,1,1,2,4
30390 DATA 4,7,7,7,7,3,2,4
30400 DATA 4,3,3,2,6,2,3,4
```

```
30410 DATA 4,5,5,5,5,5,2,4
30420 DATA 4,3,2,2,2,3,3,4
30430 DATA 4,3,2,2,3,3,3,4
30440 DATA 4,2,3,3,3,3,2,4
30450 DATA 4,2,3,3,2,5,5,4
30460 DATA 4,2,3,3,3,3,2,7
30470 DATA 4,2,3,2,2,3,3,4
30480 DATA 4,2,5,2,7,7,2,4
30490 DATA 4,2,1,1,1,1,1,4
30500 DATA 4,3,3,3,3,3,2,4
30510 DATA 4,3,3,3,3,2,1,4
30520 DATA 4,3,3,3,2,2,3,4
30530 DATA 4,3,3,1,1,3,3,4
30540 DATA 4,3,3,2,1,1,1,4
30550 DATA 4,2,7,1,1,5,2,4
30560 DATA 4,2,3,3,3,3,2,4
30570 DATA 4,6,1,1,1,1,2,4
30580 DATA 4,2,7,8,6,5,2,4
30590 DATA 4,2,7,2,7,7,2,4
30600 DATA 4,7,8,3,2,7,7,4
30610 DATA 4,2,5,2,7,3,2,4
30620 DATA 4,2,5,2,3,3,2,4
30630 DATA 4,2,7,1,1,5,5,4
30640 DATA 4,2,3,2,3,3,2,4
30650 DATA 4,2,3,2,7,7,7,4
30660 DATA 4,4,4,4,4,4,4,4
30670 DATA 9,9,9,9,9,9,9,9
30680 DATA 10,10,10,10,10,10,10,10
30690 DATA 11,11,11,11,11,11,11,11
30700 POKE 756,TOP-4
30710 DL=PEEK(560)+256*PEEK(561):POKE DL+3,PEEK(DL+3)
+2:FOR I=DL+6 TO DL+28:POKE I,4:NEXT I
30720 END
30800 FOR I=H TO J:READ G:ON G GOSUB Z+20,Z+30,Z+40,Z
+50,Z+60,Z+70,Z+80,Z+90,Z+100,Z+110,Z+120:NEXT I
30805 IF L=0 THEN RETURN
30810 RESTORE L:RETURN
30820 POKE I,32*K:RETURN
30830 POKE I,168*K:RETURN
30840 POKE I,136*K:RETURN
30850 POKE I,0:RETURN
30860 POKE I,128*K:RETURN
30870 POKE I,160*K:RETURN
30880 POKE I,8*K:RETURN
30890 POKE I,40*K:RETURN
30900 POKE I,85*K:RETURN
30910 POKE I,170*K:RETURN
30920 POKE I,255*K:RETURN
```

```
4 REM PROGRAMM 3
5 REM Demonstration des Antic Modes 4
10 DL=PEEK(560)+256*PEEK(561)
20 POKE DL+3,PEEK(DL+3)+2
30 FOR I=DL+6 TO DL+28
40 POKE I,4
50 NEXT I
60 POKE 756,PEEK(106)+1
65 SETCOLOR 1,13,4:SETCOLOR 2,1,8:SETCOLOR 0,7,2:SETC
OLOR 3,3,6
```

```
70 FOR I=0 TO 20:? "ATARI club [illegible]
club DUESSELDO RF             ";:NEXT I
75 FOR J=0 TO 30
80 FOR I=14 TO 0 STEP -1:SETCOLOR 1,8,I:NEXT I
85 FOR I=0 TO 14:SETCOLOR 1,8,I:NEXT I
90 NEXT J
91 SETCOLOR 1,13,4
92 FOR J=0 TO 30
93 FOR I=14 TO 0 STEP -1:SETCOLOR 0,8,I:NEXT I
94 FOR I=0 TO 14:SETCOLOR 0,8,I:NEXT I
95 NEXT J
96 SETCOLOR 0,7,2
97 FOR J=0 TO 30
98 FOR I=14 TO 0 STEP -1:SETCOLOR 2,8,I:NEXT I
99 FOR I=0 TO 14:SETCOLOR 2,8,I:NEXT I
100 NEXT J
101 SETCOLOR 2,1,8
110 SETCOLOR 1,2,6:SETCOLOR 2,1,8:SETCOLOR 1,13,4:SET
COLOR 2,2,6:GOTO 110
```

```
99 REM PROGRAMM 4
100 REM CHARAKTER ABSAVEN
110 C=PEEK(756):C=C*256:TRAP 160
120 OPEN #1,8,0,"D:CHSET.001"
130 DOR I=0 TO 1023
140 A=PEEK(C+I):PUT #1,A
150 NEXT I
160 CLOSE #1
```

```
100 REM PROGRAMM 5
105 POKE 106,PEEK(106)-4:GRAPHICS 0
110 POKE 756,PEEK(106)
130 TRAP 180
140 OPEN #1,4,0,"D:CHSET.001"
150 FOR I=0 TO 1023
160 GET #1,A:POKE C+I,A
170 NEXT I
180 CLOSE #1
```

Kassettenbenutzer müssen bei den Programmen 4 und 5
folgende Zeilen ändern :

```
119 REM Bei Programm 4 :
120 OPEN #1,8,0,"C:"
```

```
139 REM Bei Programm 5 :
120 OPEN #1,4,0,"C:"
```

Atari Funktionstasten

Hier ist noch ein äusserst nützliches Programm für Sie.

Wie Sie bestimmt wissen, gibt es einige Computer, die Funktionstasten haben. Man braucht nur eine Taste drücken und schon steht ein ganzer Befehl auf dem Bildschirm. Jetzt gibt es ein Programm dass ermöglicht, etwas ähnliches zu haben. Wenn Sie das folgende Programm eintippen, und laufen lassen, schreibt es auf eine Diskette einen AUTORUN.SYS. Wenn sich auf dieser Diskette ausserdem noch DOS 2 von Atari befindet, lädt der Computer sich automatisch beim Einschalten das Funktionstastenpro- gram ein. Zum Glück braucht dieses Programm kaum Speicherplatz.

In den Zeilen 20100 bis 20170 befinden sich die Commandos, die nach dem Drücken der jeweiligen Taste mit CTRL erscheinen. Also in Zeile 20100 befinden sich folgende Befehle : Mit Drücken von CTRL A erscheint auf dem Bildschirm COLOR; mit CTRL C erscheint CHR$(, mit CTRL D – DATA usw. Wie Ihr seht sind in diesen Zeilen schon Befehle vorgegeben. Ihr könnt sie natürlich weglassen und eigene Befehle dafür einsetzen. Wie Ihr an Zeile 20160 sehen könnt, muss es nicht immer ein Befehl sein. Es kann genauso eine Reihe Befehle sein. Wer z.B. ein Programm schreibt in dem zig' mal vorkommt 'X=USR(1536):ON X GOTO ...' der sollte diese Befehlssequenz einge- ben. Er erspart sich eine Menge Schreibarbeit.

Die DATA Befehle müssen mit dem Buchstaben beginnen, bei dem der folgende Befehl ausgedruckt werden soll. Danach muss der Affenschwanz (Shift+8) kommen. Achten Sie unbedingt darauf, dass anstatt einem Komma ein '/' eingetippt werden muss (siehe 20160) und dass als letzter Befehl 5 mal der Affenschwanz erscheint (siehe 20170).

