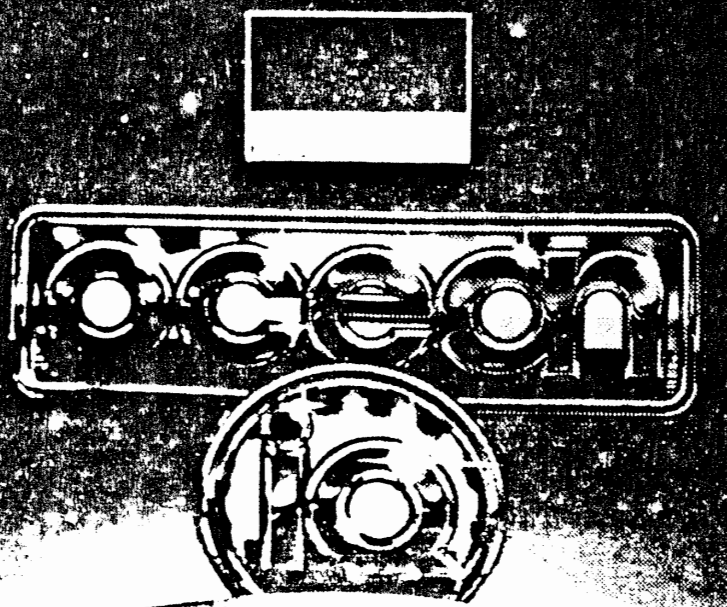
LASER BASIC



INHALT

Von John Gross

EINFÜHRUNG

Die erweiterte Laser BASIC ist eine Erweiterung des bestehenden BASIC-Befehlsatzes im BASIC ROM von Schneider. Obgleich Locomotive BASIC eine der elegantesten Implementierungen einer alten Sprache ist, waren ihre Merkmale so zu konzipieren, daß sie so flexibel wie möglich wurde. BASIC hat zahlreiche Anwendungen, jedoch ist unser spezifisches Interessengebiet Grafik und Bewegung. Laser BASIC wurde so konzipiert, daß ihre leichte Anwendung und insbesondere die Geschwindigkeit verbessert wurde, mit der sich komplizierte bewegte grafische Darstellung herstellen ließen, und hierzu wurden etwa 200 Befehle und Funktionen eingeschlossen. Die zusätzlichen Befehle werden alle als "BAR"-Befehle (RSX) implementiert, so daß das erste Symbol eines jeden Befehls ein senkrechter Strich ist (umgeschaltetes @). Wo im Text auf Befehle Bezug genommen wird, wurde der "BAR" ausgelassen.

Die Benutzer, die bereits die Lightning-Serie kennen, erkennen die meisten Befehlssätze wieder, obgleich wir der Klarheit halber Befehlszeichnungen geändert haben. Durch Laser BASIC entstehen keine selbständigen Programme (das erweiterte Interpreter Programm muß geladen sein), jedoch wurde auch ein Compiler entwickelt, durch die Ihre LASER-BASIC-Programme schneller laufen und keinen residenten erweiterten Interpreter benötigen dürfte. Das bedeutet, daß Sie Ihre Programme kommerziell vertreiben können.

Das Hauptziel von Laser BASIC ist die Verwendung von Grafik, Bewegung und Ton und so einfach und "stillschweigen" wie möglich. Der Befehlssatz wurde so konzipiert, daß jemand selbst mit bescheidenen Kenntnissen von BASIC und geringster Geduld so schnell wie möglich Ergebnisse sieht (daher so viele Beispiele). Es ist jedoch zu bemerken, daß wenn LASER BASIC lediglich zwei Dutzend sehr einfache Bewegungsbefehle hatte, der unternehmungslustigere Benutzer gut und gerne in Frage stellen könnte, ob er genug für sein Geld erhält. Das System enthält übrigens über 200 Befehle mit gewissen ziemlich fortgeschrittenen Merkmalen (zum Beispiel Verfolgungssprites und Tonschaltung). Wir hoffen daher, daß Sie das Paket auch nach monatelangem Gebrauch noch interessant finden und nicht enttäuschend

ANDERE SCHNEIDER-LASER-PRODUKTE

Obgleich Laser BASIC ein völlig in sich abgeschlossenes Paket ist, wird sein Einsatz von mehreren Begleitprodukten ergänzt. Informationen über Verfügbarkeit und Preise erhalten Sie von Ihrem Fachhändler oder direkt von uns.

Der Laser-BASIC-Compiler

Laser-BASIC-Programme benötigen einen residenten erweiterten Interpreter, und das beschränkt die mögliche Benutzerbasis für Ihre ergänzten Programme. Der Laser-BASIC-Compiler kompiliert jedoch Ihre BASIC-Programme zu schnellem Maschinencode Z80, der auf jedem Schneider Mikrocomputer Z80 ohne residente Laser BASIC betrieben werden kann. Der größte Teil von Schneider BASIC wird berücksichtigt, jedoch wird integre Arithmetik verwendet, so daß keine der Gleit-Kommandfunktionen unterstützt wird. Es ist eine gute Idee, alle Ihre Programme normal mit DEFINT A-Z zu beginnen. Das stellt nicht nur sicher, daß Sie integre Arithmetik verwendet haben, sondern Ihr Programm läuft auch schneller, und Sie benötigen weniger Speicherplatz.

Der Laser-Grafikzeichner

Obgleich Laser BASIC mit dem Sprite-Generatorprogramm geliefert wird, steht ein viel leistungsfähigeres Bildschirm-/Spritezeichenprogramm mit ICON-Laufwerk zur Verfügung, das die Grafik in einem Format erstellt, das mit allen Produkten im Laserprogramm kompatibel ist. Vergessen Sie nicht, daß kein anderes Zeichenprogramm eine kompatible Grafik erstellt.

Der Laser-Musik-/Tonkomponist

Laser BASIC wird auch hier wiederum mit einem Tongeneratorprogramm geliefert, mit dem Musik- und Toneffekte in Sprites assembliert werden können. Ein viel umfassenderes Paket, das die Entwicklung von Musik und Ton erleichtert und beschleunigt, ist ebenfalls lieferbar. Es ist das einzige Paket, das kompatible Tonsprites erzeugt.

Laser-Assembler/-Monitor

Wir haben eigentlich nicht genug Platz, um diesem Produkt gerecht zu werden. Wir glauben daß es sich um den leistungsfähigsten Assembler/Monitor handelt, der je für ein Gerät Z80 geschaffen wurde. Der Assembler besteht aus mehreren Compilers gleicher Qualität und generiert einen Code für strukturierte Programmschleifen sowie arithmetische und logische Ausdrücke. Der Monitor hat gewisse Merkmale, die bisher nur in teuren Hardware-Analysegeräten und Emulatoren zur Verfügung stand.

Obgleich der Assembler seinen Text speziell prägt, um ihn schnell und leistungsfähig zu machen, können Ursprungsdateien für zahlreiche Alternativassembler geladen und in speziell geprägtes Format für Laser-Assembler verwandelt werden.

Der Graphik-/Tonquellcode

Zur Entwicklung und Austestung der Grafik- und Tonroutinen, die Laser BASIC und andere LASER-Pakete benutzen, wurde viel Zeit und Mühe verwendet. Der fortgeschrittene Benutzer, der in Assemblersprache arbeitet, kann durch Verwendung dieser Routinen beim Schreiben von Spielen viel Arbeit sparen. Die Routinen werden als Quellcode geliefert, in einem Format, das mit dem Laser-Assembler kompatibel ist, und jeder Eingabepunkt wird sorgfältig dokumentiert (die Funktionen des Code selbst werden nicht dokumentiert). Jede Quellcode, der nicht benutzt wird, kann natürlich vor Assemblerzeit gelöscht werden, so daß der endgültige Betrieb-Programmcode wahrscheinlich viel kleiner ist als der vom Kompilierer verwendete.

Mini Laser BASIC

Eine verkleinerte Version von Laser BASIC, die nur die Hauptmerkmale von Laser BASIC unterstützt und viel mehr Speicherplatz für Sprites läßt, und BASIC-Quellcode steht ebenfalls zur Verfügung.

GLOSSAR DER IM VORLIEGENDEN HANDBUCH BENUTZTEN AUSDRÜCKE:

SPRITES

Ein Sprite ist ein durch Software steuerbares Grafiksymbol, das im Speicher in mehreren Datenbytes gespeichert wird. Der Speicherblock, der ein Sprite darstellt, kann dargestellt werden, indem es durch PUT in den Videospeicher eingegeben wird, wo es als sichtbares Bild auf dem Bildschirm erscheint. Bezüge auf "Sprites" bedeuten allgemein Datenbytes im Speicher, die das Grafiksymbol bilden, gelten jedoch gelegentlich für das sichtbare Bild auf dem Bildschirm. Mit Laser BASIC können 255 Sprites definiert werden, jedes mit seinen eigenen, vom Benutzer wählbaren Dimensionen (bis 255 x 255). Die Grenze von Spritegröße und -anzahl, die dem Benutzer zur Verfügung steht, ist bedingt durch den verfügbaren Speicherplatz.

BILDSCHIRMFENSTER

Ein Bildschirmfenster ist ein Teil des Bildschirms, der von vier Variablen bestimmt wird (siehe Laser-BASIC-Variablen): COL, ROW, HGT und LEN. COL hat einen Bereich von 0 bis 79, ROW von 0 bis 199, HGT von 1 bis 200 und LEN von 1 bis 80. Die Einheiten von COL und LEN sind Bytes – 1/2 Symbole bei Betrieb von 4 Farben und 1/4 Symbole bei Betrieb von 16 Farben. Die Einheiten für ROW und HGT sind Bildpunkte in beiden Betriebsarten. COL und ROW geben die Spalten- und Zeilenposition auf dem Bildschirm von der linken Fensterecke ausgehend an, wobei sich ROW 0 oben auf dem Bildschirm befindet und COL 0 auf der äußersten linken Seite. HGT und LEN geben die Maße des Fensters an.

Um das Beispiel eines Fensters auf dem Bildschirm zu sehen, folgende Zeile eingeben und ENTER drücken.

```
|COL,8:|ROW,8:|LEN,32:|HGT,64:|INVV
```

SPRITEFENSTER

Ein Spritefenster ist der Teil eines Sprite, der durch die Variablen SPN, COL, ROW HGT und LEN bestimmt wird. SPN gibt Sprite an, COL und ROW die Spalte und Reihe, in der sich das Sprite befindet und HGT und LEN die Größe des Fensters.

Wenn das mit diesen Variablen bestimmte Fenster außerhalb des Sprite liegt oder seine Grenze überlappt, wird der Befehl nicht ausgeführt, sondern es erscheint keine Fehlermeldung.

SPRITEPLATZ

Der Spriteplatz befindet sich in dem Speicherbereich, der alle bereits definierten Sprites enthält. Die Oberkante des Spriteplatzes ist 28671 dezimal (6FFF HEX), die Unterkante erstreckt sich von diesem Punkt aus nach unten.

BILDSCHIRMOPERATIONEN

Hierbei handelt es sich um Vorgänge, die in einem bestimmten Bereich des Bildschirm ausgeführt werden. Der benutzte Bildschirmbereich ist das sogenannte Bildschirmfenster und wurde in einem früheren Absatz definiert. Die Operationen selbst sind Durchlauf, Inversion, Reflexion, Vergrößerung usw., und die meisten Befehle in dieser Klasse haben ein nachgestelltes 'V' für "Video", also MIRV, MGXV usw. Wenn das Fenster den Bildschirmrand überlappt, wird es "beschnitten", so daß es "auf dem Bildschirm" liegt.

SPRITEOPERATIONEN

Diese erstrecken sich mehr oder weniger auf die gleichen Operationen wie bei den Bildschirmfenster-Befehlen, jedoch diesmal wird ein Sprite im Speicher behandelt anstatt auf einen Teil des Bildschirms. Die einzige verwendete Variable ist SPN, und die Syntax für diese Befehle ist die gleiche wie bei denen für Bildschirmoperationen, wobei ein 'S' das 'V' ersetzt. Das Ergebnis dieser Operationen ist nur sichtbar, wenn das Sprite erneut auf dem Bildschirm mit PUT oder MOVE dargestellt wird.

SPRITENFENSTEROPERATIONEN

Diese Operationen werden an einem Teil eines Sprite im Speicher ausgeführt, und hier stehen mehr oder weniger die gleichen Möglichkeiten zur Verfügung wie für die letzten beiden Operationssätze. Diesmal tragen die Befehle ein nachgestelltes 'P' und der Spriteteil, für den die Operation gilt, wird von SPN, COL, ROW, HGT und LEN definiert. Das Spritefenster wird "beschnitten", so daß es sich "auf dem Sprite" befindet.

SPRITE-/BILDSCHIRMOPERATIONEN

Hierbei handelt es sich um Operationen zwischen Bildschirm und Sprite. Die Maße des Sprites werden als die Maße des Bildschirmfensters benutzt, und COL und ROW werden eingesetzt, um die Koordinaten der oberen linken Fensterecke anzugeben, so daß die Operationen mit dem Variablen SPN, COL und ROW definiert werden. Wenn das Fenster teilweise "außerhalb" des Bildschirms liegt, wird es "beschnitten", so daß für einen Teil des Sprite die Befehle "PUT" oder "GOT" gelten. Befehlen in dieser Klasse wird ein 'PT' oder 'GT' vorgestellt, also GTBL, PTXR, PTND usw.

SPRITE-/SPRITEFENSTEROPERATIONEN

Hierbei handelt es sich um Operationen zwischen einem ganzen Sprite und einem Fenster in einem zweiten Sprite. Die zweiten Spritenummern werden in SP1 gehalten (das Sprite, das nicht das Fenster enthält) und in SP2 (das Sprite, das das Fenster enthält). Die Maße des Fensters sind die Maße des Sprite, das nicht das Fenster enthält und die Position des Fensters im Sprite, dessen Nummer in SP2 gehalten wird, die von SCL und SRW angegeben werden. Das Fenster wird "beschnitten", wenn es Sprite SP2 überlappt. Befehlen in dieser Gruppe wird 'PM' oder 'GM' vorgestellt

SPRITEFENSTER-/BILDSCHIRMOPERATIONEN

Hierbei handelt es sich um Operationen zwischen einem Bildschirmfenster und einem Spritefenster. Wie beschrieben, definieren ROW, COL, HGT und LEN das Bildschirmfenster, diesmal werden jedoch SCL und SRW zur Definition der Fensterposition im Sprite benutzt. Auch hier wird das Fenster

Ein Pseudosprite ist ein Sprite, das keine Daten für die [illegible] um Speichern einer Maschinencode-Subroutine, eines Feldes oder eine Tonprogramms benutzt werden oder als [illegible] einer Kollisions-Erfassungsroutine.

INK-NUMMERN

Der Schneider hat eine Palette von 27 Farbtönen, jedoch werden beim Vierfarbenmodus nur 4 für die Darstellung gewählt werden und bei 16-Farbenmodus nur 16. Die Farbtonzahl bezieht sich auf einen von 27 Farbtönen und die INK-ZAHL auf eine der 4 oder 16 INKs. Jede INK kann einem von 27 Farbtönen zugewiesen werden oder alle INKs könnten dem gleichen Farbton zugewiesen werden, das ist jedoch selten sinnvoll. LASER BASIC benutzt zwei Variablen, IK1 und IK2, die eingesetzt werden, um eine INK-Nummer, KEINE Farbtonnummer zu halten.

BILDPUNKTE

Der Bildschirm, auf dem Bilder dargestellt werden, ist in ein Bildpunktraster unterteilt. Alle Symbole und Sprites bestehen aus Bildpunkten. Im 4-Farbenmodus ist das Raster 200 hoch mal 320 breit, so daß es aus 62.000 Einzelpixeln besteht. Jwedes enthält einen Bildpunkt, der mit dem Farbton dargestellt wird, der sich in einer der vier INKs befindet. Man kann sagen, daß jede dieser 62.000 Zellen 0, 1, 2, oder 3 enthält. Wenn sie 0 enthält, zeigt sie den Farbton an, der INK 0 zugewiesen wurde usw. Beim 16-Farbenmodus ist das Raster 200 hoch aber nur 160 breit, so daß es aus nur 32.000 Einzelbildpunkten besteht. Diese sind nicht mehr "quadratische" Bildpunkte, sondern zweimal so breit wie hoch. Jeder Bildpunkt kann jetzt eine Nummer von 0 bis 15 enthalten und daher einen der 16 Farbtöne darstellen, die den 16 INKs zugewiesen wurden. Es ist wichtig, den Unterschied zwischen INK und Farbton zu verstehen, da sonst das restliche Handbuch sehr verwirrend ist. Wenn Sie daher nicht sicher sind, lesen Sie diese Definition und die vorhergehende noch einmal durch.