```
80 GRAPHICS 0:SETCOLOR 2,0,2:SETCOLOR 4,0,2:SETCOLOR 1,13,12:SETC
OLOR 0,1,6
90 DL=PEEK(560)+256*PEEK(561):POKE DL+6,7:POKE DL+7,6:DIMS$(128)
,A$(1):POKE 82,0
106 ? "}Der Atari Club Duesseldorf praesentiert:        atari
      FUNKTIONSTASTEN":REM Unterstrichen = revers
110 ? :? :? :? :? "Sollen die Kommandos ausgedruckt werden";
120 INPUT A$:IF A$="J" THEN FLAG=1
190 READ D:IF D<0 THEN 215
210 CT=CT+1:GOTO 190
215 GRAPHICS 0:SETCOLOR 2,0,2:SETCOLOR 4,0,2
220 READ S$:CT=CT+LEN(S$)
240 IF S$(1,1)="@" THEN 310
280 IF FLAG <>1 THEN LPRINT " CTRL ";S$(1,1);" = ";S$(2)
300 GOTO 220
310 CT=CT+7424-6
330 CTH=INT(CT/256):CTL=CT-CTH*256:POKE 764,255
350 ? "Druecke RETURN wenn die Diskette, auf   die der AUTORUN.SY
S geschrieben werden soll,in Drive #1 ist.";:INPUT A$
370 OPEN #3,8,0,"D:AUTORUN.SYS":RESTORE
390 READ D
400 IF D<0 THEN 470
420 IF D=500 THEN D=CTL
430 IF D=501 THEN D=CTH
440 PUT #3,D
450 GOTO 390
470 READ S$:FOR I=1 TO LEN(S$)
490 D=ASC(S$(I,I))
500 IF D=ASC("\") THEN D=ASC(",")
510 PUT #3,D:NEXT I
530 IF S$(1,2)="@@" THEN 560
540 GOTO 470
560 FOR I=1 TO 7:READ D:PUT #3,D:NEXT I
570 CLOSE #3:? CHR$(125):? "     F E R T I G":END
10040 DATA 255,255,0,29,500,501,234,160,1,173,8,2,153,182,29,200,
173,9,2,153,182,29,169,44,141,8,2,169,29
10050 DATA 141,9,2,169,500,141,231,2,24,105,112,169,501,141,232,2
,105,1,133,15,96,88,142,192,29,140,193
10060 DATA 29,173,9,210,72,41,128,208,4,104,24,144,113,169,3,133,
245,169,30,133,246,104,41,63,170,189
10070 DATA 195,29,201,255,240,94,133,247,160,0,177,245,197,247,24
0,24,201,64,240,80,200,177,245,201,64
10080 DATA 208,249,200,24,152,101,245,133,245,144,228,230,246,208
,224,230,245,208,2,230,246,174,22,228
10090 DATA 172,23,228,232,208,1,200,142,187,29,140,188,29,160,0,1
77,245,201,64,240,13,140,194,29,32,186
10100 DATA 29,172,194,29,200,24,144,237,169,0,133,77,169,48,141,4
3,2,174,192,29,172,193,29,104,64,174
10110 DATA 192,29,172,193,29,76,0,80,0,76,0,80,69,58,155,0,0,0,76
,74,59,255,255,75,43,42,79,255,80,85
10120 DATA 255,73,45,61,86,255,67,255,255,66,88,90,52,255,51,54,2
7,53,50,49,44,32,46,78,255,77,47,255
10130 DATA 82,255,69,89,255,84,87,81,57,255,48,55,255,56,60,62,72
,72,68,255,255,71,83,65,-1
20100 DATA ACOLOR @,CCHR$(@,DDATE @,FFOR @,GGOTO @
20110 DATA HGOSUB @,IINPUT @,JDRAWTO @,KGRAPHICS @,LLIST @
20120 DATA MLPRINT @,NNEXT @,OPADDLE(@,PPLOT @,QPOSITION @
20130 DATA RRETURN@,SSOUND @,TTHEN @,UREAD @,VSETCOLOR @,WSTICK(@
20140 DATA XSTRIG(@,YSTR$(@,2LOAD "D:@,3SAVE "D:@
20150 DATA 4LIST "P:"@,5LIST "D:@
20160 DATA 0FOR I=0 TO 3:SOUND I\0\0\0:NEXT I:SETCOLOR 2\11\0:SET
COLOR 4\11\0:SETCOLOR 1\1\15@
20170 DATA 6ENTER "D:@,@@@@@
30040 DATA 0,224,2,225,2,0,29
```

Externe Tastatur für den Atari 400

Die Folientastatur des Atari 400 ist zwar nach einer kurzen Eingewöhnungszeit genauso schnell zu bedienen wie eine richtige; wer aber einmal für längere Zeit an einer Schreibmaschinentastatur gesessen hat, möchte diese nicht mehr missen. Es gibt im Moment zwar schon Tastaturen für den nachträglichen Einbau, sie kosten aber ca. 140.- DM. Hier nun eine Anleitung zum Selbstbau einer solchen Tastatur.

Notwendige Bauteile:

1 Tastatur

2 24-polige Ic Fassungen

min. 2 Meter 22 poliges Flachbandkabel

Notwendiges Werkzeug:

1 Lötkolben max. 30 Watt

1 Abisolierzange

1 Seitenschneider

und Elektronik Lötzinn

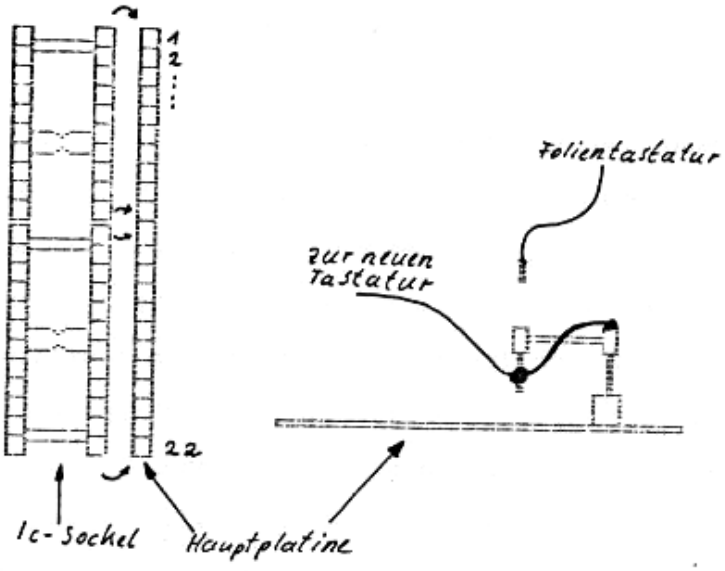
Die Tastaturen kann man oft fertig kaufen. Wenn möglich, sollte man eine wählen, die noch keinerlei Verdrahtung besitzt. Meistens sind diese Tastaturen aber bereits verdrahtet. In diesem Fall muss man sämtliche Leiterbahnen durchtrennen.

Als erstes löst man sämtliche Kabel am Atari und entfernt das Modul. Nun werden die 4 Schrauben auf der Unterseite herausgedreht und der der Atari wieder umgedreht. Man öffnet die Modulklappe und hebt das Oberteil soweit ab, bis man in der Lage ist, die Tastatur vom oberen Gehäuseteil zu lösen. Wie man nun sieht, ist die Tastatur über ein 22 poliges Flachbandkabel über einen Stecker mit der Hauptplatine verbunden.

Nun nehmen wir uns die Ic-Fassungen vor. Da wir nur 22 Pins benötigen, sägen (oder schneiden) wir an einer Fassung 2 Pins ab.

Die nun folgende Verdrahtung ist aus der folgenden Zeichnung ersichtlich : Man lötet ein Kabelstück oben an der Fassung an; das andere Ende wird mit dem entsprechenden Pin der gegenüberliegenden Reihe verbunden und geht dort zur externen Tastatur. Zu empfehlen ist hier der Einbau einer 25 poligen RS 232 Buchse im Atari. Man kann dann seine externe Tastatur jederzeit vom Computer trennen. Nun wird die Originaltastatur mit dem Ic- Sockel verbunden, an dem Gehäuse befestigt und der Atari wieder zugeschraubt. Natürlich sollte man sich vorher die Bedeutung der herausgelegten Kabel (bzw. der Pins am RS 232 Stecker) notieren.
Die nun folgende Verdrahtung ist das schwierigste. Am besten geht man schrittweise vor; d.h. Sie verdrahten immer nur jeweils 4 oder 5 Tasten und überprüfen diese dann bei eingeschaltetem Computer. Die Tasten müssen alle funktionieren, es darf auch kein dauerndes Repeat auftreten. Nach der erfolgreichen Verdrahtung, wird die Tastatur in ein geeignete Gehäuse eingebaut. Diese werden mittlerweile vom Elektronikfachhandel angeboten. Man kann jetzt seine tastatur jederzeit erweitern (z.B. durch Bau einer Zehnertastatur usw.)
Es folgt nun der Schaltplan und die Belegung in Form einer Tabelle.

Taste	Anschlusspins
System Reset	18 - 22
Option	19 - 22
Select	20 - 22
Start	21 - 22
Break	1 - 9
Delete B/B	1 - 17
Invert	1 - 16
Clear	1 - 15
0	1 - 14
9	1 - 13
8	1 - 12
7	10 - 1
6	10 - 2
5	12 - 2
4	13 - 2
3	14 - 2
2	15 - 2
1	16 - 2
Escape	17 - 2
Tab	17 - 4
Q	10 - 4
W	15 - 4
E	14 - 4
R	13 - 4
T	12 - 4
Y	10 - 4
U	10 - 3
I	12 - 3
O	13 - 3
P	14 - 3
-	15 - 3
^	16 - 3
Return	17 - 3
Caps/Lowr	17 - 6
*	16 - 5
+	15 - 5
;	14 - 6
L	13 - 5
K	12 - 5
J	11 - 5
H	11 - 6
G	12 - 6
F	13 - 6
D	14 - 6
S	15 - 6
A	16 - 6
CTRL	9 - 5
Shift	8 - 9
Z	16 - 8
X	15 - 8
C	14 - 8
V	13 - 8
B	12 - 8
N	7 - 10
M	12 - 7
, (Komma)	13 - 7
. (Punkt)	14 - 7
/(Querstr.)	15 - 7
Space	7 - 11
Atari Symbol	16 - 7


1
2
22
Folientastatur
zur neuen Tastatur
Ic-Sockel
Hauptplatine

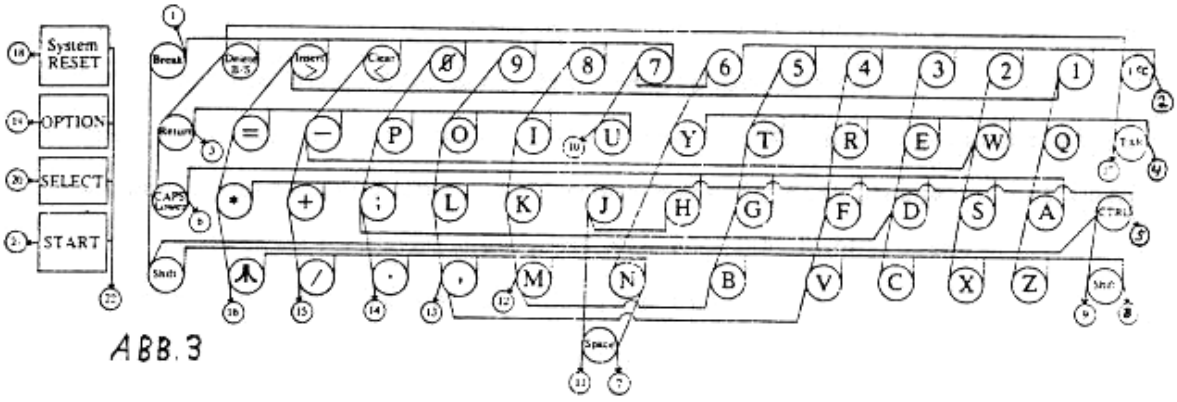

System RESET
OPTION
SELECT
START
Break
Return
Shift
Space
Insert
Clear
Ctrl
TAB

ABB. 3

Micro DOS

Micro DOS ist eine Diskutility für eine oder zwei Diskettenstationen. Micro DOS benötigt nur 3 Pages im RAM und ist somit permanent verfügbar. Und es verträgt sich mit dem im RAM befindlichen Programm. MEM.SAV kann somit vergessen werden. Wenn man DOS eintippt braucht man keine Angst mehr zu haben, dass das Basicprogramm gelöscht wird. Dies Utility kann auch ohne Cartridge betrieben werden. Das Micro DOS beinhaltet sämtliche DUP Funktionen ausser File- und Diskduplikationen. Aber Micro DOS hat dafür noch einige andere Kommandos mehr : Hex oder Dez. RUN und einen ständig verfügbaren Hex.-Dez. und Dez.-Hex. Konverter. Damit kann endlich die lästige rumrechnerei entfallen. Natürlich kann man vom Micro DOS auch das normale DUP einladen.

Die Funktionen des Micro DOS sind am Anfang vielleicht etwas schwer zu behalten, aber Sie werden merken, dass diese Funktionen bald einfacher zu Handhaben sind als die DUP Funktionen.

Folgendes müssen Sie bei der Eingabe beachten :

Das Kommando für's Directory, Formatieren und die Schreibfunktionen benutzen D# und erwarten die Eingabe der Stationsnummer. Ein RETURN bedeutet Station Nr.1. Zum Formatieren erscheint ein '?' als Kontrolle. Hier ist ein 'Y' notwendig wenn man formatieren möchte. Die Funktionen die einen Filename benötigen zeigen ein 'FN?'. Der Device Name muss mit angegeben werden (D:; D1: oder D2:). Die Renamefunktion benötigt nur einmal den Devicenamen (D:TEST.COM,TEST.BAS).

Die Kommandos :

DIR – Disketten Directory. Benötigt Eingabe der Stationsnr.

1-4 oder RETURN für 1

REN – Rename.

* - Lock. File sperren

Un* - Unlock. File entsperren

FMT – Formatieren. Fragezeichen mit 'Y' beantworten.

ERS – Erase. File löschen.

CRT – Cartridge. Zurück zum Basic.

LOD – binary load

§RN – Run at adress. Hex. Oder Dezimal

WDS – Schreibe DOS.SYS (kein DUP)

$>. – von Hex. Nach Dezimal. Gib $ und # zusammen ein

.>$ - von Dezimal nach Hex. Gib '.' und # zusammen ein

!DS – Gehe ins Atari DOS (DUP.SYS)

Das Micro DOS befindet sich in dem Autorun.Sys, den das folgende Programm auf die Diskette schreibt. Sie müssen dann natürlich das DOS 2.0S auf die Diskette schreiben. Denn sonst kann das Micro DOS nicht geladen werden.

```
20 DIM H$(104):TRAP 95:OPEN #1,8,9,"D:AUTORUN.SYS":FO
R N=1 TO 16:READ H$:B=0
30 FOR C=1 TO 99 STEP 2
40 IF H$(C,C+2)="END" THEN GOSUB 80:END
50 A=16*(ASC(H$(C))-48-7*(ASC(H$(C))>57))
60 A=A+ASC(H$(C+1))-48-7*(ASC(H$(C+1))>57):B=B+A
70 PUT #1,A:NEXT C:GOSUB 80:? "Zeile ";N*100;" ist in
 Ordnung.":NEXT N
80 IF B<>VAL(H$(LEN(H$)-3)) THEN ? "ERROR IN ZEILE ";
N*100:STOP
95 RETURN
96 END
100 DATA FFFFFC1C082020AE1E84F2888CE3028CE10284FFA280
A00586F384F4A03A9848B9A71F20B21E68A88810F3A9088D5A032
0766200
200 DATA 1EA004AD8005D9E21FF01C8810F8C944D06DA9068D5A
0320741D206A1EA21020761E20701E10F6B9E71F48C9FEF009209
51D5209
300 DATA 68206C1E109CA93F20B21E20761EAD8005C959D08D20
741D30E6A94420B21EA92320B21E20761EAD80058DED1F8DF41FA
0064896
400 DATA B9EC1F91F38810F860A94620B21EA94E20B21EA93F20
B21E4C761EC957D01120741DC8B9F31F91F388D0F8206A1E10B1C
94C5965
500 DATA D03485FF20951D4E5A03206A1E20991EA9FFC5D4D004
C5D5F0F320B6DD20991E20701F20E61DF0E5ADE302F008200517A
9005814
600 DATA BDE30260C953D03720951DA0FFC8B1F3C92FD007EE5A
03A99B91F3C99BD0EE206A1EA9FF85D485D520951E20ED1E20951
E206602
700 DATAB6DD20ED1E20951E20701F1039C940D00620EED1E6CD4
00C943D0034C74E4C921D00EA940850CA915850D2040154C9F17C
9244940
800 DATA D00820081F20541F100AC92ED00620021F202D1F106F
A903A210D008A200A909D002A9059D4203A91E9D4803A9059D450
3A93740
900 DATA 809D4403A9009D49032056E4302B60A90BD002A907A2
109D4203A9029D4803A9009D4503A9D4D0D9A90C10BA85D4A200A
90B4938
1000 DATA 9D4203A90110E3C003F0D1C088F0159848A9C520B21
E2044DA6885D420541F68684CFC1CA5FFF0F720E61DADE102F0EF
20086297
1100 DATA 17D0EAA94020B21E20761EAC8005C02EF006C024F00
8D0BD2000D84CD2D92044DAA001B1F3C99BF01938E930C90A3002
E9075889
1200 DATA A20406D426D5CAD0F905D485D4C810E160A000A201A
9F035D44A4A4A4A20481FA90F35D420481FCA10EB3021C90A3002
69065311
1300 DATA 693091F3C86020AAD920E6D8A000B1F3C8C98030F92
97F8891F3C8A99B91F74C701EE6D4D002E6D5A210A5E09D4403A5
E19D7399
1400 DATA 450338A5D4E5E09D4803A5D5E5E14C8C1E204015A9F
EBDE702A91F8DE802A9FC850AA91C850B603E9B9B5344A120243E
AE205778
1500 DATA 2E3EA4205344D7204E52C020444FCC205641D39B545
2C320544DC6205352C5204D4ED2202A4ED5204B4CAA205249C49B
52454936
1600 DATA 2A554620212324FE44313A2A2E2A9B44313A444F532
E5359539BA991850CA91F850D4C941FE202E302FE1FEND0000000
00003710
```

Wie der Atari sich selbst programmieren kann

Wir werden jetzt etwas lernen, dass Sie später nicht mehr vermissen möchten. Es gibt eine Möglichkeit, wie der Atari sich selbst programmieren kann. Man kann mit der gleichen erklärten Möglichkeit z.B. Zeilen löschen, neue Programmzeilen schreiben und ähnliches. Dies ist die Technik des Dynamik Keyboard. Es wird mit POKE 842,13 eingeschaltet.