AUSGANGSDATEN

Mehrere Befehlsgruppen in Laser BASIC bewirken die Bewegung von Daten zwischen Sprites und Bildschirm. Die Ausgangsdaten beziehen sich auf die Daten, die bewegt werden.

ZIELDATEN

Diese beziehen sich auf Daten, die sich ursprünglich in dem Speicher-oder Bildschirmbereich befanden, in die die Ausgangsdaten bewegt werden.

BAND-/DISKETTENVERZEICHNIS

BANDPOSITION	BAND/DISKETTE DATEIBEZEICHNUNG	BEZEICHNUNG
Band 1 – Seite A	"LB"	Die Laser-BASIC-Erweiterung
	"SPT2SPR"	Die in den Beispielprogrammen benutzten Beispielspr
Band 1 – Seite B	"DEMO"	Die Laser-BASIC-Erweiterung (modifiziert)
	"DEMO"	Vorfuhrungsprogramm, das automatisch bei Betrieb s Sprites ladt.
	"SPT3SPR"	Die vom Vorfuhrprogramm benutzten Sprites.
	"SPT4SPR"	
	"SPT5SPR"	
Band 2 – Seite A	"SPTGEN"	Das Sprite-Generatorprogramm
	"SPT1SPR"	Die Arkadensprites

"MUSICSPR" Die Beispie lodien

BETRIEBSANLEITUNG

Die Laser-BASIC-Erweiterung

(i) Durch Eingabe von RUN"LB, gefolgt von ENTER, laden.

(ii) Laser BASIC lädt sich, läßt den Copyright-Hinweis erscheinen und wartet auf das Drücken einer Taste.

Das Vorführprogramm

(i) RUN"DEMO eingeben für die Bandversion.

oder RUN"LB eingeben – jede beliebige Taste drücken und dann RUN"DEMO für die Diskette.

Das Spritegenerator-Programm

(i) Zuerst Laser BASIC unter Verwendung des vorstehenden Verfahrens Laden.

(ii) Der Spritegenerator kann dann mit RUN"SPTGEN, gefolgt von ENTER geladen werden und ablaufen. LASER BASIC ist von Band 1 zu laden. Alle zur Zeit im Speicher befindlichen Sprites gehen verloren, und die maximale Sprite-Standardzahl ist 120. ANMERKUNG: Vor Eingabe von RUN sicherstellen, daß die Tastatur auf Großbuchstaben geschaltet ist (falls nicht, CAPS SHIFT drücken).

Der Tongenerator

(i) Zuerst Laser BASIC mit dem vorbeschriebenen Verfahren laden.

(ii) Dann kann der Tongenerator unter Verwendung von RUN"SNDGEN, gefolgt von ENTER geladen werden und ablaufen. Laser BASIC ist von Band 1 zu laden. Der Tongenerator kann an jeder Stelle "unterbrochen" und erneut betrieben werden. ANMERKUNG: Vor Eingabe von RUN sicherstellen, daß die Tastatur auf Großbuchstaben geschaltet ist (falls nicht, CAPS SHIFT drücken).

STARTEN MIT LASER BASIC

Zuerst das Gerät durch gleichzeitiges Drücken von SHIFT, CRTL und ESC rückstellen. Zum LADEN der Laser BASIC Erweiterung nur RUN"LB eingeben und ENTER drücken. Wenn Sie mit Band laden, müssen Sie auch WIEDERGABE am Kassettengerät drücken. Das Programm lädt sich und läßt den Copyright-Hinweis erscheinen. Jetzt drücken Sie jede beliebige Taste, worauf der Bildschirm gelöscht wird und in der oberen linken Ecke "Ready" erscheint. Bei diesem Übungslauf benutzen wir die Beispielsprites. Wenn Sie daher das Band verwenden, spulen Sie es nicht zurück. Jetzt eingeben:

```
AS="SPT2SPR":GSPR,@AS
```

Die Mustersprites sind jetzt geladen, und wir sind startbereit. Um sicher zu sein, halten Sie sich zur Zeit an die Beispiele im Text und geben Sie keine anderen Befehle von Laser oder Locomotive BASIC ein.

Bevor wir w...rgehen, erwähnen wir am besten kurz, wie Laser BASIC seine eigenen Variabl behandelt. Diese unterscheiden sich von normalen Variablen, und eine genaue Liste befindet sich Kapitel "Laser-BASIC-Spezialvariablen". Da wir damit beginnen, uns einmal die Bildschirmope tionen anzusehen, führen wir nur 5 Variablen ein – COL, ROW, LEN, HGT und SET. Laser BASIC l 16 Variablen-"SäTZE", und jeder SATZ besteht aus 17. Beginnen wir mit dem Definieren ein Bildschirmfensters under Verwendung von SET 0.

```
|SET,0:|COL,24:|ROW,64:|HGT,64:|LEN,8
```

Wir haben SET 0 gewählt und arbeiten weiter mit SET 0, bis wir zu einem anderen SET übergeh Sehen wir uns einmal zuerst das gewählte Bildschirmfenster an. Hierzu geben wir ein:

```
|INVV
```

Am Anfang war das Bildschirmfenster leer, aber jetzt "invertieren" wir es. Das bedeutet, da

die auf INK 0 gesetzten Bildpunkte jetzt auf INK 3 gesetzt sind
die auf INK 1 gesetzten Bildpunkte jetzt auf INK 2 gesetzt sind
die auf INK 2 gesetzten Bildpunkte jetzt auf INK 1 gesetzt sind
die auf INK 3 gesetzten Bildpunkte jetzt auf INK 0 gesetzt sind

Das Fenster war ursprünglich leer (alle Bildpunkte waren auf INK 0 gesetzt). Durch Ablauf von IN (invert video) haben wir also alle Bildpunkte auf INK 3 gesetzt, also rot. Wenn wir den Vorga wiederholen durch Eingabe von:

```
|INVV
```

erscheint das Fenster, denn alle Bildpunkte kehren zu ihrer Originalfarbe INK 0 zurück. Wiederho Sie das also noch einmal, so daß Sie das Fenster erneut sehen können – eingeben:

```
|SCLS:|INVV
```

Das Fenster ist jetzt wieder rot. Definieren wir einmal ein zweites Fenster im Originalfenste eingeben:

```
|SET,1:|COL,26:|ROW,80:|HGT,32:|LEN,4
```

Wir haben jetzt ein zweites Fenster im ersten definiert, und um es zu sehen, geben wir e

```
INVV
```

Das zweite Fenster ist jetzt klar sichtbar, denn alle Bildpunkte wurden von INK 3 wieder auf IN invertiert. Setzen wir jetzt einmal die Bildpunkte in diesem zweiten Fenster auf INK 1. Denken daran, daß wir zur Zeit mit SET 1 arbeiten. Um die Bildpunkte auf INK 1 zu setzen – einget

```
IK1,1::STCV
```

Denken Sie daran, daß der Befehl STCV (Bildschirmfarbesetzung) eine neue Variable IK1 benu die die INK-Zahl zum Farben des Fensters enthält. Versuchen wir jetzt einmal etwas Interessante einzugeben:

```
SET,0:FOR I%=1 TO 500:INVV:NEXT I%
```

Damit wir das von SET 0 definierte Fenster (das praktisch das von SET 1 definierte Fenster enth 500 mal invertiert. Bevor wir etwas nutzlicheres tun, sehen wir uns einmal ein weiteres Beispie den Einsatz des Variablensatzes an. Geben Sie folgendes ein:

```
SCLS:FOR I%=1 TO 50:SET,0:INVV:FOR J%=1 TO 10:|SET,1:|INVV:NEXT J%:NEXT I
```

der äußeren Schleife (Schleife I%) enthalten, das durch Invertieren des [illegible]
(das beide Fenster enthält) und ließ dann die innere Schleife ablaufen. Dieser Einsatz des Variablensatzes läßt Ihre Programme nicht nur viel sch[illegible]er ablaufen, sondern komprimiert auch Ihr Programm, denn statt aller vier (COL, ROW, LEN und HGT) wird nur eine Variable zugewiesen (der SET). Wir verlassen die Laser-BASIC-Variablen jetzt einmal, aber ihr Einsatz wird bei weiteren Beispielen deutlich.

BILDSCHIRMOPERATIONEN

Wir haben uns bereits eine Bildschirmoperation (INVV) angesehen, aber es gibt einige, die wir an den Bildschirmfenstern durchführen können, daher sehen wir uns einmal den Rest an. Zuerst machen wir zwei einfache Beobachtungen und erwähnten einige Gebote und Verbote.

1 Wenn Sie eine Zeile unten auf dem Bildschirm eingeben, werden Sie bemerken, daß Sie normal nach oben durchläuft. Das kommt daher, weil Laser BASIC den Bildschirmdurchlauf des Schneider zu einem "Software-Durchlauf" zwingt und nicht zu einem "Hardware-Durchlauf". Dies wird durch Benutzen eines Textfensters über die ganze Bildschirmbreite erzielt, das jedoch nur 24 Symbolzeilen hoch ist, anstatt der vollen 25. Die 25. Symbolzeile wird daher nie für den Text benutzt. Lassen Sie sich nicht zwischen den Textfenstern des Schneider und den Bildschirmfenstern von Laser BASIC verwirren. Das Textfenster, mit dem Laser BASIC arbeitet, benutzt Strom 0, wenn Sie daher Strom 0 für den ganzen Bildschirm einsetzen, müssen Sie darauf achten, daß er durch folgende Eingabe rückgestellt wird:

```
|SCLS
```

Das löscht den gesamten Bildschirm und setzt Strom 0 auf dem laufenden Schirm mit den Dimensionen, die Laser BASIC verwendet, ein. Wenn Sie den Bildschirm durch Strom 0 einen "Hardware-Durchlauf" ausführen lassen oder durch einen anderen Strom, lassen Sie stets SCLS ablaufen, bevor Sie die Befehle von Laser BASIC verwenden. Wir führen in den Probesitzungen keine Hardware-Durchläufe aus, so daß wir uns damit zur Zeit nicht befassen müssen.

2. Laser BASIC enthält Befehle, die in MODE 1 (4-farbige Betriebsart) oder MODE 0 (16-farbige Betriebsart) betrieben werden, aber für den Einsatz von MODE 2 besteht keine andere Beschränkung als daß gewisse Befehle von Laser BASIC in dieser Betriebsart nicht richtig eingesetzt werden können (siehe "BEFEHLSÜBERBLICK"). Ihr Programm kann ohne weiteres zwischen den Betriebsarten umschalten, jedoch sollten in Mode 2 nur gewisse Befehle von Laser BASIC benutzt werden. Die Befehle "MODE 0" und "MODE 1" werden von ihre Äquivalenten für Laser BASIC "ONLO bzw. ONHI" ersetzt. "MODE 0" und "MODE 1" funktionieren bei Locomotive BASIC noch normal, jedoch ist bei Laser BASIC "ONLO" und "ONHI" auszuführen,bevor das System weiß, daß die Betriebsart geändert wurde.

Gehen wir nun zu den Beispielen über und bringen den vordefinierten Sprite auf den Bildschirm, was genau genommen keine Bildschirmoperation, jedoch nötig ist, damit wir die Vorgänge, die wir ausführe, genau verstehen können. Eingeben:

```
|SET,0:|SCLS:|COL,40:|ROW,8:|SPN,6:|PTBL:|LEN,28:|HGT,32
```

Wir beginnen mit der Wahl von SET 0 und löschen den Bildschirm. Dann setzen wir das Ziel für das Fenster mit Spalte 40 (halb über den Bildschirm) und Zeile 8 (ein Symbol von oben) ein. Dann wählen wir Sprite 6 und bringen es mit PTBL auf den Bildschirm. Abschließend setzen wir ein Bildschirmfenster un das Sprite mit den gleichen Maßen wie das Sprite (32 Bildpunkte hoch und 28 Bytes breit). Es ist zu bemerken daß 32 Bildpunkte hoch 4 Symbole bedeutet, und in 4-farbiger Betriebsart bedeutet 28 Bytes breit 14 Symbole breit. Nachdem wir ein Bildschirmfenster mit bedeutungsvollen Daten gefullt haben, können wir jetzt Operationen darin ausführen. Es ist auch zu bemerken, daß sich die Operationen, die wir an diesen Daten ausfuhren, keinesfalls auf Sprite 6 auswirken, das sich noch im Speicher befindet. Beginnen wir einmal, indem wir uns Durchlaufvorgange ansehen. Wir lassen das Fenster um einen Bildpunkt nach rechts durchlaufen. In 4-farbiger Betriebsart kommen auf ein Byte 4 Bildpunkte (8 pro Symbol). Wir lassen rollen mit Up in [illegible]

Symbol kommen, könnten wir das ganze [illegible]
wir den Vorgang 112 mal wiederholen, und da wir das Bild bereits einmal durchlaufen [illegible]
ENTER dreim[illegible]rücken, um den Textkursor nach unten zu bringen, dann eingeben:

```
FOR I%=1 TO 111:|WVR1:NEXT I%
```

Wir hätten übrigens das gleiche Ergebnis eleganter erzeilt, ohne die relativ langsame FOR-NEXT Schleife zu benutzen. Gewisse Befehle von Laser BASIC, für die Durchläufe (Scrolls) ein Beispie sind, können in eine Maschinencode-Schleife ausgeführt werden. Es gibt zwei verschieden Maschinencode-Schleifen, solche, die auf Bild-Horizontalrücklauf synchronisieren und die, die e nicht tun. Bild-Horizontalrücklauf ist für den Neuling etwas, was 50 mal pro Sekunde geschieht, wen der "Punkt", der Ihr Fernsehbild erzeugt, das Abtasten des Bildschirms beendet und dann "horizont zurückläuft", um den nächsten Abtastvorgang zu beginnen. Durch Synchronisieren Ihres Betriebe auf Bild-Horizontalrücklauf kann normalerweise eine viel glattere Bewegung erzielt werden. Sie i natürlich nicht so schnell, aber 50 mal pro Sekunde ist für die meisten Anwendungen schnell genu Um den Befehl in einer Maschinencode-Schleife auszuführen, müssen Sie den Befehl mit zw Parametern verfolgen. Das kann durch gültige BASIC-Ausdrücke erfolgen. Der erste Paramet definiert die Häufigkeit für die Ausfuhrung des Befehls, während der zweite diktiert, ob die Schle zwischen den Ausführungen auf Bild-Horizontalrücklauf wartet. Wenn der Wert des zweit Ausdruckes 0 ist, wird die Ausführung nicht synchronisiert. Wenn es sich aber um einen ander Wert handelt, erfolgt die Synchronisierung. Sehen wir uns einmal ein Beispiel an, das unser Fens nach links durchlaufen läßt, um 112 Bildpunkte ohne Horizontalrücklauf und es dann invertiert u den Horizontalrücklauf wiederholt.

```
FOR I%=0 TO 1:|WVLI,112,I%:|INVV:NEXT I%
```

Insgesamt gibt es übrigens 12 Bildschirmdurchläufe. Hierbei gilt folgende Syntax:

Erstes Symbol :	"W" für Durchlauf oder "S" kein Durchlauf.
Zweites Symbol :	Stets "V" für Bildschirm.
Drittes Symbol :	"R" gleich rechts oder "L" gleich links.
Viertes Symbol :	"1" gleich Bildpunkt, "4" gleich 1 Byte, "8" gleich 2 Bytes.