```
10 FOR I=1000 TO 1500 STEP 10:?CHR$(125):POSITION 2,5:? I;" REM Zeile ";I:? :? "CONT"
20 POSITION 2,0:POKE 842,13:STOP
30 POKE 842,12:NEXT I
```

Geben Sie jetzt mal das nächste Programm ein. Wie Sie sehen, hat das Programm sich selbst 50 neue Programmzeilen geschrieben. Wenn nun die folgenden Zeilen gelöscht werden, werden diese 50 Zeilen wieder gelöscht.

10 FOR I=1000 TO 1500 STEP 10:?CHR$(125):POSITION 2,5:? I:? "CONT"

Jetzt sehen Sie ungefähr, wie das Ganze funktioniert. Hier aber noch eine genaue Erklärung. Mit POKE 842,13 wird der automatische Cursor eingeschaltet. Mit POKE 842,12 wird der Cursor wieder ausgeschaltet. Wie Sie sehen, wird erst die einzulesende Nachricht auf den Bildschirm geschrieben. Dann wird die Nachricht CONT geschrieben. Jetzt wird der Cursor mit POSITION 2,0 wieder oben auf den Screen gesetzt. Nun POKE 812,13 und STOP. Der Cursor läuft nun den Screen antlang, bis er CONT erreicht hat. Die Anweisungen, die er zwischendurch durchlaufen hat, sind jetzt eingelesen. Nachdem er CONT eingelesen hat, wird der Programmablauf fortgesetzt. Geben Sie nun noch das folgende Programm ein. Es ist nur ein Beispiel, wofür das Dynamik Keyboard hervorragend eingesetzt werden kann.

```
10 DIM FORMEL$(30):? "Gib die zu zeichnende Formel ein.     Z.B. Y=SIN(X)*40+50 ";:INPUT FORMEL$
20 ? CHR$(125):POSITION 2,5:? "120 ";FORMEL$:? :? "CONT":POSITION 2,0:POKE 842,13:STOP
30 POKE 842,12
100 GRAPHICS 8:SETCOLOR 2,7,0:SETCOLOR 1,1,15:COLOR 1:LIST 120
110 DEG :FOR X=0 TO 1000 STEP 10
120 REM Hier steht dann die eingegebene Formel
130 PLOT X/3.15,Y:NEXT X
```

Rettung von defekten Kassettenprogrammen

Wer keine Diskettenstation besitzt, muss mit einigen gravierenden Nachteilen leben. Bei einer Diskettenstation dauert das Laden eines 16K Programms noch ca. 30 Sekunden. Ein Kassettenrecorder benötigt dafür immerhin noch stattliche 7 Minuten. Ausser diesem und noch ein paar anderen Nachteilen, ist ein defektes Programm auf einer Kassette so gut wie nicht mehr zu retten. Wir wollen hier einige Möglichkeiten aufzählen, welche Möglichleiten ein verzweifelter Computerbesitzer zur Rettung hat.

*** Wenn man ein Programm von Kassette einlädt, stoppt der Recorder automatisch nach dem Einladen. Meist wird jetzt angefangen zu spielen oder zu programmieren. Indes drückt die Gummirolle des Recorders munter gegen den Transportstift. Dazwischen befindet sich das Band. Jeder kann jetzt ausprobieren, wie das Band nach 5 Min. aussieht. Man sieht einen Knick. Dieser Knick hat im Moment auf dieser Kassettenseite keine weiteren Folgen. Sobald aber das Programm gelöscht und ein neues Programm auf diese Seite geschrieben wird, passiert es: das Programm wird über den Knick im Band hinaus geschrieben. Dieser Knick kann, wenn er stark genug ist, dass erfolgreiche Laden des Programms verhindern. Falls sich dazu noch auf der Rückseite der Kassette Programme befinden, sind diese natürlich auch von diesem Bruch betroffen.

Tip Nr. 1:

Sofort nach dem Laden die STOP-taste drücken.

*** Kassetten haben die unangenehme Eigenschaft, durch Reibung am Tonkopf, kleine Teilchen dort zurückzulassen. Verwendet man allzu billige Kassetten, kann es schon nach 5 Min. dazu führen, dass der Tonkopf völlig zugekleistert ist. Dann lässt sich kaum ein Programm noch laden oder abspeichern.

Tip Nr. 2:

Regelmässig (alle 2 Betriebsstunden und vor wichtigen Aufnahmen) den Tonkopf mit Spiritus oder Alkohol reinigen. Reinigungskassetten sind rausgeschmissenes Geld. Achtung ! Kein Terpentin oder Verdünnung zum Reinigen nehmen. Am besten und billigsten ist immer noch Spiritus. Ausserdem wird am falschen Ende gespart, wenn man Billigkassetten kauft. 3,- DM sollte man schon hinlegen. Aufgrund der Sicherheit sollte man nur C 60 verwenden. Die sogenannten Computerkassetten (C 10 o.ä) sollte man nicht kaufen. Diese enthalten in der Regel das schlechteste Bandmaterial.

*** Dem besten Rekorder kann es schon mal passieren, dass er Bandsalat produziert. Das Band ist dann so gut wie verloren. Wenn es gerissen ist, gibt es für das in diesem Teil befindlichen Programm keine Rettung mehr.

Tip Nr. 3:

Ist das Band nur zerknittert, kann man versuchen es zu bügeln. Dies ist oft erfolgreicher als es auf den ersten Blick aussieht. Stellen sie das Bügeleisen auf die kleinste Stufe und lassenSie es nicht zu lange auf einer Stelle. Dies kann auch angewendet werden, wenn sich ein Knick von der Andruckrolle auf dem Band befindet.

*** Oft gibt es Lesefehler von Fremdkassetten. Man bekommt eine Kassette zugeschickt und wenig später macht sich Verzweiflung breit : Das Programm lässt sich nicht laden. Es gibt zwei Möglichkeiten zur Rettung.

Tip Nr. 4:

Als erstes versucht man den Tonkopf vom Recorder zu verstellen. Dazu befindet sich hinter dem Tonkopf eine kleine Schraube. Oft reicht nur eine halbe Umdrehung um das Programm laden zu können.

Die beste Stellung kann folgendermassen herausgefunden werden:
Als erstes wird die Kassette eingelegt und PLAY gedrückt. Wenn die Piepstöne, die jetzt ganz leise aus dem Fernsehlautsprecher kommen, am lautesten sind, ist der Tonkopf richtig eingestellt. Versuchen Sie nun das betreffende Programm wieder zu wieder zu laden. Merken Sie sich bei dieser Prozedur aber unbedingt die Ausgangsstellung der Tonkopfschraube, damit sie später Ihre eigenen Programme wieder laden können.
Wenn das alles nichts hilft, gibt es noch die Möglichkeit, das betreffende Programm mit zwei Recordern zu kopieren. Man benötigt dazu zwei HiFi Kassettenrekorder. Man überspielt nun das Programm mit sehr hoher Aussteuerung auf eine andere Kassette. Auch hier muss beim abspielenden Recorder der Tonkopf eingestellt werden. Falls das immer noch nicht hilft, können Sie das Programm abschreiben und anfangen die Unordnung, die Sie mittlerweile verursacht haben, aufzuräumen.
*** Wem ist es noch nicht passiert. Man hat seit 4 Stunden ein Programm eingetippt oder entwickelt und es abgespeichert. Spätere Ladeversuche werden mit einem hämischen ERROR- 148 oder ERROR-133 quittiert. Da steht man nun und kann sein eigenes Programm nicht mehr laden. Die Ladefehler steigen übrigens exponential mit der Wichtigkeit und Länge des Programms an. Meist ist es auch noch so, dass der Computer wohl noch einen Teil einlädt, dann aber (vorzugsweise kurz vor Ende des Programms) einen Ladefehler erkennt. Das ärgerliche ist nun, dass das gesamte Programm verloren ist, obwohl ja immerhin ein Teil eingeladen wurde. Abhilfe schafft nun das folgende Programm. Nach dem Eingeben speichern Sie es erstmal ab. Nachdem Sie es haben laufen lassen, müssen Sie ? USR(1536) eingeben. Nun wird das defekte Programm

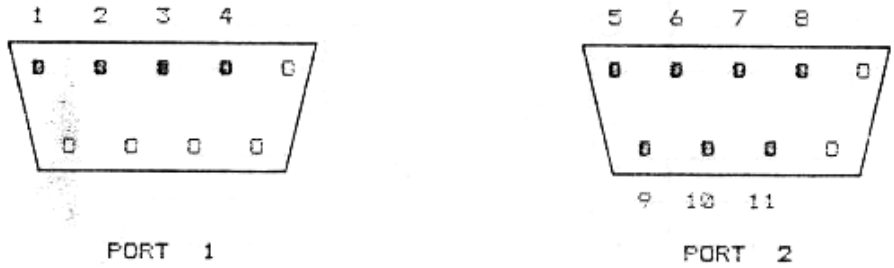
eingelegt, PLAY gedrückt und dann zum Laden noch RETURN gedrückt. Das Programm lädt jetzt das defekte Basicprogramm bis zum Fehler ein. Jetzt können Sie das Programm wieder vervollständigen. Eventuell bringt die letzte Zeile den Computer durcheinander. Falls das der Fall ist, müssen Sie das Programm jetzt erstmal mit LIST"C:",0,E abspeichern. E ist hierbei die letzte vollständige Zeile im Programm. Nun wird abgespeicherte Programm mit ENTER"C:" wieder eingeladen. (Vorher Atari aus-einschalten).