Befehle:
SVR1, SVL1, SVR4, SVL4, SVR8, SVL8
WVR1, WVL1, WVR4, WVL4, WVR8, WVL8

Sehen wir uns einmal einen Durchlauf ohne Umlauf an. Wenn wir unser Fenster ohne Ur durchlaufen lassen, verlieren wir natürlich die Bildschirmfensterdaten. Vergrößern wir einma Fenster nach rechts und lassen das Sprite ohne Umlauf diesmal in größeren Stufen von 1 Byte Symbol in 4-farbiger Betriebsart, 1/4 Symbol in 16-farbiger Betriebsart) umlaufen. Eingebe

```
|SCLS:|COL,0:|LEN,56:|PTBL:|SVR4,56,1
```

Denken Sie daran, daß das 56-malige Wiederholen von SVR4 genug war, um das Sprite gar dem Fenster rollen zu lassen, denn jeder Durchlauf betrug 1 Byte (4 Bildpunkte). Die Bewegur diesmal viel schneller f, und weil das Sprite viel großer war, schien es auf halben Wege nach zu flimmern. Dies geschah, weil der "Punkt" halbwegs durch den Durchlauf auftrat, und die später genauer erklart. Bevor wir weitergehen können, müssen wir das Sprite mit PUT wieder a Bildschirm bringen. Diesmal bringen wir 2 Sprites durch PUT in unser Fenster, da es jetzt 56 breit ist. Wir bringen das Sprite durch PUT in die linke Fenstersetle, lassen das Sprite nach umlaufen und bringen das Sprite durch PUT wieder so hinein, daß wir das Fenster mit 2 Kopi Sprite zweimal fullen. Funfmal ENTER eingeben, um den Cursor zu bewegen, dann ein

```
FOR I%=1 TO 2:PTBL:WVR8,14,1:NEXT I%
```

Es ist zu bemerken, daß WVR8 die Daten um 2 Bytes bewegt, so daß es 14 mal zu wiederho um 28 Bytes Platz (4 Symbole in 4-farbiger Betriebsart) zum Einsetzen des Sprite durch I

Ab... horizontalen Durchlauf des vorherigen Beisp... gibt es auch zwei Befehle für den senkrechten Durchlauf. WVVN und SVVN lassen senkrecht ... und ohne Umlauf durchlaufen. An dieser Stelle führen wir eine neue Variable, NPX (Anzahl der Bildpunkte) ein. Wenn NPX ein positiver Wert ist, läßt er die Daten nach oben umlaufen, und wenn NPX negative ist, laufen die Daten nach unten um. Diese Befehle werden wiederum nur einmal ausgeführt, wenn keine Parameter folgen oder werden wiederholt ausgeführt, wenn Sie von zwei Parametern gefolgt werden. Eingeben:

```
FOR I%=16 TO -16 STEP -1 :|NPX,I%:|WVVN,8,1:NEXT I%
```

Dieses Beispiel beginnt mit dem Durchlaufen des Bildschirmfensters nach oben mit Umlauf. Die Durchlaufgeschwindigkeit verlangsamt sich und beginnt schließlich den Abwärtsdurchlauf. Der Abwärtsdurchlauf beschleunigt sich wiederum und hält dann an, wobei der Fensterinhalt so verbleibt, wie er vor der Operation war.

Kommen wir nun zu ungewöhnlicheren Bildschirmoperation, jedoch mit dem gleichen Fenster. Beginnen wir, indem wir uns X-Expansion ansehen. MGXV vergrößert die linke Hälfte des Bildschirmfensters so, daß sie das ganze Fenster ausfüllt – eingeben:

```
|MGXV
```

Es ist zu bemerken, daß die Daten in der rechten Hälfte durch die expandierten Daten der linken ersetzt wurden und verloren gingen. Wir können natürlich die Operation so oft wir wünschen wiederholen, und das Fenster expandiert weiter, unabhängig davon ,was sich in der linken Hälfte befindet. Geben wir also wieder ein:

```
|MGXV
```

Das Originalbild des Sprite wurde zweimal vergrößert und füllt jetzt das Fenster. In der gleichen Weise können wir die gleiche Operation in senkrechter Richtung ausführen – eingeben:

```
|MGYV
```

Diesmal wurde die obere Fensterhälfte senkrecht expandiert, um das ganze Fenster zu füllen. Die Daten in der oberen Hälfte gehen wiederum verloren, und wir könnten die Operation wieder so oft wir es wünschen wiederholen.

Jetzt löschen wir alle Daten im Fenster, bevor wir uns andere Bildschirmoperationen ansehen – eingeben:

```
|CLSV
```

Dieser Befehl löscht allen Daten im Fenster zu INK 0. Bevor wir weitergehen, bringen wir das Sprite durch PUT wieder auf den Bildschirm und verkleinern das Fenster, so daß es wieder der Breite des Sprite entspricht – eingeben:

```
|SCLS:|PTBL:|LEN,28
```

dann 5 mal ENTER drücken.

Das löscht den gesamten Bildschirm, bringt den Cursor nach oben links ƒ, wodurch das Sprite löschend überschreiben wurde, so daß es 5 Zeilen nach unten bewegt wird.

Sehen wir uns jetzt einmal die Spiegelbildbefehle an – eingeben:

```
|MIRV
```

Sie werden feststellen, daß das Fenster von links nach rechts spiegelbildlich dargestellt wurde. Wir können natürlich das gleiche senkrecht tun, hierzu geben wir ein:

```
|FIPV
```

Das Fenster wurde jetzt von oben nach unten spiegelbildlich dargestellt. Bevor wir w... ...len, verbreitern wir das F... ...ler und wiederholen den Vorgang zweimal – eingeben:

```
|MIRV:|LEN,56:|MIRV
```

Es ist zu bemerken, daß keine Daten wirklich verloren gehen, denn alle Daten linke vom derzeitigen Fesnter wurden auf die rechte Seite übertragen und umgekehrt. Es ist in der Tat eine zweite spiegelbildliche Darstellung, bei der Daten von der linken Hälfte zu der rechten Hälfte dargestellt werden aber keine Daten von der rechten Hälfte in der linken Hälfte. Es entsteht also ein symmetrisches Fenster des halben Bildes. Die beste Möglichkeit, dies zu zeigen, ist ein Beispiel, daher eingeben:

```
|SCLS:|PTBL:|LEN,14:|MORV:|LEN,28:|MORV
```

und wiederum ENTER fünfmal drücken.

Nach Löschen des Bildschirms wird das Sprite in das linke Fensterviertel gebracht. Die Fensterlänge wird dann auf 14 gesetzt, und durch MORV entsteht einen spiegelbildliche Kopie der linken Hälfte in der rechten Hälfte des 14 Byte breiten Fensters. Die Länge wird dann auf 28 erhöht, und der Vorgang wird mit dem dargestellten Ergebnis wiederholt. Wir können übrigens das gleiche senkrecht unter Verwendung von FOPV tun. Um die Ergebnisse zu sehen, eingeben:

```
|FOPV
```

Sie werden feststellen, daß das Endergebnis von oben nach unten symmetrisch ist. Wenn Sie diese Befehle mit den Durchlaufbefehlen kombinieren, können interessante Effekte entstehen. Geben Sie folgendes ein:

```
!SET,1:|COL,14:|ROW,8:|LEN,14:|HGT,16
```

Damit wird ein Fenster oben links im Fenster eingerichtet, das von SET 0 definiert wird. Jetzt eingeben:

```
|NPX,2;FOR I%=1 TO 1024:|SET,1:|WVL1:|WVVN:|SET,0:|MIRV:|MORV:|FOPV:|INVV:NEXT I%
```

Damit entsteht ein Kaleidoskopeffekt, der natürlich stark verbessert könnte, und das überlassen wir dem Benutzer zu einem späteren Zeitpunkt als Übung. Im Moment kommen wir zu weiteren Bildschirmoperationen. Sehen wir uns einmal SPNV an, das das derzeitige Bildschirmfenster um 90 Grad in Uhrzeigersinn dreht. Wir müssen hier 2 neue Variablen einführen, SCL und SRW, die allgemein benutzt werden, um die Spalte und Linienposition eines Spritefensters innerhalb eines Sprite anzugeben, die jedoch auch in diesem einen Fall als Spalte und Zeile des Ziels für die Drehung dienen. Drehen wir das Fenster, an dem wir gearbeitet haben, einmal und senden wir die Ergebnisse zur rechten Seite desselben – eingeben:

```
!SCLS:|PTBL
```

dann 5 mal ENTER drücken und eingeben:

```
SCL,72:!SRW,8:!SPNV
```

SPNV ist der eine Befehl, der nur im 4-Farbenmodus ausgeführt werden kann, da ihn die Rechteckform der Bildpunkte im 16 Farbenmodus sinnlos macht. Es ist auch zu bemergken, daß die Höhe des gedrehten Fensters nach unten auf das nächste Vielfache von Bildpunkten abgerundet wird.

Das berücksichtigt alle Bildschirmfenster-Transformationen, aber es sind hier noch 3 weitere Bildschirmoperationen zu erwähnen. Diese beziehen sich auf die Färbung in bestimmten Bereichen des Fensters. Sehen wir uns zuerst einmal FILL an. Hierbei handelt es sich um ein etwas ungewöhnliches FILL-Dienstprogramm, das einen langsamen, aber für des Speicher leistungsfähigen Algorithmus zum Füllen benutzt. Auch hier verdeutlichen wir dies am besten durch ein Beispiel – eingeben:

|COL,22:|ROW,104:|HGT,42:|LEN,36:|INVV
LOCATE 13,16:PRINT"THE FILL COMMAND"

Es ist hier darauf hinzuweisen, daß FILL die Bildschirmfenster unberücksichtigt läßt und so lang füllt, bis es entweder zum Bildschirmrand kommt oder zu einer anderen Farbe nach Anfang de Füllung. Wenn daher die Bildpunkte an der Stelle, wo der Füllvorgang begann, rot wären, würde nu der rote Bereich gefüllt, der den Bildpunkt enthält. Die Farbe, mit der der Bereich zu füllen ist, befinde sich in der Variablen IK1. Zum FÜLLEN des Fensters, das wir definiert haben, könnten wir an eine Stelle beginnen, jedoch FÜLLEN wir den Hintergrund für das Sprite und beginnen oben links a Fenster. Auch hier führen wir wieder eine neue Variable ein – XCL, bei der es sich um die horizontal Position handelt. Diesmal wird diese in Bildpunkten gemessen und nicht in Bytes. Bei 4-farbige Betriebsart besteht Spalte 32 aus Bildpunkten mit den X-Werten 128, 129, 130 und 131. Be 16-farbiger Betriebsart würde Spalte 32 nur aus Bildpunkten mit dem X-Wert 64 und 65 bestehe Das kommt daher, daß zwar jedes Byte 4 Bildpunkte in 4-farbiger Betriebsart darstellt, aber nur Bildpunkte in 16-farbiger Betriebsart. FILL benutzt ROW als seine senkrechte Koordinate, und i beiden Betriebsarten befindet es sich im Bereich 0 bis 199, wie dies stets der Fall war. Um daher de Bereich zu FÜLLEN, brauchen wir nur einzugeben:

|SET,1:|IK1,2:|XCL,165:|ROW,125:|FILL

Für diejenigen unter Ihnen, die interessiert sind, stammt diese ungewöhnliche Weise, in der diese FÜLLEN ausgeführt wird, vom benutzten Algorithmus – "im entgegengesetzten Uhrzeigersin aufzeichnen, falls möglich". Hier sollte erwähnt werden, daß bei FILL weitgehender Gebrauch vor Maschinenstapel gemacht wird. Als allgemeine Richtline braucht man mehr Platz, je komplizierter de zu füllende Gegenstand ist. Wenn der Platz nicht ausreicht, kann die Ausführung auf halbem Weg angehalten werden, und es erscheint **OUT OF MEMORY**.

Wir kehren jetzt zu einem Befehl zurück, der von einem Fenster bestimmt wird, das im laufenden SE definiert wurde, bei dem es sich in diesem Fall noch um SET 0 handeln könnte. Wir ändern jetzt all Bildpunkte im Fenster, die INK 0 haben, auf INK 2 (damit werden alle Bereiche gefärbt, zu dene FILL nicht gelangte!) Wir führen eine weitere Variable, IK2, ein, bei der es sich um den neuen Farbto handelt, wobei IK1 jetzt als Ausgangsfarbe benutzt wird. Das Fenster ist bereits definiert, so daß w die Variablen IK1 und IK2 setzen und dann den Befehl ausführen können, der SETV ist. Eingeben

|SET,0:|IK1,0:|IK2,2:|SETV

Es ist wichtig zu bemerken, daß dies nicht die gleiche Wirkung wie die änderung der Farbtöne in de BASIC-Palette hat, denn anstatt der Farbe, in der ein INK dargestellt wird, ändern wir die INK-Za von bestimmten Bildpunkten im Fenster. Die Ausführung dieses Befehls reduziert häufig die ir Fenster dargestellte Farbzahl, und nach Ausführung kann ein Fenster das den Bildpunkt mit allen (oder 16) Farbtönen enthielt, nur 3 (oder 15) bestimmte Farbtöne enthalten. Die genaue Wirkun dieses Befehls wird bei Einsatz klar.

Schließlich ist der letzte Befehl in diesem Kapitel ein Befehl, auf den wir kurz zu Beginn des Kapitel zu sprechen kamen – STCV. Dieser setzt jeden Bildpunkt im Fenster auf die INK-Zahl in IK1, das w bei diesem Beispiel auf 2 setzen. Eingeben:

|IK1,2:|STCV

Damit ist die Einfuhrung zu den Bildschirmoperationen abgeschlossen. Vielleicht möchten Sie m den Befehlen, die wir Ihnen soweit vorgestellt haben, experimentieren, bevor Sie zum nächte Kapitel übergehen, das sich mit Spriteoperationen befaß.

SPRITEOPERATIONEN

Unter all den Befehlen, die wir bisher berücksichtigt haben, gab es keinen, der sich auf ein Sprite Speicher ausgewirkt haben wurde Mit zwei Ausnahmen (FILL und SPNV) können aber al Operationen, die wir bisher betrachtet haben, auch an Sprites im Speicher ausgeführt werden. diesem Fall ist das Ergebnis der Operation nicht eher sichtbar, bis das Sprite durch Verwendung v "PUT" auf den Bildschirm gebracht wird Die Spriteoperationen sind ähnlich wie die Bildschirmoper n, daher behandeln Sie diese Befehle kurzer als die Aquivalentbefehle fur "BILDSCHIRM"

Hinsicht, im Gegensatz zu den Bildschirm
Spritebefehlen ein "S" nachgestellt. Im Gegensatz zum Bildschirmfenster, das von 4 Variablen definiert wurde – COL OW, LEN und HGT, wird das Sprite nur von einem definiert – SPN, der Spritezahl, für die die Operation gilt. Denken Sie daran, daß diese Befehle einige der Sprites im Speicher ändert, so daß Sie diese nach dieser Sitzung laden wollen, wenn Sie glauben, daß sich eines der Sprites, das Sie benutzen wollen geändert haben könnte. Erneutes Laden während dieses Kapitels kann sich jedoch auf die Ausführung gewisser Beispiele auswirken und ist zu vermeiden.

Sie haben wahrscheinlich genug von Sprite 2 und 6, daher verwenden wir jetzt andere Sprites. Wir beginnen mit dem Löschen des Bildschirms und indem wir ein Sprite durch PUT in die Nähe der Mitte der oberen Bildschirmzeile bringen. Eingeben:

|SCLS:|SET,0:|COL,60:|ROW,8:|SPN,3:|PTBL

Obgleich wir PTBL benutzen, um das Sprite durch "PUT" auf den Bildschirm zu bringen, gibt es 6 verschiedenen Möglichkeiten, ein Sprite durch "PUT" auf den Bildschirm zu bringen, die jedoch in einem späteren Kapitel behandelt werden. Im Moment gehen wir die Spriteoperationen einmal in ähnlicher Reihenfolge durch wie im letzten Kapitel – eingeben:

|COL,68:|INVS:|PTBL

Wir haben das Sprite invertiert und ihn durch PUT neben das Original gebracht. Wenn wir das Sprite nicht mehr ändern, entsteht durch weitere PUT stets das invertierte Sprite – eingeben:

|COL,56:|PTBL

Das Sprite ist immer noch invertiert, jedoch können wir ihn leicht durch folgende Eingabe zu seinem früheren Zustand zurückbringen:

|INVS

Um zu beweisen, daß es im Speicher geändert wurde, sehen wir es uns noch einmal an – eingeben:

|PTBL

Nun benutzen wir die Spritedurchläufe, um einen reibungslosen Diagonaldurchlauf zu erhalten – eingeben.

!NPX,1:FOR I%=1 TO 512:|WSVN:|WSL1:FOR J%= 40 TO 64 STEP 8:|COL,J%:|PTBL:NEXT J%:NEXT I%

Bei diesem Beispiel wird ein senkrechter und horizontaler Durchlauf kombiniert, so daß ein diagonaler Durchlauf entsteht, wodurch das Sprite wiederholt auf 4 benachbarte Bildschirmpositionen gerät. Eine ähnliche Wirkung wurde bei Durchlauf von 4 benachbarten Bildschirmfenstern erzielt worden sein, aber das Ergebnis wäre eine Zackenbewegung.