```
10 REM Basic Rettungsprogramm
20 REM    von Bob Fine
30 REM aufrufen mit ?USR(1536)
40 TRAP 60: RESTORE :I=0
50 READ A:POKE 1536+I,A:I=I+1:X=X+A:GOTO 50
60 IF X<>29124 THEN ? "DATA Fehler":END
70 TRAP 40000:NEW
1000 DATA 104,165,128,133,0,165,129,133,1,230,1,162,16,169,3,157
1010 DATA 66,3,169,251,157,68,3,169,6,157,69,3,169,4,157,74
1020 DATA 3,169,128,157,75,3,32,86,228,48,75,160,0,140,255,6
1030 DATA 32,207,6,48,65,72,32,207,6,48,59,170,104,172,255,6
1040 DATA 24,101,128,153,128,0,200,138,101,129,153,128,0,200,192,
14
1050 DATA 208,219,165,140,133,142,133,144,133,14,165,141,133,143,
133,145
1060 DATA 133,15,32,207,6,48,15,160,0,145,0,230,0,166,0,208
1070 DATA 2,230,1,76,98,6,192,136,240,22,162,0,142,255,6,189
1080 DATA 218,6,32,164,246,174,255,6,232,224,11,208,239,76,163,6
1090 DATA 162,0,142,255,6,189,229,6,32,164,246,174,255,6,232,224
1100 DATA 10,208,239,162,16,169,12,157,66,3,32,86,228,162,0,142
1110 DATA 255,6,189,238,6,32,164,246,174,255,6,232,224,14,208,239
1120 DATA 174,252,2,224,255,240,249,169,255,141,252,2,76,116,228,
162
1130 DATA 16,169,7,157,66,3,32,86,228,96,70,101,104,108,101,114
1140 DATA 32,32,32,32,155,75,111,109,112,108,101,116,116,155,80,8
2
1150 DATA 69,83,83,32,82,69,84,85,82,78,155,67,58,155,0,0
```

Druckerinterface für Centronics Drucker

So ziemlich jeder möchte wohl einen Drucker haben. Da die Ataridrucker nun nicht gerade das gelbe vom Ei sind und der Atari keine Centronics Schnittstelle besitzt, a steht dieser Anschaffung noch einiges im Wege. Nun gibt es mittlerweile zwar schon Centronicsinterface's zum Nachrüsten, aber diese kosten immerhin ca. 300,- DM. Ebenfalls werden auch schon Druckerinterface's für Joystickports angeboten. Aber auch die kosten alle etwa 100,- DM. Deshalb wird hier ein Bauplan veröffentlich, dessen Verwirklichung Sie nur den Preis für die notwendigen Stecker kosten wird. Eventuell kommt da noch ein Ic zu, der aber nur 2.50 DM kostet. Dieser wird nötig, wenn das Signal vom Computer zu schwach ist, um den Drucker anzusteuern.
Um das Interface anzusteuern, ist eine spezielle Software nötig. Diese muss einmal eingeladen werden. Dann kann der Drucker mit den normalen Basic Kommandos angesteuert werden. Das Interface wird an die Joystickports 1 und 2 angeschlossen. Mit dem Interface können die ersten 128 ASCII Zeichen übertragen werden. Graphik auszudrucken ist nicht möglich. Aber dies ist mit den meisten Interface's, sie an die Ports angeschlossen werden, nicht möglich.

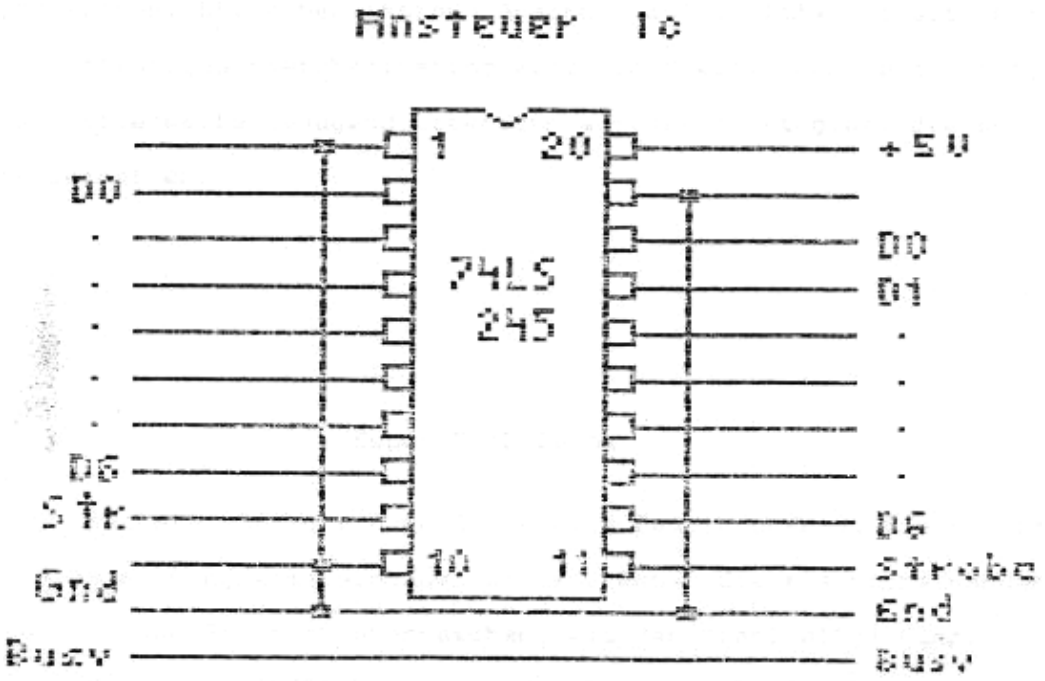
Anschlussbelegung der Joystickports



D0	= 1	D1	= 2
D2	= 3	D3	= 4
D4	= 5	D5	= 6
D6	= 7	Strobe	= 8
Busy	= 9	+5 Volt	= 10
GND	= 11		