Wir können wieder einen Kaleidoskopeffekt erzeugen, diesmal unter Verwenduhg eines Sprite. Geben Sie das folgende kurze Programm ein und lassen Sie es laufen:

```
5 ' KALEIDOSKOP
10 |SCLS:|CLLO:|COL,20:|SPN,1:|PTBL
20 |NPX,1:|COL,40
30 FOR I=1 TO 256
40   |ROW,8 :|PTBL:|MIRS
50   |COL,44:|PTBL:|FIPS
60   |ROW,24:|PTBL:|MIRS
70   |COL,40:|PTBL:|FIPS
80   |WSVN:|WSL1
90 NEXT I
```

Wirkung von PTOR und GTOR sind nicht ganz so einfach erklären, und wenn Sie nichts von Booleescher Algebra und Binärzahlen verstehen, wäre langfristig gesehen, eine kurze Mathematikstunde sehr vorteilhaft. Wenn Sie diese jedoch nicht benötigen, überschlagen Sie das nächste Kapitel.

Alle Informationen im Schneider (oder in jedem anderen Mikrocomputer von 8 Bit) werden in mehreren "Bytes" gespeichert. Ein Byte enthält 8 "Bits", und jedes Bit kann eine 1 oder 0 enthalten. Ein Spritebyte oder Symboldaten im Schneider stellen 4 Bildpunkte in 4-farbiger Betriebsart dar oder 2 Bildpunkte in 16-farbiger Betriebsart. Das bedeutet, daß ein Bildpunkt in 4-farbiger Betriebsart 2 Bit enthält und 2 Bildpunkte in 16-farbiger Betriebsart 4 Bit. Fassen wir das einmal nachstehend zusammen, in 4-farbiger Betriebsart.

Bildpunktzustand	INK-ZAHL
00	0
01	1
10	2
11	3

und in 16-farbiger Betriebsart:

Bildzustand	INK-ZAHL
0000	0
0001	1
0010	2
0011	3
0100	4
0101	5
0110	6
0111	7
1000	8
1001	9
1010	10
1011	11
1100	12
1101	13
1110	14
1111	15

TABELLE 1

Was geschieht, wenn zwei Binärzahlen mit OR verbunden werden, kann beim nächsten Beispiel festgestellt werden. Nehmen wir einmal an, daß wir einen Bildpunkt mit INK 10 und einem Bildpunkt mit INK 12 durch 'OR' verbinden:

```
        1010        INK 10
OR      1100        INK 12

=       1110        INK 14
```

Wenn ein Bit im ersten Bildpunkt eingesetzt wird OR (oder) der zweite Bildpunkt, erscheint es im Ergebnis – so:

```
1 OR 1 = 1
0 OR 1 = 1
1 OR 0 = 1
0 OR 0 = 0
```


Nun sehen Sie sich einmal die vorstehende Summe erneut an und stellen Sie fest, ob Sie wissen, warum INK 10 OR INK 12 = INK 14 ist. Wir haben hier nicht genug Platz, um die ganze Tabelle auszudrucken (es gibt 256 Möglichkeiten!), aber Sie können bei Durcharbeiten der nachstehenden 4 Beispiele zu den gleichen Ergebnissen kommen.

```
INK 5  OR INK 8  = INK 13
INK 7  OR INK 9  = INK 15
INK 11 OR INK 4  = INK 15
INK 5  OR INK 13 = INK 13
```

Wenn Sie sich fragen, wie sich das alles auf die Grafik anwenden läßt, sehen wir uns einmal einige Beispiele an – eingeben:

```
|SCLS:|SPN,54:|COL,40:|ROW,8:|PTBL:|COL,42:|ROW,16:|SPN,53 :|PTOR
```

Der einzige Unterschied zwischen diesem Beispiel und dem vorherigen besteht übrigens daraus, daß das zweite Sprite mit dem ersten OR behandelt wurde anstatt mit "Block PUT". Sie werden dabei feststellen, daß nichts vom ersten Sprite durch die Blankoteile des zweiten Sprites gelöscht wurde, denn alles, was durch OR mit 0 verbunden wurde, bleibt unverändert. PTOR und GTOR haben übrigens weniger Anwendungen als die anderen Moglichkeiten, aber eine ziemlich nützliche Eigenschaft ist die, daß eine geeignete Wahl der INK-Zahl den Daten Anzeigerprioritäten einräumt. Wenn ein Sprite mit INK 1 zum Beispiel zwei Bereiche von INK 4 bzw. 3 überlappt, wird das Sprite im früheren Bereich mit INK 5 dargestellt, erscheint jedoch nicht im Bereich mit INK 3, wodurch die Zweibereichsperspektive entsteht

Logische ANDs – 'ND'

Allen Bewegungen mit logischem AND (und) wird ein "ND" nachgestellt. Die Wirkung von PTND und GTND wird am besten durch weiteren Bezug auf Tabelle 1 erklärt. Wenn zwei Bildpunkte mit AND verbunden werden, bleiben nur die Bit eingesetzt, die in Quellen- und Zielbildpunkten eingesetzt waren. Was geschieht, wenn wir einen Bilkpunkt, das INK 10 enthält, logisch durch AND mit einem Bildpunkt mit INK 12 verbinden?

```
        1010        INK 10
AND     1100        INK 12

=       1000        INK 8
```

Wenn ein Bit im ersten Bildpunkt eingesetzt ist UND im zweiten Bildpunkt, wird es im Ergebnis eingesetzt – wie

```
1 AND 1 = 1
0 AND 1 = 0
1 AND 0 = 0
0 AND 0 = 0
```

Nun sehen wir uns einmal die vorstehende Summe erneut an und stellen fest, ob Sie herausfinden konnen, warum INK 10 und INK 12 = INK 8. Nachstehend einige weitere Beispiele, in diesem Fall von logischen AND-Operationen. Seien Sie sicher, daß Sie jedes Ergebnis verstehen, bevor Sie weitergehen.

```
INK  5 AND INK  8 = INK 0
INK  7 AND INK  9 = INK 1
INK 11 AND INK  4 = INK 0
INK  5 AND INK 13 = INK 5
```

Sehen wir uns einmal ein weiteres Beispiel an, bevor wir seine Anwendungen auf das Schreiben von Spielen besprechen – eingeben:

```
|SCLS:|SPN,53:|COL,40:|ROW,8:|PTBL:|COL,42:|ROW,16:|SPN,54 :|PTND
```

Jetzt stellen Sie fest, ob Sie das = 3

Die Hauptanwendung für die logische Verbindung mit AND ist die Kollisonserfassung, aber der kombinierte Gebrauch von AND und OR kann auch zu Perspek[illegible]effekten führen, wie im Abschnitt über OR besprochen. Die Kollisionserfassung ist eine komplizierte Sache, und sie wird in einem besonderen Kapitel später im Handbuch besprochen. Im Moment verwenden wir nur einen einfachen Fall. Nehmen wir einmal an, wir schrieben ein Spiel, wo die "Guten" mit den geradzahligen INKs gezeichnet würden und die "Bösen" mit ungeradzahligen INKS. Wenn wir eine Spriteattrappe (Blindsprite) einrichten und sie mit INK 1 füllen würden, und dann durch GTND zu einem Bildschirmbereich damit gingen, wäre das Sprite leer, wenn es keine "Guten" gabe, enthielte jedoch Daten, wenn "Böse" vorhanden wären. In einem späteren Kapitel besprechen wir, wie ein Sprite geprüft werden kann, um festzustellen, ob es Daten enthält.

Logisch XOR – 'XR'

Allen Bewegungen mit logischem XOR wird ein 'XR' vorgestellt. XOR bedeutet exklusiv-OR, und wir zeigen seinen Einsatz unter Bezugnahme auf Tabelle 1. Um dies so einfach wie möglich zu beschreiben, wenn benachbarte Bits in den beiden Bildpunkten gleich sind, d.h. beide 1 oder 0, ist das Ergebnis 0. Wenn benachbarte Bits in den Bildpunkten anders sind, d.h. wenn eins 0 und das andere 1 ist, ist das Ergebnis 1.

```
        1010        INK 10
XOR     1100        INK 12

=       0110        INK 6
```

Wenn die Bits die gleichen sind, ist das Ergebnis 0, wenn sie anders sind, ist es 1 – daher.

```
1 XOR 1 = 0
0 XOR 1 = 1
1 XOR 0 = 1
0 XOR 0 = 0
```

Nun sehen wir uns einmal die vorstehende Summe erneut an und überprüfen, ob Sie feststellen können, warum INK 10 XOR INK 12 = INK 6 ist. Nachstehend weitere vier Beispiele. Stellen Sie sicher, daß Sie jedes Ergebnis verstehen, bevor Sie weitergehen.

```
INK 5  XOR INK  8 = INK 13
INK 7  XOR INK  9 = INK 14
INK 11 XOR INK  4 = INK 15
INK 5  XOR INK 13 = INK 8
```

Um die XOR-Funktion in Operation zu sehen, geben Sie ein:

```
!SCLS: SPN,53:!COL,40: ROW,8: PTBL: COL,42: ROW,16: SPN,54 : PTXR
```

Obgleich das zweite Sprite das erste Sprite zerhackt hat, mogen Sie der Ansicht sein, daß es in gewisser Weise erkennbar ist als die beiden ersten Ergebnis. Nun kommen wir zum interessantesten und nutzlichsten Merkmal von XOR – geben Sie ein:

```
!PTXR
```

Wenn Sie das vorstehende Beispiel nicht abgetippt haben sollten Sie feststellen, daß das erste Sprite völlig wiederhergestellt wurde und [illegible] Sprite nicht sichtbar ist. Das kommt daher, daß XOR wie die Inversion, umkehrbar ist. Jetzt geben Sie wieder PTXR ein und notieren das Ergebnis. Dann geben Sie ein:

```
!SCLS: PTXR
```

Das geschieht, weil [illegible]
hat. Jetzt eingeben:

```
!PTXR
```

Sie sollten feststellen, daß das Sprite völlig verschwunden ist. Diese zweite Eigenschaft (ein direktes Ergebnis der ersten) bedeutet, daß wir die XOR-Operationen zum Erkennen der Muster verwenden können. Wenn wir das Sprite, auf das wir testen, in eine Spriteattrappe geben und GTXR benutzen und dann die Spriteattrappe abtasten, bedeutet ein Ergebnis von 0 eine genaue Übereinstimmung. Jedes andere Muster und die Attrappe würden gewisse Daten enthalten haben. Die XOR haben daher mehrere Anwendungen. Die erste Eigenschaft der Umkehrbarkeit kann benutzt werden, um Spritebilder zerstörungsfrei über die Bildschirmdaten zu bewegen (siehe XMOV, XMVJ, XBNC), und die zweite Eigenschaft kann zur Mustererkennung benutzt werden und damit zur Kollisonserfassung.

Das bringt das Kapitel über logische Datenbewegungen zum Abschluß, jedoch hat der Schneider zwei weitere PUT und GET, die wir im nächsten Kapitel besprechen.

Vorstellen – 'IF'

Die Möglichkeit, ein Sprite durch PUT vor Bildschirmdaten zu bringen, ist normalerweise nur mit Hardwaresprites verbunden. Wir glaubten, daßdiese Möglichkeit so nutzlich ist, daß sie dem Paket zugefügt werden mußte. Zu dieser Operation wird INK 0 mit einer besonderen Eigenschaft versehen – Transparenz. Das bedeutet, daß wenn wir ein PTIF ausführen, das ganze Sprite genau wie gezeichnet verschwindet, daß jedoch in den freien Bereichen (die INK 0 enthalten) die darunter befindlichen Bildschirmdaten durchscheinen. Das kann sehr einfach mit unseren beiden früheren Beispielsprites verdeutlicht werden – eingeben:

```
'SCLS:|SPN,53:|COL,40:|ROW,8:|PTBL:|COL,42:|ROW,28:|SPN,54 :|PTIF
```

In diesem Fall erscheint Sprite 54 völlig entwirrt, wobei Sprite 53 deutlich durchscheint. Wenn Sie PTIF auf andere daruber befindliche Sprites anwenden würden, würde der "Stapel" unbegrenzt "aus dem Bildschirm" wachsen. GTIF funktioniert genauso, aber diesmal haben die Bildschirmdaten Vorrang vor den Spritedaten, und das Ergebnis bleibt natürlich im Sprite. Wir können in gleicher Weise Daten durch PUT "hinter" die Bildschirmdaten bringen.

Setzen hinter – 'BH'

Dies beruht auf einer ähnlichen Idee wie bei den 'IF'-Befehlen, aber hierbei werden Daten von der Quelle am Ziel hinter die Daten gesetzt, und sie sind nur durch die transparenten (INK 0) Teile der Zieldaten sichtbar – eingeben:

```
'SCLS:|SPN,53:|COL,40:|ROW,8:|PTBL:|COL,38:|SPN,54:|ROW,4:|PTBH
```

Auch hier werden beim wiederholten Gebrauch von PTBH "Bilder gestapelt", diesmal jedoch "in den Bildschirm". GTBH funktioniert in der gleichen Weise wie PTBH, aber diesmal werden Daten durch "GOT" hinter die Spritedaten bewegt, und das Ergebnis bleibt im Sprite.

Zusammenfassend haben wir folgende 12 neue Befehle besprochen:

PBTL, PTOR, PTND, PTXR, PTIF, PTBH, GTBL, GTOR, GTND, GTXR, GTIF, GTBH.

Übrigens hat jeder dieser 12 Befehle einen Optionsparameter, den wir jetzt besprechen.

Kollisionszahlung mit GET und PUT

Es wäre nutzlich zu wissen, wenn wir die Spritebilder bewegen, ob an der Quelle andere Bilder mit dem Zielbild kollidieren, unabhangig von der auszufuhrenden Operation. Laser BASIC bietet diese Möglichkeit, jedoch wird sie als Wahlfunktion behalten, da ihr Entschluß die Befehlausfuhrung in gewisser Weise verlangsamt. Eine Kollision bedeutet, daß ein Bildpunkt, der keinen INK-Wert von 0 hat (d.h. das nicht transparent ist) mit einem Bildpunkt kollidiert, der ebenfalls keinen INK-Wert von 0

Der Einsatz ist ziemlich einfach. Wenn Sie PUT oder GET mit Kollisionserfassung ausführen wollen, folgt auf den Befehl ein Einzelparameter, bei dem es sich um die Adresse einer ganzzahligen BASIC-Variablen handelt (nachgestelltes %, oder sie befindet sich in einem DEFINT-Befehl). Wenn eine Kollision auftritt, wird die BASIC-Variable um eins erhöht, falls nicht, bleibt der Originalwert bestehen. Verdeutlichen wir einmal den Einsatz mit einem kurzen BASIC Programm. Jedes laufende Programm mit "NEW" beginnen, dann eingeben:

```
5 ' KOLLISIONSZAHLUNG
10 X%=0:Y%=0:!SCLS:!SPN,2
20 FOR I=1 TO 256
30  !COL,RND*76:!ROW,RND*180:!PTXR,@X%
40  IF X%<>Y% THEN !PTXR:Y%=X%
50 NEXT I
```

Zeile 10 Erklärt 2 ganzzahlige BASIC-Variablen, löscht den Bildschirm und gibt das Sprite an, das wir benutzen wollen.
Zeile 20 Richtet die Hauptschleife ein.
Zeile 30 Wählt regellose Werte für ROW und COL und bringt dann durch XOR das Sprite auf den Bildschirm. Wenn etwas darunter ist, wird X% um eins erhöht
Zeile 40 Stellt durch Vergleich durch Y% fest, ob X% erhöht wurde. Wenn sie sich unterscheiden, wurde eine Kollision festgestellt, und das Sprite wird zum zweiten Mal mit PTXR behandelt, so daß es sich löscht. Y% wird dann gleich X%, wiederum bereit für die nächste Schleife.
Zeile 50 Schleife zurück zu Zeile 20.

Drücken Sie den Wert X% aus, um festzustellen, wie viele Kollisionen aufgetreten sind. Es ist zu bemerken, daß falls X% größer als 32767 wäre, eine negative Zahl erscheinen wurde. Wenn übrigens genau 65536 Kollisionen auftreten würden, enthält X% 0. Besprechen wir jetzt einmal die anderen beiden Typen GET und PUT.