```
10 REM Funktioniert nur mit DOS 2.0 von Atari
100 FOR I=1664 TO 1776:READ X:POKE I,X:P=P+X:NEXT I
105 IF P<>13142 THEN ? "DATA Fehler":STOP
110 POKE 1777,PEEK(12)
120 POKE 1778,PEEK(13)
130 POKE 12,216:POKE 13,6
140 ? "Druecke RESET um das Programm einzu-  schalten"
150 NEW
160 REM Daten fuer das Druckerprogramm
170 DATA 142,6,146,6,212,6,169,6,212,6,212,6,76,215,6,169,255,208
,2,169,0,162,56,142,2
180 DATA 211,141,0,211,169,60,141,2,211,169,255,141,0,211,160,1,9
6,172,17,208,208,251,120,201,155
190 DATA 208,2,169,13,9,128,141,0,211,32,140,6,41,127,141,0,211,1
60,0,173,17,208,208,7,136
191 REM ODER ANSTATT 41,127 -> 9,128
200 DATA 208,248,160,138,88,96,160,1,88,96,160,146,96,160,0,185,2
6,3,201,80,240,5,200,200,200
210 DATA 208,244,169,128,153,27,3,169,6,153,28,3,76
```

Falls Ihr Drucker einen Low Strobe benötigt müssen Sie
In Zeile 190 die Zahlen 41,127 in 9,128 ändern
Falls das Signal des Computers nicht ausreicht um den
Drucker zu steuern, wird der folgende IC eingesetzt.
Dies ist auch nötig, wenn das Kabel zum Drucker länger
Als 1.50 Meter ist.



Hier noch ein paar interessante POKE's

Mit POKE 173,1 können XL Besitzer den Cursorton ausschalten

Mit POKE 16,64 und POKE 53774,64 wird die BREAK Taste ausgeschaltet

Mit ?PEEK(20)/50+PEEK(19)*5.1+PEEK(18)*1300.5 hat man eine genaugehende Uhr.

Mit POKE 65,0 werden Disketten und Cassettengeräusche ausgeschaltet

POKE 77,0 sollte in einem Spiel ohne Tastatureingabe gelegentlich Ausgeführt werden. (Damit die Farben sich nicht ändern)

Mit POKE 752,0 wird der Cursor ausgeschaltet

Mit POKE 755,4 hat man Spiegelschrift.

Mit POKE 838,166 und POKE 839,238 geht alles was sonst auf dem Bildschirm erscheint jetzt auf den Drucker.

Mit PEEK(53279) werden die START usw. Tasten abgefragt.

Mit POKE 54018,52 wird der Motor vom Kassettenrecorder eingeschaltet. Man kann dann Musik über den Fernseher hören.

Dies ist selbstverständlich kein vollständiges Speicherlisting. Die obenstehenden Zellen sind meist nur als kleine Aufbesserungen zu gebrauchen. Sie erhebt keinen Anspruch auf Vollständigkeit. Auf ein vollständiges Speicherlisting wurde in diesem Buch verzichtet, da es mittlerweile genügend Literatur auf dem Markt gibt, die so etwas enthalten.

Super Farb Demos

Auf der nächsten Seite finden Sie einige Demoprogramme, die Sie in Ihre eigenen Programme einbauen können. Geben Sie sie nacheinander ein und lassen Sie sich überraschen, was der Atari alles kann. Drücken Sie immer RESET bevor sie das Programm laufen lassen.

```
0 FOR I=0 TO 57:READ X:POKE 1536+I,X:NEXT I
1 DATA 104,104,104,170,104,104,133,204,169,0,133,203,133,19,133,2
0,169,3,133,205,230
2 DATA 203,230,203,198,205,240,244,173,11,212
3 DATA 10,24,101,203,141,10,212,141,10,212
4 DATA 157,22,208,173,11,212,201,130,240,229,165,19,197,204,208,2
27,96
5 GRAPHICS 7:COLOR 1:PLOT 0,0:DRAWTO 20,20:COLOR 2:DRAWTO 40,40:C
OLOR 3:DRAWTO 50,50
6 FOR Y=1 TO 4:A=USR(1536,Y-Y*(Y=3),0.5):NEXT Y


10 D=PEEK(560)+256*PEEK(561):POKE D+2,PEEK(D+2)+128
20 FOR F=1536 TO 1576:READ A:POKE F,A:NEXT F
30 DATA 72,169,0,141,255,5,238,255,5,173,255,5,141,10,212,201,189
,208,243
32 DATA 173,20,0,201,200,208,5,169,0,141,20,0,141,16,6,169,0,141,
23,208,104,64
40 POKE 512,0:POKE 513,6:POKE 54286,192


10 D=PEEK(560)+256*PEEK(561):POKE D+2,PEEK(D+2)+128
20 FOR F=1536 TO 1584:READ A:POKE F,A:NEXT F
30 DATA 72,169,0,141,255,5,238,255,5,173,155,5,141,10,212,201,189
,208,243
32 DATA 173,20,0,201,200,208,5,169,0,141,20,0,141,16,6,169,10,141
,10,212,141,24,208,169,0,141,23,208,104,64
40 POKE 512,0:POKE 513,6:POKE 54286,192


10 D=PEEK(560)+256*PEEK(561):POKE D+2,PEEK(D+2)+128
15 FOR F=1536 TO 1570:READ A:POKE F,A:NEXT F
20 DATA 72,138,72,174,255,5,138,141,10,212,141,24,208,238,255,5,1
73,255,5,201,0,240
30 DATA 3,76,3,6,169,50,141,255,5,104,170,104,64
40 POKE 512,0:POKE 513,6:POKE 54286,192


10 D=PEEK(560)+256*PEEK(561):POKE D+2,PEEK(D+2)+128
20 FOR F=1536 TO 1570:READ A:POKE F,A:NEXT F
30 DATA 72,169,0,141,255,5,238,255,5,173,255,5,141,10,212,141,24,
208,201,189,208,240
32 DATA 173,20,0,141,2,6,105,189,141,19,6,104,64
40 POKE 512,0:POKE 513,6:POKE 54286,192


10 D=PEEK(560)+256*PEEK(561):POKE D+2,PEEK(D+2)+128
20 FOR F=1536 TO 1573:READ A:POKE F,A:NEXT F
30 DATA 72,169,0,141,255,5,238,255,5,173,255,5,141,10,212,141,24,
208,201,189,208,240
32 DATA 173,20,0,141,19,6,169,0,141,10,212,141,24,208,104,64
40 POKE 512,0:POKE 513,6:POKE 54286,192


0 GRAPHICS 2:POKE 710,0:FOR F=0 TO 11:? #6;"DER ATARI COMPUTER":N
EXT F
10 D=PEEK(560)+256*PEEK(561):POKE D+2,PEEK(D+2)+128
20 FOR F=1536 TO 1573:READ A:POKE F,A:NEXT F
30 DATA 72,169,0,141,255,5,238,255,5,173,255,5,141,10,212,141,22,
32 DATA 173,20,0,141,19,6,169,0,141,10,212,141,22,208,104,64
40 OKE 512,0:POKE 513,6:POKE 54286,192
50 END
```

Riesen Tastaturbuffer

Der Ataricomputer hat leider ein kleines Manko. Es wäre praktisch, wenn er einen grösseren Tastaturbuffer hätte. Wie sie sicher schon gemerkt haben, speichert der Computer immer die zuletzt gedrückte Taste. Manchmal wäre es aber recht praktisch, wenn er mehr als nur ein Zeichen speichern könnte. Mit dem folgenden Programm ist es möglich.

```
30000 REM Riesen Buffer
30010 DATA 104,173,8,2,141,96,6,173,9,2,141,97,6,169,0,141,14,212
,120,169,52,141,8,2
30020 DATA 169,6,141,9,2,169,98,141,36,2,169,6,141,37,2,169,192,1
41,14,212,169,0,133,204
30030 DATA 133,205,88,96,173,9,201,159,240,36,152,72,173,252,
2,201,255,240,19,164,204,192,100
30040 DATA 240,9,230,204,200,173,9,210,153,143,6,104,168,104,64,1
65,204,197,205,208,231,104,168,76
30050 DATA 95,6,173,252,2,201,255,208,35,165,204,197,205,240,23,2
30,205,164,204,192,120,176,15,164
30060 DATA 205,192,120,176,9,185,143,6,141,252,2,76,98,228,169,0,
133,204,133,205,76,98,228
30070 IF PEEK(521)=6 THEN STOP
30080 FOR I=0 TO 142:READ N:X=X+N:POKE 1536+I,N:NEXT I
30090 ? USR(1536)
```

Farbspeicherung

Fast jeder ändert die Bildschirmfarbe bevor er anfängt zu programmieren. Kaum einer setzt sich dem Hellblau, das der Bildschirm beim einschalten des Computers zeigt, für längere Zeit aus. Jeder hat seine Lieblingsfarbe (z.B.SETCOLOR 2,8,0 usw.). Leider muss man es aber jedesmal nach Drücken von RESET neu eintippen. Deshalb hier die Abhilfe. Das Programm funktioniert nicht mit OSS A+ DOS.

```
10 FOR F=1536 TO 1548:READ A:POKE F,A:NEXT F
11 ? "Hintergrundfarbe :";:INPUT A:? "Helligkeit :";:INPUT B:? "Z
eichenhelligkeit :";:INPUT C:D=A*16+B
12 POKE 1537,D:POKE 1542,C
20 DATA 169,0,141,198,2,169,0,141,197,2,76,64,21
30 POKE 12,0:POKE 13,6
```

Wer wollte nicht schon immer einen Lightpen haben. Wir veröffentlichen hier für Sie einen solchen. Er ist nicht allzu nachzubauen. Wer allerdings keinerlei Erfahrung mit den Umgang mit elektronischen Bauelementen hat, sollte lieber die Finger davon lassen. Aber man hat ja immer einen Bekannten, der so etwas kann.

Nun zur Funktion des Lightpens :
Ein Fernsehbild wird zeilenweise von einem Elektronenstrahl geschrieben. Dieser Elektronenstrahl erzeugt beim Auftreffen auf die Leuchtschicht des Bildschirms einen leuchtenden Punkt. Aufgrund der Trägheit unserer Augen und der Nachleuchtdauer der Leuchtschicht, sehen wir keinen huschenden Punkt, sondern ein ganzes Bild. Der Lightpen dagegen ist schnell genug, diesen Punkt zu erkennen. Wenn der Leuchtpunkt auf den Phototransitor trifft, gibt der Ligthpen ein Signal an den Computer ab. In unserem Atari wertet nun ein Grafikbaustein diese Meldung aus. Er schreibt in die Speicherzellen 564 und 565 (dez.) die horizontale und vertikale Position des Lightpens. Diese kann man nun ohne weiteres abfragen.

Es ist darauf zu achten, dass der Bildschirm eine bestimmte Helligkeit hat. Die Empfindlichkeit des Pens kann man an dem Poti einregeln. Das untenstehende Programm dient nur zum ausprobieren. Es dürfte wohl keine Schwierigkeiten bereiten, ein komfortableres Lightpenprogramm zu schreiben. Da die horizontalen Werte naturgemäss etwas schwanken, sollte man hier aus einigen Werten den Mittelwert errechnen (evtl. den höchsten und den niedrigsten Wert weglassen).

Lightpenprogramm :

```
0 SETCOLOR 2,0,12: SETCOLOR 1,0,0
10 POKE 53277,0
20 POKE 53277,1 : REM Joystickport Nr. 1
30 H=PEEK(564) : REM Horizontaler Wert
40 V=PEEK(565) : REM Vertikaler Wert
50 PRINT H,V : GOTO 30
```