SPRITEFENSTER-/BILDSCHIRMOPERTIONEN

Die 12 Befehle in dieser Gruppe sind analog den bereits besprochenen 12 Befehlen, jedoch statt daß die Bewegung zwischen einem ganzen Sprite und dem Bildschirm stattfindet, findet die Bewegung zwischen einem Spritefenster und dem Bildschirm statt. Das Fenster wird von den Variablen SPN, SCL, SRW, HGT und LEN in genau der gleichen Weise definiert wie für die Spritefensteroperationen im letzten Kapitel. Die Kollisionserfassung und die 6 Bewegungsarten werden völlig unterstützt. Durch die große ähnlichkeit zwischen diesen und Sprite-/Bildschirmoperationen wird hiermit nur ein Beispiel angegeben. Das folgende Programm (zuerst NEW für alle vorher eingegebenen Programme eingeben) eingeben, dann RUN.

```
5 ' SPRITEFENSTER/SCHIRMOPERATIONEN
10 !SCLS:FOR I%=1 TO 4 : PRINT "ABCDEFGHIJKLMN":NEXT I%
20 !LEN,5 :!HGT,32:!SPN,6:!ROW,0:!SRW,0
30 !COL,0 :!SCL,0 :!PWBL
40 !COL,5 :!SCL,5 :!PWOR
50 !COL,10:!SCL,10:!PWND
60 !COL,15:!SCL,15:!PWXR
70 !COL,20:!SCL,20:!PWIF
80 !COL,25:!SCL,25:!PWBH
90 LOCATE 1,10
```

Zeile 10 Löscht den Bildschirm und druckt einen Textbereich in der Größe des Sprite, das wir benutzen.
Zeile 20 Höhe und Länge von Sprite 6 werden eingerichtet, und dessen Zeile und Spritezeile werden auf 0 gesetzt.
Zeile 30 Bringt das erste Sechstel des Sprite unter Verwendung von PTBL auf den Bildschirm
Zeile 40 bis 80 Setzt den Vorgang fort und bringt die restlichen fünf Sechstel auf den Bildschirm
Zeile 90 Bewegt den Textcursor nach unten

Damit ist das Kapitel zur Behandlung der Spritefenster-/Bildschirmoperationen abgeschlossen, wir brauchen nur noch eine weitere Bewegungsart zu besprechen, nämlich die Bewegung Sprite zum Spritefenster.

SPRITEFENSTER-/SPRITEOPERATIONEN

Durch diese dritte Gruppe GET und PUT werden wiederum die gleichen 6 Operationen behandelt wie bei den letzten zwei Gruppen besprochen. Die Kollisionserfassung wird auch in genau der gleichen Weise vorgenommen. Diesmal werden jedoch Daten zwischen einem ganzen Sprite (dessen Zahl in SP2 gehalten wird) und einem Spritefenster bewegt, das sich in einem zweiten Sprite befindet, dessen Zahl in SP1 gehalten wird. Die Maße des bewegten Datenblocks sind die des ganzen Sprite (Sprite SP2), und das Spritefenster wird durch die normalen Spritefenster-Variablen SCL und SRW in Sprite SP1 gebracht. Allen Befehlen in dieser Gruppe wird 'PM' oder 'GM' vorgestellt, im Gegensatz zu 'PW' und 'GW' oder 'PT' und 'GT'. Auch hier wird wiederum nur ein Beispiel wirklich benötigt, weil eine sehr große Ähnlichkeit zwischen diesen und den letzten zwei Gruppen besteht. Bei bestehenden Programmen "NEW" eingeben und dann die folgenden Zeilen sowie RUN.

```
5 ' SPRITE/SPRITEFENSTEROPERATIONEN
10  !SCLS :!SPN,4 :!HGT,16:!LEN,4:!CSPR:!SPN,5:!CSPR
20  !SPN,6:!COL,26:!ROW,84:!PTBL:FOR T=1 TO 100:NEXT T
30  FOR L=1 TO 2
40   FOR N=0 TO 6
50    !SRW,0 :!SP1,4 :!SP2,6 :!SCL,N*4:!GMBL:!SRW,16:!SP1,5:!GMBL
60    !SRW,0 :!SP1,5 :!PMBL
70    !SPN,6 :!COL,26:!ROW,84:!PTBL:FOR T=1 TO 100:NEXT T
80    !SRW,16:!SP1,4 :!PMBL
90    !SPN,6 :!COL,26:!ROW,84:!PTBL:FOR T=1 TO 100:NEXT T
100  NEXT N
110 NEXT L
```

Zeile 10 Löscht den Bildschirm und erzeugt Sprite 5 und 4.
Zeile 20 Bringt Sprite 6 (das OCEAN-ZEICHEN) auf den Bildschirm und pausiert.
Zeile 30 und 40 Sind FOR-NEXT Schleifen.
Zeile 50 Bringt durch GET 1/7 der obersten Hälfte von Sprite 6 in Sprite 4 und 1/7 der unteren Hälfte in Sprite 5, unter Verwendung von GMBL.
Zeile 60 Sprite 5 wird jetzt durch PUT in Sprite 6 gebracht, wobei die Daten von Sprite 4 aus dem Einsatz von PMBL stammten.
Zeile 70 Sprite 6 wird jetzt erneut auf den Bildschirm gebracht.
Zeile 80 Sprite 4 wird durch PUT in Sprite 6 gebracht, wobei die Daten von Sprite 5 aus der Verwendung von PMBL stammten.
Zeile 90 Sprite wird auf den Bildschirm gebracht.
Zeile 100 und 110 vervollständigen die Schleife.

SPRITEDIENSTPROGRAMME

Bevor wir zu exotischeren Eigenschaften von Laser BASIC übergehen, müssen wir uns noch eine Befehlsgruppe ansehen, die Schaffen, Streichen, Laden, Zwischenspeichern und Mischen von Sprites vornimmt. Wir beginnen, indem wir kurz die Art und Weise schildern, in der Sprites gespeichert werden. Hierbei handelt es sich nicht um wesentliche Informationen, und sie werden lediglich aus akademischem Interesse geliefert. Wenn Sie es wünschen, können Sie diese Kapitel überschlagen und zu den Befehlen selbst übergehen.

Spriteorganisation

Die Sprites sind in zwei konsekutiven Speicherblocks organisiert. Der untere Block ist eine Tabelle, die Informationen über die Spritedaten selbst enthält, die sich im oberen Speicherblock befinden. Für die Spritespeicherung gibt es fünf Hinweismarken.

Hinweis-marke	Anwendung
MBOT	Weist auf das erste freie Byte über BASIC. Durch Abziehen des letzten von den Sprites benutzten Byte kann der verfügbare freie Platz für Schaffen neuer Bytes oder zum Durchlauf des Zwischenspeichers berechnet werden (siehe FREE).
SPST	Diese weist auf das erste Byte der Spritedaten, bei dem es sich übrigens um das erste Byte nach Ende der Tabellendaten handelt.
STAB	Diese weist auf das erste Byte der Tabellendaten und ist übrigens die niedrigste Speicherposition, die zum Speichern des Sprites benutzt wird.
SPND	Diese weist auf das erste freie Byte nach Ende des Spriteplatzes und enthält allgemein dieses Zeichen &7000.
SMAX	Enthält die höchste für den Benutzer verfügbare Spritezahl. Durch Erhöhen des Wertes unter Verwendung von ESPR wird automatisch zusätzlicher Tabellenplatz zugewiesen.

Tabellendaten

Die ersten vier Bytes der Tabellendaten entsprechen Sprite 0 und sind für den Systemgebrauch reserviert. Die nächsten vier Bytes sind Informationen von Sprite 1, die nächsten vier Informationen von Sprite 2 usw. bis zu den letzten vier Bytes, die die Informationen für das letzte Sprite enthalten, deren Zahl in SMAX gehalten wird. Die Größe der Tabelle wird daher von SMAX bestimmt, unter Verwendung von 4x(SMAX+1) Bytes. Die Tabellengröße schwankt daher von 8 Bytes für nur ein Sprite bis 1024 Bytes für 255 Sprites. Wenn der Benutzer daher 32 Sprites wünschte, würde die Tabelle 132 Bytes benutzen, und jeder Versuch, ein Sprite zu schaffen oder zu zerstören, dessen Zahl über 32 läge, würde zu einem Fehler führen.

Jede Spriteeingabe in der Tabelle nimmt 4 Bytes ein. Die ersten beiden Bytes halten die Addresse der Spritedaten im Spritedatenblock. Wenn ein Sprite gelöscht oder noch nicht geschaffen wurde, enthalten diese beiden Bytes 0. Die nächsten beiden Bytes sind die Breite bzw. die Höhe des Sprite.

Spritedaten

Die Spritedaten selbst sind seriell organisiert. Das bedeutet, daß der oberen Bildpunktzeile die zweite Bildpunktzeile folgt und so weiter, bis zum Ende des Sprite. Wenn ein neues Sprite geschaffen wird, wird die Tabelle im Speicher nach unten bewegt, und Sprite gelöscht wird, bewegen sich alle darunter befindliche Sprites nach oben, um den Platz einzunehmen. Die Tabelle wird also nach oben bewegt und die Hinweismarken in der Tabelle werden entsprechend angeglichen.

Dienstprogrammbefehle im Detail

SSPR,e1,e2

Dieser Befehl richtet den Spriteplatz ein. Zuerst wird die Tabelle auf 0 gelöscht, und SPST weist auf SPND (keine Spritedaten vorhanden). Der Befehl hat zwei Parameter, die sich beide in jedem gültigen BASIC-Ausdruck befinden können. Der erste Parameter, e1, sagt dem System, für wie viele Sprites in der Tabelle Platz reserviert werden soll. Dieser Parameter kann jeden Wert von 1 bis 255 haben und später geändert werden. Der zweite Parameter, e2, sagt dem System, wo der Spriteplatz einzusetzen ist. e2 sollte übrigens den Wert des ersten Byte über dem Spriteplatz enthalten, der sich dann nach unten aufbaut. Wenn Sie Laser BASIC laden, ist dieser Wert HEX 7000. Damit werden die Sprites direkt unter den Laser-BASIC-Code gebraucht, und es besteht eigentlich nie die Notwendigkeit, dies zu ändern, so daß SSPR selten oder nie ausgeführt wird, außer wenn der Benutzer alle Sprites aus dem Speicher löschen möchte.

DSPR

Dieser Befehl löscht das Sprite, dessen Nummer sich in SPN befindet. Wenn das Sprite nicht bereits besteht, erscheint die Fehlermeldung "** SPN DOESN'T EXIST **". Der Speicher verkleinert sich [illegible] ben, um die gelöschten Bytes auszugleichen. Wenn Sie eine nicht umstellbare geschaffenen Sprites chronologisch geschaffenen Sprites umgestellt. Zum Löschen von Sprites eingeben:

|SPN,7:|DSPR

ESPR

Dieser Befehl ist vorgesehen, so daß die Tabellengröße in jeder Phase geändert werden kann. Wenn die Tabelle erweitert (SMAX erhöht) wird, nimmt sie nach unten zu, und dies wirkt sich nicht auf die im Speicher gehaltenen Sprites aus. Wenn die Tabelle jedoch schrumpft (SMAX verkleinert), werden alle zur Zeit geschaffenen Sprites mit einer Zahl, die über dem neuen Wert von SMAX liegt, gelöscht. ESPR kann daher als eine Art Blocklöschung benutzt werden. Dieser Befehl ist mit Vorsicht anzuwenden! ESPR benutzt SPN, um den neuen Wert von SMAX zu halten. Wenn SMAX auf 100 geändert werden soll, eingeben:

|SPN,100:|ESPR

CSPR

Dieser Befehl schafft ein Sprite im Speicher. Die vier Bytes Spriteinformationen werden in die Tabelle eingegeben, und die gesamte Tabelle wird dann im Speicher nach unten bewegt, um das neugeschaffen Sprite aufzunehmen. In dieser Phase enthält sie keine bedeutungsvollen Daten. Zum Laden des Sprite mit einem Bild ist ein GET-Befehl erforderlich. Wenn das Sprite, das wir schaffen wollen, bereits besteht, wird "** SPN ALREADY EXISTS **" angezeigt. Wenn ein Versuch gemacht wird, ein Sprite mit einer Zahl zu schaffen, die den laufenden Wert SMAX übersteigt, wird "** SPN TOO HIGH **" angezeigt. CSPR verwendet drei Parameter. Die Zahl des geschaffenen Sprite, die Breite des Sprite in Bytes und die Höhe des Sprite in Bildpunkten. Diese befinden sich in SPN, LEN bzw. HGT. Zum Schaffen von Sprite 15 mit einer Höhe von 64 Bildpunkten (8 Symbole) und einer Breite von 16 (8 Symbole im 4-Farbenmodus, 4 Symbole in 16-Farbenmodus Betriebsart) würden wir eingeben:

|SPN,15:|HGT,64:|LEN,16:|CSPR

Bei diesem Beispiel würde unser Sprite HGTxLEN=64x16=1024 Bytes verbrauchen.

WICHTIGER HINWEIS:

Achten Sie darauf, daß Sie jedesmal, wenn Sie mit dem MEMORY-Befehl den oberen Wert von BASIC angleichen, dem System sagen, daß Sie dies getan haben. Hierzu müssen Sie MSET,e1 verwenden, wobei e1 HIMEM+1 ist. Wenn wir den oberen Wert von BASIC auf 20000 dezimal andern würden, würden wir eingeben:

MEMORY 20000:|MSET,20001

Das System wurde jetzt von der änderung informiert, und der Befehl FREE kann benutzt werden um zu prüfen, daß genug Platz vorhanden ist, um ein neues Sprite zu schaffen. Ein einfacher Test zur Prüfung, ob unser letztes Beispiel (Schaffen eines Sprite von 64x16) funktioniert, sähe wie folgt aus:

X%=0: FREE,ta X%:IF X%<64*16 THEN PRINT "NO ROOM"

Wenn ein Versuch gemacht wird, ein Sprite zu schaffen, das zu groß ist, erscheint "OUT OF MEMORY", und es wird keine Maßnahme ergriffen.

Zwischenspeicherplatz

Der freie Platz zwischen dem oberen BASIC-Wert und dem unteren Spritewert wird auch von anderen Laser-BASIC-Befehlen benutzt, und es ist sehr darauf zu achten, daß genug freier Platz zur Ausführung besteht. Nachstehend finden Sie die Befehle, die Zwischenspeicherplatz benötigen [illegible]

WSVN
WPVN
WVVN

Das sind die senkrechten Durchlaufbefehle. Sie benötigt. NPX das Vielfache der Breite des durchlaufenden Bereichs Bytes. Zum Durchlauf eines Sprite von 12 Bytes Breite in jeder Richtung um 8 Bildpunkte benötigen Sie daher 96 Bildpunkte. Ein Fenster von 40 Bytes Breite x 5 Bildpunkte durchzulaufen würden Sie 200 Bytes benötigen.

Wir haben nun die Speicherverwaltung besprochen, so daß wir jetzt zu den restliche Sprite-Dienstprogrammbefehlen kommen.

RSPR,e1

Dieser Befehl stellt den gesamten Spriteplatz um. Das bedeutet, daß er die Tabelle und die Spritedaten bewegt und alle Eingaben in der Tabelle und die Spriteplatz-Hinweismarken angleicht. Dieser Befehl wird übrigens sehr selten benutzt, da durch Transformation nach oben von der Standardposition die Systemvariablen von Laser BASIC löschend überschrieben wurden und da sich durch Transformation nach unten der freie Platz reduzieren würde. Er hat jedoch kompliziertere Anwendungen, die über den Umfang dieses Kapitels hinausgehen. Er hat nur einen Parameter, e1, der Spriteraum zu einer höheren Adresse umgewandelt, und mit einem negativen Wert zu einer niedrigeren Adresse umgewandelt. Nehmen wir einmal an, wir möchten 256 Bytes für eine Maschinencode-Subroutine zum Laden unter Laser BASIC bei &6EFF reservieren. Dazu würden wir folgende Eingabe benutzen:

|RSPR,-256

|PSPR,@V$

Dieser Befehl nimmt die Zwischenspeicherung der laufenden Sprites auf Band oder Diskette vor. Er benutzt nur einen Parameter, und dieser wird zur Weitergabe der Dateibezeichnung benutzt. Diese Dateibezeichnung wird in einer BASIC-Reihenvariablen gespeichert, so daß ihr ein '@' vorgestellt werden muß, da sonst ein Fehler entsteht. Die Dateibezeichnung selbst ist in Großbuchstaben einzugeben und darf 8 Symbole "SPR" sein. Laser BASIC speichert übrigens die Sprites in drei Dateien. Die erste enthält die Systemvariablen STAB, SPST, SPND und SMAX. Die Dateibezeichnung wird von Laser BASIC so geändert, daß das "SPR" am Ende der Dateibezeichnung zu "SYS" wird. Die zweite Datei enthält die Tabelle, und diesmal wird die Dateibezeichnung so geändert, daß die letzten drei Symbole "TAB" werden. Schließlich werden die Spritedaten selbst zwischengespeichert, und diesmal ist die benutzte Dateibezeichnung die ursprünglich an den Befehl weitergegebene, d.h. sie endet mit "SPR". Wenn wir zum Beispiel die derzeitigen Sprites auf Band oder Diskette mit der Dateibezeichnung "TESTSPR" zwischenspeichern wollten, würden wir folgende Eingabe benutzen:

A$="TESTSPR":|PSPR,@A$

Damit würden drei Dateien zwischengespeichert: "TESTSYS", "TESTTAB" und "TESTSPR". Sie werden alle als binäre Dateien zwischengespeichert und benötigen kein Spezialformat.

GSPR,@V$

Dieser Befehl lädt die drei zwischengespeicherten Dateien unter Verwendung des Befehls PSPR. Auch hier wird nur ein Parameter benutzt, die Dateibezeichnung, und es gelten die gleichen Beschränkungen. Man kann nicht zufällig eine weitere binäre Datei laden (außer wenn sie auch mit "SPR" endet), denn die angegebene Dateibezeichnung muß mit "SPR" enden (oder "SPR.BAK"). Diese Beschränkung kann ärgerlich erscheinen, wenn Sie aber in der Lage wären, zufällig eine weitere binäre Datei zu laden, hatte dies ernste Folgen. Die drei Dateien werden geladen, und die Hinweismarken des Systems werden auf die geladenen Werte gesetzt (so daß das System genau so organisiert ist wie bei Zwischenspeicherung der Sprites), was bedeutet, daß die Sprites dort begannen und endeten, wo dies bei der Zwischenspeicherung der Fall war. Achten Sie darauf, daß Sie genug Platz zum Laden einer bestimmen Spritedatei haben, sonst erscheint "OUT OF MEMORY", und Ihre derzeitigen Sprites werden verstummelt. Zum Laden der Datei "TESTSPR" benutzen:

A$="TESTSPR":|GSPR,@A$

MSPR,@V$

Dieser Befehl verschmilzt eine bezeichnete Spritedatei mit den bereits im Speicher befindlichen Sprites. Der zur Zeit im Speicher befindliche Spriteplatz wird nach unten erweitert, um die verschmolzenen Sprites aufzunehmen. Die Systemvariablen werden zuerst geladen und überprüft. In dieser Phase wird, die geladene Tabelle mit der im Speicher befindlichen verglichen um sicherzustellen, daß nicht zwei Sprites mit der gleichen Zahl geschaffen werden. Wenn ein Sprite in der residenten und geladenen Tabelle besteht, endet die Ausführung mit einer Fehleranzeigen ** SPRITE ALREADY EXISTS **. Wenn der Versuch gemacht wird, ein Sprite mit einer Zahl zu verschmelzen, die größer als die residente SMAX ist, zeigt das System eine Fehlermeldung **SPN TOO HIGH** an. Wenn einmal festliegt, daß beide Tabellen bestimmte Sprites enthalten, werden sie praktisch verschmolzen, und die neuen Spritedaten selbst werden an das Ende der alten Spritedaten mit der entsprechenden Angleichung der Hinweismarken gehängt. Hierbei sind zwei Dinge sorgfältig zu berücksichtigen, wird der obere BASIC-Wert löschend überschrieben und zweitens: wenn ein Ladefehler zwischen den verschmolzenen Tabellen und den geladenen Sprites auftritt, haben Sie einen freien Spriteplatz und müssen von vorn beginnen. Stellen Sie daher sicher, daß Sie Kopien von beiden Spritedateien sicher auf Band oder Diskette zwischengespeichert haben!! Wenn Sie eine Datei mit der Bezeichnung "TESTSPR" mit den zur Zeit im Speicher befindlichen Sprites mischen wollen, geben Sie ein:

A$="TESTSPR":|MSPR,@A$

Übrigens, die Verschmelzung kann benutzt werden, um die Sprites zu zwingen, daß sie in jede gewünschte Adresse geladen werden. Nehmen wir einmal an, Sie möchten eine Spritedatei in HEX 6000 laden, und die maximale Spritezahl in der zu verschmelzenden Datei ist 32 (hier wird angenommen, daß Sie Ihre residenten Sprites nicht erhalten möchten). Dann geben Sie ein:

|SSPR,32,&6000:A$="TESTSTPR":|MSPR,@A$

Gelegentlich sind Sie nicht in der Lage, zwei Spritesätze zu verschmelzen, wegen der in Konflikt stehenden Zuweisung der Spritezahlen. Die nächsten beiden Befehle wurden zur Umgehung dieses Problems zur Verfügung gestellt.

RNUM

Dieser Befehl hat zwei Parameter, die sich in SP1 und SP2 befinden. SP1 wird eingesetzt, um die Zahl des Sprite zu halten, das neu numeriert wird, und SP2 wird eingesetzt, um die neue Nummer zu halten, die dem Sprite zugewiesen wird. Das alte Sprite wird gestrichen, und das neue Sprite wird geschaffen. Der Vorgang findet übrigens ganz innerhalb der Tabelle statt und wirkt sich nicht auf die Spritedaten aus. Wenn SP1 nicht besteht, entstehen Fehlermeldungen oder wenn Sprite SP2 bereits besteht oder wenn durch Neunumerierung ein Sprite mit einer Zahl geschaffen wird, die SMAX übersteigt.

ADNM

Hierbei handelt es sich um einen besonders nützlichen Befehl, wenn er in Verbindung mit der verschmelzung benutzt wird. Mit ihm kann der Zahl aller Sprites, die bisher geschafften wurden, ein Versatz zugefügt werden. Der Versatz kann übrigens negative sein, so daß bereits geschaffene Sprites in absteigender Folge neu numeriert werden können. SPN wird zur Aufnahme dieses Versatzes benutzt. Angenommen, es beständen 12 Sprites, die von 5 bis 16 numeriert sind. Zur Neunumerierung dieser Sprites, so daß sie mit 20 bis 31 numeriert sind, eingeben:

SPN,15: ADNM

Dieser Befehl wirkt sich nicht auf SMAX aus, auch nicht auf die Systemhinweismarken. Nehmen wir jetzt einmal an, daß wir die bereits neu numerierten Spritezahlen zu ihren Originalwerten zurückbringen möchten, d.h. wir reduzieren sie um 15. Wir haben dabei ein Problem, dann wir können SPN keine negative Nummer zuweisen. Alles was wir übrigens tun, ist, 15 durch 2 zu ergänzen, also 256-15. Hierzu eingeben

|SPN,256-15:|ADNM

Die Tatsache, daß SPN in diesem Fall als eine 2er Ergän gszahl behandelt wird, kann zu Verwirrungen führen, daher denken wir einmal etwas weiter darüber nach. In diesem Fall wird angenommen, daß SPN einen Wert im Bereich von -128 bis +127 enthält. Wenn wir daher SPN 128 zuweisen, wir er als -128 enthaltend behandelt. Das bedeutet, daß das größte positive Inkrement, das wir addieren können, 127 ist. Wenn wir ein negatives Inkrement benutzen wollen, ziehen wir es in der vorgenannten Weise vn 256 ab.

ISPR,@V1,@V2@V3,@V4

Dieser Befehl wird in verschiedenen Anwendungen zum Abfragen von sprite- und System-details benutzt. Vor Ausführung wird SPN mit der abzufragenden Anzahl Sprites geladen. Wenn das Sprite nicht vorhanden ist, wird SPN auf 0 gesetzt, und es werden keine bedeutungsvollen Informationen übertragen. Wenn jedoch das Sprite, dessen Zahl sich in SPN befindet, besteht, werden folgende Informationen angezeigt:

Die ganzzahlige BASIC-Variable V1 enthält STAB
Die ganzzahlige BASIC-Variable V2 enthält SPST
Die ganzzahlige BASIC-Variable V3 enthält SPND
Die ganzzahlige BASIC-Variable V4 enthält die Adresse der virtuellen seriellen Spritedaten.

Abgesehen davon werden HGT und LEN eingesetzt, die die Maße des abgefragten Sprite enthalten. Achten Sie darauf, daß Sie die "@ " vor den vier ganzzahligen BASIC-Variablen in der Parameterliste nicht vergessen. Wenn eine der vier Variablen nicht früher erklärt wurde, entsteht eine Fehlermeldung "Improper Argument".

```
5 ' BEISPIEL VON SPRITEABFRAGE
10  INPUT;"SPRITENUMMER";S%:PRINT
20  V1%=0:V2%=0:V3%=0:V4%=0:H%=0:L%=0
30  !SPN,S%:!ISPR,@V1%,@V2%,@V3%,@V4%
40  IF V4%=0 THEN PRINT "SPRITE ";S%;" NICHT VORHANDEN":GOTO 110
50  PRINT"STAB    =";V1%
60  PRINT"SPST    =";V2%
70  PRINT"SPND    =";V3%
80  PRINT"ADRESSE=";V4%
90  !HGTQ,@H%:PRINT"HOHE   =";H%
100 !LENQ,@L%:PRINT"LANGE =";L%
110 PRINT
120 GOTO 10
```

FREE,@V1

Die anderen Sprite-Dienstprogrammbefehle haben Spezialanwendungen, die wir in einem späteren Kapitel behandeln. Ein weiterer Befehl ist jedoch von unmittelbarem Interesse in diesem Kapitel – FREE. Dieser ist ähnlich wie das BASIC-Äquivalent FRE, bei dem es sich um eine Funktion handelt, die den freien Platz im BASIC-Bereich anzeigt. FREE zeigt den freien Platz zwischen der obersten Zeile von BASIC und der untersten Zeile der Sprites an. Wir haben bereits die Operationen besprochen, die diesen Bereich fur Arbeitsplatz nutzen (siehe CSPR), und es ist klug, den Wert im Auge zu halten, den dieser Befehl anzeigt. Es ist zu bemerken, daß der angegebene Wert nur genau ist, wenn die Systemvariable MBOT jedesmal dann aktualisiert worden ist (unter Verwendung von MSET), wenn der BASIC-Befehl MEMORY benutzt wird. Auch hier muß die benutzte BASIC Variable zum Halten des Ergebnisses vor der Ausfuhrung dieses Befehls erklart worden sein. Um festzustellen, wie viel Platz ist, wurden Sie benutzen

!FREE,@X% PRINT X%

Das bringt das Kapitel über die Sprite-Dienstprogramme zum Abschluß

Sie haben wahrsche inzwischen bemerkt, daß Laser BASIC ein ziemlich ungewöhnliches Regime zur Behandlung seiner Parameter benutzt. Es werden insgesamt 17 verschiedene Variablen eingesetzt und 16 verschiedene v+nsätze. Der Hauptgrund für diese Lösung ist die Reduktion unnötiger Ausdruckbewertungen, die allgemein mehr Prozessorzeit brauchen als der Befehl selbst. Meistens werden nur eine Parameter zwischen den Ausführungen geändert, und es lohnt sich nicht, unnötige Neuberechnungen vorzunehmen.

Dieses Verfahren spart auch Platz in der Quellendatei und bedeutet, daß die Grafikroutinen selbst sehr schnellen Zugriff zu ihren Daten haben. Der volle Variablensatz und die Beschreibungen sind im Kapitel "ERWEITERETE LASER-BASIC-BEFEHLE" unter der Überschrift PARAMETERBE-ZOGENE BEFEHLE" enthalten. Alle Variablenbezeichnungen enthalten drei Symbole und werden bei jeder Zuweisung auf Fehler im "Bereich" geprüft. Viele Variablen haben mehrere Anwendungen, und dies bedeutet, daß die Bereichtsprüfung nicht so streng ist, wie sie sein sollte. HGT und LEN zum Beispiel werden allgemein zur Angabe von Fensterabmessungen benutzt, finden jedoch auch Anwedung als Erhöhungen in MOVE-Befehlen und müssen daher gelegentlich negative Zahlen enthalten können.

Wir haben bereits zahlreiche Beispiele für Variablenzuweisungen gesehen. Jetzt sehen wir uns einmal an, wie wir die Variablen abfragen können. Genauso wie bei den 17 Zuweisungen gibt es auch 17 Abfragungen, und jede hat die gleiche Form – 'BAR', gefolgt von drei Symbolen der Variablenbezeichnung mit nachgestelltem 'Q' (Rückfrage – query), gefolgt von einem Parameter, bei dem es sich um die Adresse einer früher erklärten ganzzahligen BASIC-Variablen handeln muß. Zum Abfragen und Drucken der Bildschirmfenster-Maße konnten Sie eingeben:

!COLQ,@C%:|ROWQ,@R%:|LENQ,@L%:|HGTQ,@H%:PRINT C%,R%,L%,H%

Es ist zu bemerken, daß falls beim vorgenannten Beispiel eine der Variablen C%, R%, L% oder H% nicht erklärt worden wäre, eine Fehlermeldung "Improper Argument" entstände.

Abgesehen von der Zuweisung von Werten für die Variablen und Abfragen ihrer Werte, gibt es fünf weitere verwandte Befehle, die Platz sparen und die Programmausführung beschleunigen sollen.

EXXV

Jedesmal, wenn ein Laser-BASIC-Befehl ausgeführt wird, wählt er die benötigten Parameter aus einem von 16 Variablensätzen. Der laufende SET wird vom Wert in der Variablen diktiert, SET. Wenn Sie die Schneider-Befehle EVERY und AFTER benutzen, kann ein Problem entstehen. Wenn die Routine, die Sie unter Interrupt ausführen, eingegeben wird, ist der laufende Wert in SET unbestimmt. Das bedeutet, daß er bei Eingabe der Routine erhalten bleiben und bei Leeranweisung wiederhergestellt werden muß. Das könnte typisch wie folgt erreicht werden:

```
1000 SETQ,@X%:REM PRESERVE CURRENT SET
1010 SET,Y%.....


1100 SET,X% REM RESTORE SET
1110 RETURN
```

Dies bezieht sich auf drei Befehle, von denen jeder die Beurteilung und die Suche nach dem Variablenbereich vornimmt, was relativ zeitraubend ist. Um dieses Problem zu umgehen, benutzt LASER BASIC eine alternative SET-Variable. Bevor wir weitergehen, sehen wir uns einmal anhand eines Beispiels an, wie dies funktioniert. Geben Sie das folgende kurze Programm ein

```
10 !SCLS:!SET,0:!HGT,64:!LEN,8 :!COL,40:!ROW,
20 !EXXV:!SET,1:!HGT,32:!LEN,16:!COL,48:!ROW,
30 EVERY 20,0 GOSUB 50
40 !INVV:GOTO 40
50 !EXXV:!INVV:!EXXV:RETURN
```

Zeile 10 löscht den Bildschirm und richtet ein Fenster unter Verwendung von SET 0 ein.
Zeile 20 EXXV macht SET 0 zum Alternativsatz. SET 1 wird dann zum laufenden Satz, und es wird ein Fenster unter Verwendung von SET 1 definiert.
Zeile 30 EVERY GOSUB wird eingerichtet
Zeile 40 Diese führt nur wiederholt einen INVV Befehl aus, der das Fenster, das in SET 1 definiert ist, invertiert.
Zeile 50 Hierbei handelt es sich um eine Subroutine, die unter Interrupt abläuft. Sie tauscht die 'laufenden' und 'alternativen' SET-Variablen aus, invertiert das in SET 0 definierte Fenster, stellt die 'laufenden' und 'alternativen' SET-Variablen wieder her und ZEIGT AN.

EXXV wurde für den Gebrauch mit EVERY und AFTER vorgesehen, kann jedoch auch bei normalen Anwendungen Zeit und Speicherplatz sparen, besonders dann, wenn zwei SETs wiederholt ausgetauscht werden.

SWPS

Laser BASIC bietet 12 verschiedene Betriebsarten für die Spritebeweung, und in jedem Fall steht Bewegung durch vier Teilbilder zur Verfügung. Dies wird durch Einsatz einer Folge von vier Sprites erzielt, deren Nummern in SP1, SP2, SP3 und SP4 gehalten werden. Jedesmal, wenn eine Bewegung gemacht wird, werden die Nummern rundum verschoben, so daß SP1 zu SP2 wird, SP2 zu SP3, SP3 zu SP4 und SP4 zu SP1. Dies bedeutet, daß der Zyklus ständig in der Reihenfolge SP1, SP2, SP3, SP4, SP1, SP2... und so weiter wiederholt wird. Gelegentlich muß die Folge umgekehrt werden, d. h. SP4, SP3, SP2, SP1, SP4, SP3... usw. Der Befehl SWPS erzielt dies effektiv durch einfachen Austausch von SP2 und SP4. Die Folge läuft von diesem Punkt an umgekehrt und kann natürlich in jeder Phase erneut umgekehrt werden. Übrigens werden die 'Sprungbewegungen' automatisch bei SWPS jedesmal ausgeführt, wenn ein Sprung auftritt, und diese werden in einem späteren Kapitel beschrieben.

ASTV

Dieser Befehl weist die Daten in einem Sprite dem laufenden Variablen-SET zu. Dies spart nicht nur viel Platz in einem BASIC-Programm, sondern die Parameter können auch in einem Bruchteil der Zeit eingerichtet werden, wie dies bei der 'altmodischen' Art und Weise der Fall war. Er erhöht auch effektiv die Variablensatzzahl und ist daher besonders nützlich bei der Verfolgung von Spriteanwendungen. Der Befehl benutzt nur zwei Parameter. Den laufenden SET und SPN, die zur Aufnahme der Spritezahl eingesetzt werden, die die Variablen SET Daten enthalten, und das die Daten enthaltende Sprite sind mit einer Höhe von 1 und einer Breite von 20 zu schaffen, da sonst ein Bereichtsfehler angezeigt wird. Die Daten können direkt durch Verwendung von ISPR zum Finden der Adresse in das Sprite eingegeben werden und dann durch Suche der Daten aus den Datenanweisungen durch POKE. Es ist naturlich wirksamer, das Sprite-Generatorprogramm hierzu zu benutzen, da dies Quellencodeplatz im BASIC-Programm spart. Die meisten Zuweisungen benutzen 8 oder mehr Bytes. Um daher 6 Parameter zuzuweisen, benötigte man etwa 50 Bytes und ziemlich lange Prozessorzeit. Wenn die Daten aber in einem Sprite vorgespeichert würden, brauchte man dazu nur 20 Bytes und weniger Zeit als für eine Einzelzuweisung. Wenn wir zum Beispiel in Sprite 32 die Parameter gespeichert haben, die wir zur Zuweisung derselben zu SET 9 benotigen und wunschen, würden wir benutzen:

```
|SET,9:|SPN,32:|ASTV
```

AVTS

sich die Daten in die entgegengesetzte Richtung bewegen, d.h. vom Variablen-SET zum [illegible]. Auch dieser Befehl kann zur Erweiterung der Anzahl Variablen-SETs benutzt werden und komprimiert und beschleunigt bei geeigneter Anwendung die BASIC-Programme. Wie bereits erklärt, gibt es nur zwei Parameter, der laufende Satz und die Bestimmungsspritezahl. Auch hier ist das Sprite mit einer Höhe von 1 und einer Breite von 20 zu schaffen.

ESAV

Hierbei handelt es sich um eine Variierung der letzten beiden Befehle, mit der der laufende Variablen-SET gegen die im angegebenen Sprite enthaltenen Daten ausgetauscht werden kann. Die gleichen beiden Parameter werden benutzt, und wiederum muß das Sprite eine Höhe und eine Breite von 20 haben.

Zusammenfassend ist zu sagen, daß es angebracht ist, wenn Sie sich mit dem Gebrauch der Befehle EXXV, ASTV, ATVS und ESAV vertraut machen, da ihr leistungsfähiger Einsatz nicht nur Ihre Programmausführung beschleunigen sondern Ihnen auch viel Platz im BASIC Programm sparen kann. Es ist daher sicherlich der Mühe wert, einige Zeit mit dem Experimentieren zu verbringen, bevor Sie zum nächsten Kapitel übergehen.

MOVE-BEFEHLE IM DETAIL

Kommen wir jetzt einmal zu interessanteren Befehlen. Die Befehle, die wir in diesem Kapitel einführen, werden wahrscheinlich in Ihren Spielen häufiger auftreten als die anderen bisher besprochenen, so daß es sehr wichtig ist, daß Sie sie richtig meistern.

Wir sollten zuerst die vier Möglichkeiten betrachten, in denen Laser BASIC die Spritebewegung bewirken kann. Wir beginnen daher mit dem unkompliziertesten und arbeiten uns dann nach und nach zu den komplizierteren Systemen durch.

Löschendes überschreiben von Blocks.

Bei diesem System wird das Sprite bewegt, indem er sich einfach löschend überschreibt. Das bedeutet, daß sich das Sprite nicht zerstörungsfrei bewegt, und dieses Verfahren kann nicht verwendet werden, wenn sich Daten im Weg des Sprite befinden, die nicht zerstört werden sollten. Bei Situationen, wo ein Sprite nicht beschränkt wird, um sich zerstörungsfrei zu bewegen, ist diese Bewegungsart stets zu benutzen. Dieses Verfahren ist nicht nur zweimal so schnell wie andere (häufig zehnmal schneller!), sondern es erzeugt auch eine sehr glatte, flimmerfreie Bewegung an jeder beliebigen Stelle des Bildschirmes, ohne daß eine Synchronisation mit Bild-Horizontalrücklauf zu erfolgen hat. Die beliebtesten kommerziellen Spiele verwenden dieses Verfahren weitgehend, da häufig nur die Hauptperson (normalerweise vom Joystick gesteuert) zerstörungsfrei bewegen muß. Dieses Verfahren unterstützt die Kollisonserfassung nicht effektiv, aber die meisten Anwendungen, die mit diesem Verfahren arbeiten, benötigen es wiederum nicht.

Die einzigen Vorsichtsmaßnahme, die der Benutzer zu ergreifen hat, ist sicherzustellen, daß das Sprite um seinen Umfang einen (oder einen entsprechend gefärbten) Rand hat, so daß es sich völlig loschend überschreibt und keine Daten hinterläßt. Der Rand muß mindestens so groß wie die Erhohungen sein, um die sich das Sprite bewegt.

Sehen wir uns einmal ein Beispielprogramm an. Jedes Programm im Speicher mit 'NEW' beginnen, so daß Sie sicher sind, daß Sie die Mustersprites im Speicher haben, bevor Sie dieses Beispiel eingeben und betreiben.

```
5 ' BEISPIEL VON BLOCKUBERSCHREIBUNG
10 !SCLS
20 !SPN,8:!COL,40:!ROW,0
30 !HGT,1
40 !LEN,0
50 !SP1,8:!SP2,8:!SP3,8:!SP4,8
60 FOR I=0 TO 183
70  !WMOV
80   FOR W=1 TO 50 :NEXT W
90 NEXT I
```

Zeile 30 und 40 Die Höhe wird auf +1 gesetzt, so daß sich das Sprite um je 1 Bildpunkt nach unten bewegt, sobald WMOV ausgeführt wird. LEN wird auf 0 gesetzt, d. h. keine Horizontalbewegung.
Zeile 50 Die Variablen SP1, SP2, SP3 und SP4 werden mit Sprite 8 eingerichtet, d.h. keine Bewegung.
Zeile 60 Die Hauptschleife.
Zeile 70 Der WMOV.
Zeile 80 Da WMOV sehr schnell ist, wird eine Verzögerungsschleife eingerichtet.
Zeile 90 Schleife zurück zu Zeile 60.

Exclusive-OR.

Hierbei handelt es sich um das beliebteste Verfahren für die zerstörungsfreie Beweung von Sprites auf dem Bildschirm (wenn der in Frage stehenden Mikrocomputer keine Hardware-Sprites hat), und auf diese Weise erhält man ein sehr genaues 'Gefühl' für ein Spiel. Diejenigen unter Ihnen, die den Spectrum kennen, haben diese Bewegungsart in einer Reihe von Spielen gesehen, obgleich Sie sich vielleicht nicht dessen bewußt waren, was Sie wirklich sahen. Wenn Sie die Fallen kennen, gehen Sie zum nächsten Kapitel über. Wir zeigen hier die Wirkung in unserem nächsten Beispiel und verlangsamen es für Sie, so daß Sie es deutlich sehen können. Zuerst bewegen wir ein kleines Sprite über den Bildschirm, der auf Bild-Horizontalrücklauf synchronisiert ist.

```
5 ' BEISPIEL VON EXKLUSIV-OR
10 !SCLS
20 !SP1,8:!SP2,8:!SP3,8:!SP4,8:!SPN,1
30 FOR I=1 TO 100
40   !COL,RND%76: !ROW,RND%184:!PTIF
50 NEXT I
60 !COL,-10:!ROW,50:!HGT,1:!LEN,1
70 !XMOV,512,1
```

Zeile 10 Löscht den Bildschirm.
Zeile 20 Setzt alle Spritevariablen ein.
Zeile 30 bis 50 Bringt 100 Sprites auf regellose Positionen auf dem Bildschirm.
Zeile 60 Setzt die Ausgangsposition für XMOV ein und Höhe und Länge, die für die Diagonalbewegung eingesetzt werden.
Zeile 70 XMOV wird 512 mal ausgeführt mit Bild-Horizontalrücklauf.

Das Sprite scheint sich 'durch' die Daten zu bewegen, nicht hinter oder vor den Daten. Die Bewegung ist sehr glatt und angenehm für das Auge. Nun lassen Sie einmal das gleiche Programm laufen, diesmal jedoch verzögert, und sehen wir, das geschah, als wir uns 'durch' die Daten bewegten.

```
5 ' BEISPIEL VON EXKLUSIV-OR (LANGSAM)
10  Q%=0
20  !SCLS
30  !SPN,1:FOR I=1 TO 100
40  !COL,RND%77:!ROW,RND%184:!PTIF:NEXT I
50  LOCATE 7,1:PRINT "  ":LOCATE 8,1:PRINT "  "
60  !COL,0:!ROW,50:!SPN,8:!PTBL
70  !HGT,1:!LEN,1
80   FOR I=1 TO 260
90    !XMOV,1,1
100     FOR W=1 TO 50:NEXT W
110   !COLQ,@Q%:IF Q%=81 THEN !COL,-8:!ROW,50
120  NEXT I
```

Zeile 20 Löscht [illegible]z für das Sprite.
Zeile 30 und 40 Bringt Sprite 1 durch PUT auf 100 regellose Positionen.
Zeile 50 Löscht [illegible]z für das Sprite.
Zeile 60 und 70 Richt[illegible]e Ausgangspositionen und Richtungen für Sprite 8 ein
Zeile 80 Beginnt die Hauptschleife.
Zeile 90 Führt 1 XMOV mit Synchronisation durch Bild-Horizontalrücklauf aus.
Zeile 100 Richtet eine 'Hilfsschleife' ein, die alles verlangsamt.
Zeile 110 Prüft, ob Sprite 8 den Bildschirmrand erreicht hat. Falls ja, wird er auf die andere Bildschirmseite gebracht.
Zeile 120 Schleife zurück zu 80.

Ziemlich verwirrend, nicht wahr? Erstaunlicherweise sehen wir dies ganz anders, solange es schnell genug geschieht. Diese Bewegungsmethode hat übrigens mehrere Nachteile, die wir mit einigen Beispielen verdeutlichen wollen. Zuerst lassen wir einmal das Beispiel erneut laufen und diesmal mit voller Geschwindigkeit anstatt es auf Bild-Horizontalrücklauf zu synchronisieren.

```
5 ' BEISPIEL VON EXKLUSIV-OR
6 ' OHNE RAHMENRUCKSPRUNG
7 ' SYNCHRONISIERUNG
10  Q%=0
20  !SCLS
30  !SPN,1:FOR I=1 TO 100:!COL,RND%77
40  !ROW,RND%184:!PTIF:NEXT I
50  LOCATE 7,1:PRINT "  ":LOCATE 8,1:PRINT "  "
60  !COL,0:!ROW,50:!SPN,8:!PTBL
70  !HGT,1:!LEN,1
80  FOR I=1 TO 260
90    !XMOV
100   !COLQ,@Q%:IF Q%=81 THEN !COL,-8:!ROW,50
110 NEXT I
```

Es flimmert sehr stark – was ist also falsch? Das Problem bei dieser und den nächsten beiden Methoden, die wir Ihnen vorstellen, besteht daraus, daß wir uns zwischen zwei Übeln entscheiden müssen. Einerseits müssen wir darauf achten, daß der 'Punkt' (der das Bild auf dem Monitor oder auf dem Fernsehgerät entstehen läßt) verschwindet, so daß wir uns bewegen können, ohne anzustoßen, und andererseits müssen wir alles schnell genug tun, damit die 'Bewegung glatt' erscheint. Beim letzten Beispiel haben wir nicht auf den 'Punkt' gewartet. Die Bewegung selbst wird in zwei Phasen ausgeführt, die erste Phase nach XOR aus dem letzten Bild und die zweite nach XOR in das neue Bild. Wenn der Punkt zwischen diesen beiden Operationen ankommt (und es besteht eine gute Chance, daß dies geschieht, wenn wir uns schnell genug bewegen), 'verschwindet' das Sprite vor unseren Augen. Wir können diesen Effekt deutlicher zeigen, wenn wir ein großes Sprite mit Synchronisierung durch Bild-Horizontalrücklauf bewegen. Geben Sie dieses Beispiel ein und lassen Sie es laufen.

```
5 ' BEISPIEL ZUR BEWEGUNG EINES GROSSEN SPRITES
10 !SCLS
20 !SPN,10:!COL,40:!ROW,200:!SP1,10:!SP2,10:!SP3,10:!SP4,10
30 !PTXR
40 !HGT,-1:!LEN,0
50 !XMOV,512,1
```

Diesmal können Sie sehen, daß wir über einen Teil des Bildschirms eine kontinuierliche, glatte Bewegung erzielen, auf einem anderen einen Teil des Sprite verlieren und auf einem anderen wiederum das ganze Sprite. Worauf ist das zurückzuführen? Wir können es nicht verlangsamen, um [illegible] das in Zeitlupe zu zeigen, denn diese Wirkung tritt bei Zeitlupe nicht auf. Was geschieht ist folgendes: weil wir auf Bild-Horizontalrücklauf synchronisiert haben, wird der Entfernungs- und Ersatzzyklus für das Sprite auf den 'Punkt' synchronisiert. Das Sprite wird entfernt, wenn der 'Punkt' dran kommt und verschwindet einfach. Diese Beschränkung hat dazu geführt, daß einige Spieldesigner folgendes tun

a) [illegible] der Bewegung von großen Sprites auf sichere Bereiche des Bildschirms.
b) Gewährleistung, daß Sprites, die über den ganzen Bildschirm wandern, relativ klein sind.
c) Pause zwischen den Bewegungen, um sicherzustellen, daß der 'Punkt' auftritt und Bewegung, um die Sprunghaftigkeit zu verbergen.
d) Verwendung eines komplizierten (und normalerweise Spezial) Systems

Hiervon ist Möglichkeit c) bei weitem die nützlichste für uns, da die meisten Anwendungen auf jeden Fall Bewegung benötigen. Folgende Beispiele zeigen, wie sehr annehmbare Bewegungen XOR (exclusive – OR) entstehen können.

```
5 ' BEISPIEL EINER AKZEPTABLEN BEWEGUNG
6 ' MITTELS EXKLUSIV-OR
10 !SCLS
20 !SPN,12:!COL,72:!ROW,182:!SP1,12:!SP2,12:!SP3,12:!SP4,12
30 !PTXR
40 !HGT,0:!LEN,-1
50 FOR I=1 TO 4
60  !XMOV,80,1
70  !COL,72:!PTXR
80  !LEN,-I
90 NEXT I
```

Zeile 10 Löscht den Bildschirm.
Zeile 20 Gibt Sprite 12 als das Sprite an, das wir mit XMOV bewegen und bringt ihn in Ausgangsposition. Es ist zu bemerken, daß SP1, SP2, SP3 und SP4 den gleichen Wert haben, so daß keine Bewegung stattfindet.
Zeile 30 Bringt das Sprite mit XOR auf den Bildschirm.
Zeile 40 Setzt die Richtung des Sprite fest, das sich langsam nach links bewegt
Zeile 50 Hauptschleife.
Zeile 60 Bringt das Sprite durch XMOV über den Bildschirm
Zeile 70 Stellt die Ausgangsspalte zurück und wendet erneut 'PUT' auf sie an.
Zeile 80 Die horizontale Geschwindigkeit des Sprite wird der Schleifenvariablen I zugewiesen.
Zeile 90 Schleife zurück zu 50.

Bewegung 'vor'

Obgleich die Bewegung exclusive-OR, die im letzten Kapitel beschrieben wurde, eine annehmbare Bewegung erzeugt, werden diejenigen unter Ihnen, die auf Hardwaresprites beruhende Grafiken gesehen haben, den Vorteil einer echten zerstörungsfreien Beweung zu schätzen wissen. Bedauerlicherweise hat der Schneider keine Hardwaresprites, jedoch können wir einige ihrer Merkmale durch Bewegungen 'vor' und 'hinter' simulieren.

Über die Funktionen dieses Verfahrens müssen wir uns nicht länger aufhalten, es genügt zu sagen, daß hierfür eine Technik verwendet wird, die mit der bei der Filmherstellung benutzten verwandt ist, wenn z.B. 'fliegende Untertassen' über London erscheinen sollen. Wenn diese Methode funktionieren soll, müssen wir jedoch speziell präparierte (abgedeckte) Sprites benutzen. Die Sprites sind mit der doppelten Breite des Bildes zu schaffen, das wir darstellen wollen. Die Teile des Bildes, die nicht 'transparent' sind, mussen in INK 0 gehalten werden. Das Bild muß ganz in der linken Spritehälfte enthalten sein. Wenn Sie ein Sprite in dieser Form geschaffen haben, brauchen Sie nur noch den Befehl MASK verwenden, um ihn in einen (abgedeckten) MASK-Sprite zu verwandeln. Wenn dieses Sprite mit den normalen PUTs und GETs dargestellt wird, erscheinen die Daten unorganisiert. Machen Sie sich daruber keine Sorge, denn es gibt einen Befehlssatz, der ausschließlich MASK-Sprites behandelt. Ein MASK-Sprite wird übrigens als ein Sprite seiner halben wirklichen Größe von den Befehlen angesehen, die es behandelt. Jeder Versuch, kein Sprite mit einer 'ungeraden' Breite abzudecken, führt zu einer Fehleranzeige "CAN'T MASK". Sie dürfen daher nur Sprites mit 'gerader' Breite benutzen. Das Bild kann naturlich eine ungerade Breite aufweisen. Abdecken (maskieren) eines bereits abgedeckten Sprites zerstört die Daten.

Diese Bewegungsart leidet an den gleichen Zeitbeschränkungen wie exclusive-OR, jedoch bestehen keine weiteren Beschränkungen für den Einsatz, die nicht auch bei exclusive- OR auftreten. Jedesmal, wenn ein abgedecktes Sprite auf den Bildschirm gebracht wird, werden die Daten, die löschend überschrieben werden, gleichzeitig zu späteren Ersatz in das Sprite gehoben. Das bedeutet, daß sich der Hintergrund bei der Bewegung nicht weiter entwickeln darf, und diese Bewegungsart wird auf stationäre Bildschirmdaten beschränkt. Das 'vor' bewegte Sprite darf daher nicht von einem beweglichen Gegenstand loschend überschrieben werden. Diese Regel gilt übrigens ebenfalls für die Bewegung 'hinter'. Nachstehend ein Beispiel der Bewegung 'vor'. Lassen Sie das Programm ganz ablaufen, denn dieses Sprite kann durch Drucken von Pause verstummelt werden. Wenn Sie das Programm erneut laufen lassen wollen, streichen Sie Zeile 40, da sonst das bereits abgedeckte Sprite abgedeckt wird.

```
5  ' BEISPIEL VON IN-FRONT-BEWEGUNG
10 !SCLS
20 !SPN,1:FOR I=1 TO 200:!COL,RND*77
30 !ROW,RND*184:!PTIF:NEXT I
40 !SPN,13:!MASK
50 !SP1,13:!SP2,13 :!SP3,13:!SP4,13
60 !COL,72:!ROW,100:!HGT,0 :!LEN,-1
70 !FSWP
80  FOR I=1 TO 3
90   !FMOV,236,1
100 NEXT I
```

Zeile 10, 20 und 30 Löschen den Bildschirm und bringen Sprite 1 200 mal auf regellose Positionen.
Zeile 40 Maskiert Sprite 13.
Zeile 50 Setzt alle vier Bilder auf die gleiche Spritenummer, d.h. keine Bewegung.
Zeile 60 Richtet Ausgangsposition und Richtung ein.
Zeile 70 'Tauscht' Sprite 13 auf dem Bildschirm 'nach vorn'.
Zeile 80 3 malige Schleife.
Zeile 90 Bringt das Sprite durch 'FMOV' über die volle Bildschirmhöhe mit Synchronisation durch Bild-Horizontalrücklauf.
Zeile 100 Schleife zurück.

Bewegung 'hinter'

Diese beruht auf der gleichen Idee wie die Bewegung 'vor', mit dem einzigen Unterschied, daß das Sprite hinter andere Daten bewegt wird, über die es sich bewegt. Das Sprite ist wiederum in der gleichen Weise, wie im letzten Kapitel beschrieben, zu überdecken, und hier treten die gleichen Bewegungsbeschränkungen auf. Sie müssen sicher sein, daß das Sprite nicht durch andere bewegliche Sprites löschend überschrieben wird, während er sich selbst bewegt.

Die Überdeckungsstruktur wird übrigens durch die Bewegung 'hinter' geändert, und der Original-MASK ist zu rekonstruieren, bevor sich ein bestimmtes Sprite, das sich 'hinter' bewegt hat, dann 'vor' bewegt werden kann. Hierzu steht ein Befehl zur Verfügung – RMSK. Beim folgenden Beispiel wird ein Sprite über den Bildschirm bewegt, sowohl hinter und vor die Bildschirmdaten, und es ist besondere Aufmerksamkeit auf die erneute Abdeckung (unter Verwendung von RMSK) zwischen der Bewegung hinter und der Bewegung vor die Bildschirmdaten zu richten.

```
5 ' BEISPIEL VON NEUMASKIERUNG
10  !SCLS
20  !SPN,2:FOR I=1 TO 100:!COL,RND*77
30  !ROW,RND*184:!PTIF:NEXT I
40  !SPN,13
50  !SP1,13:!SP2,13:!SP3,13:!SP4,13
60  !ROW,100:!HGT,0:!LEN,-1
70  FOR N=1 TO 5
80    !COL,72:!FSWP
90    !FMOV,81,1
100   !COL,72:!BPUT
110   !BMOV,81,1
120   !RMSK
130 NEXT N
```

Zeile 20 und 30 Bringt Sprite 2 regellos auf dem Bildschirm.
Zeile 40 Setzt SPN für die erneute Abdeckung ein.
Zeile 50 Setzt SP1, SP2, SP3 und SP4 auf 13. [illegible]e Bewegung.
Zeile 60 Setzt den Wert für ROW und die Bewegungsrichtung in HGT und LEN ein.
Zeile 70 Ist die Schleife.
Zeile 80 Setzt den Ausgangswert COL ein und bringt den Sprite unter Verwendung von |FSW auf den Bildschirm.
Zeile 90 Bewegt das Sprite unter Verwendung von |FMOV vor die Daten.
Zeile 100 Setzt den Ausgangswert COL ein und bringt das Sprite unter Verwendung von |BMO auf den Bildschirm.
Zeile 110 Bewegt das Sprite unter Verwendung von |BMOV hinter die Daten.
Zeile 120 Nimmt die erneute Abdeckung des Sprite vor.
Zeile 130 Schleife zu Zeile 70.

Nachdem wir uns jetzt die verfügbaren Verfahren für die Bewegung mit Laser BASIC angesehe haben, kommen wir zu den verschiedenen Bewegungsarten im Einzelnen und stellen fest, wie sie der Praxis eingesetzt werden. Denken Sie daran, was wir über die Vorteile und Nachteile de verschiedenen Methoden gesagt haben und verwenden Sie immer das einfachste System. Wenn S sich kommerzielle Spiele ansehen, ist es nicht überraschend festzustellen, daß sie häufig alle dies Faktoren berücksichtigen. Ein typisches Bühnenspiel zum Beispiel benutzt fast für die gesam Bewegung das löschende Überschreiben von Blocks, und der Hintergrund ist hierauf sorgfält abgestimmt. Ein Sprite, das eine Leiter hochklettert, braucht kein XOR, wenn die Leiter als Te seines Bildes ausgelegt ist!

Die Linearen MOVE-Befehle

Diese Befehle unterstützen alle vier Bewegungsmethoden, jedoch beginnen wir, indem wir un WMOV ansehen, das auf dem löschenden Überschreiben von Blocks beruht. Der linear Bewegungsbefehl beruht auf 8 Grafikvariablen und 2 oder 3 Parametern nach Wahl.

WMOV

COL	Die Spalte, von der die Bewegung ausgeht, in Bytes gemessen.
ROW	Die Zeile, von der die Bewegung ausgeht, in Bildpunkten gemessen.
LEN	Das Inkrement der horizontalen Bewegung, gemessen in Bytes.
HGT	Das Inkrement der senkrechten Bewegung, gemessen in Bildpunkten.
SP1	Die Zahl des zu bewegenden Sprite, in diesem Fall der löschend zu überschreibend Block.
SP2	Die Zahl des Sprite, das den bewegten Sprite ersetzt.
SP3	Die Zahl des Sprite, das SP2 ersetzt.
SP4	Die Zahl des Sprite, das SP3 ersetzt.

Hierzu sind einige Punkte zu notieren:

1. Positive Werte für LEN verursachen die Bewegung nach rechts, negative Werte die Bewegun nach links.

2. Positive Werte für HGT verursachen die Bewegung auf dem Bildschirm nach unten, negativ Werte die Bewegung zur Oberkante des Bildschirms.

3. SP1, SP2, SP3 und SP4 enthalten die Spritezahlen der vier Bewegungsbilder. Sie können je gewünschte Zahl enthalten, und die Zahlen müssen nicht aufeinander folgen. Sie könnten al die gleiche Zahl enthalten, wenn das Sprite sich nicht bewegen mußte.

4. Wenn WMOV oder jeder andere MOVE Befehl ausgeführt wird, wird angenommen, daß SP früher durch PUT auf den Bildschirm gebracht wurde. Im Falle von WMOV muß SP1 nicht a dem Bildschirm sein, aber bei den anderen drei Befehlen in der Gruppe (XMOV, BMOV, FMO ist dies nötig. Das erste Sprite, das mit WMOV auf dem Bildschirm erscheint, ist SP2.

Da WMOV das [illegible] fassung nicht bedeutungsvoll, da eine Kollision normalerweise beim vorherigen Bild festgestellt wurde, das lösch[illegible] überschrieben wird.

Denken Sie daran, daß ein Sprite, wenn es durch 'WMOV' bewegt wird, einen Rand rundum benötigt, der mindestens so groß ist wie die Bewegungsinkremente, da sonst das Sprite eine Spur hinterläßt.

Das erste Beispiel zeigt die Bewegung eines Sprite durch 'WMOV' ohne Rand – und Sie sehen die Spur, die es hinterläßt.

```
5 ' BEISPIEL VON BLOCKUBERSCHREIBUNGSBEWEGUNG
10 !SCLS:!COL,0:!ROW,100
20 !SPN,14:!SP1,14:!SP2,14:!SP3,14:!SP4,14:!LEN,1:!HGT,0
30 !WMOV,256,1
```

Bei diesem zweiten Beispiel schaffen wir ein größeres Sprite und setzen dieses kleinere (den wir mit WMOV bewegen möchten) hinein – die Spur verschwindet.

```
5 ' BEISPIEL VON BLOCKUBERSCHREIBUNGSBEWEGUNG II
10 !SCLS
20 !COL,2:!ROW,8:!SPN,14:!PTBL:!SPN,15:!DSPR:!HGT,18:!LEN,
   12:!COL,0:!ROW,6:!SPN,
15:!CSPR:!GTBL
30 !SPN,15:!SP1,15:!SP2,15:!SP3,15:!SP4,15
40 !LEN,1:!HGT,0
50 !WMOV,512,1
```

Zeile 10 Löscht den Bildschirm.
Zeile 20 Bringt Sprite 14 auf den Bildschirm, schafft dann Sprite 15 (etwas größer als 14) und bringt dann das Bildschirmbild von Sprite 14 in Sprite 15.
Zeile 30 Gibt die Bewegungsvariablen an:
Zeile 40 Setzt die Richtung ein, in der sich das Sprite bewegt.
Zeile 50 Bewegt Sprite 15 durch WMOV 512 mal mit Synchronisierung durch Bild-Horizontalrücklauf.

Der Befehl WMOV kann, wie alle anderen MOVE-Befehle, in einer Maschinencode-Schleife [illegible]usgeführt werden, mit oder ohne Kollisionserfassung und Synchronisation durch Bild-[illegible]orizontalrücklauf. Um WMOV in einer Maschinencode-Schleife auszuführen, folgen wir einfach dem Befehl mit zwei Parametern. Der erste gilt dafür, wie häufig wir den Befehl ausführen wollen, und der zweite sagt dem System, ob mit Bild-Horizontalrücklauf zu synchronisieren ist oder nicht. Die [illegible]schenden Überschreibungsoperationen der Blocks müssen übrigens nicht unbedingt mit Bild-[illegible]orizontalrücklauf synchronisiert werden, denn sie leiden nicht an den gleichen Flimmerproblemen [illegible]ie andere Methoden. An diesem nächsten Beispiel zeigen wir die Bewegung mit und ohne Synchronisation durch Bild-Horizontalrücklauf und zeigen ferner die Bewegung.

```
' BEISPIEL VON BLOCKUBERSCHREIBUNGS-
' BEWEGUNG MIT TRICKFUNKTION
:SCLS
!SPN,17:!LEN,0 :!HGT,1:!SP1,17:!SP2,18:!SP3,19:!SP4,20:!COL,40
!ROW,0 :FOR L=1 TO 200
 !WMOV,1,1
  FOR I=1 TO 60:NEXT I
NEXT L
!ROW,0 :FOR L=1 TO 200
 !WMOV
  FOR I=1 TO 60:NEXT I
NEXT L
```

Zeile 10 Löscht den Bildschirm und richtet die Ausgangsposition ein.
Zeile 20 Richtet eine Spritezahl ein, die Richtung und [illegible]enfolge der Bewegung, unter Verwendung von SP1, SP2, SP3 und SP4.
Zeile 30 Richtet die erste Hauptschleife ein.
Zeile 40 Bewegung durch WMOV mit Bild-Horizontalrücklauf.
Zeile 50 Kleine Warteschleife, die das Sprite daran hindert, daß es sich zu schnell bewegt, um sichtbar zu sein.
Zeile 60 Schleife zurück zu 30.
Zeile 70 Richtet die zweite Hauptschleife ein:
Zeile 80 Bewegung mit WMOV ohne Bild-Horizontalrücklauf.
Zeile 90 Eine weitere kleine Pausenschleife.
Zeile 100 Schleife zurück zu 70.

XMOV

Bei XMOV werden die gleichen 8 Grafikvariablen wie bei WMOV benutzt (ebenfalls bei FMOV und BMOV), und diesen können, wie bei allen MOVE Befehlen bis zu drei Parameter folgen. Auch hier sind einige Punkte zu erwähnen:

1. XMOV setzt voraus, daß Sprite SP1 bereits mit XOR auf dem Bildschirm in den jeweiligen COL und ROW-Postionen behandelt wurde. Wenn dies nicht geschehen ist, bleibt eine Kopie von SP in dieser Position.

2. SP1 ist übrigens nicht mit XOR zu behandeln, wenn die Bewegung von einer Postion außerhalb des Bildschirm beginnen soll.

3. Die Kollisons-Erfassung kann durch XMOV vorteilhaft durchgeführt werden.

4. Wenn XMOV wiederholt mit Synchronisation durch Bild-Horizontalrücklauf ausgeführt wird, kann der obere Bildschirmteil nicht die flimmerfreie Bewegung großer Sprites zulassen. In der Praxis ist das 'no-go' Band umso tiefer, je größer das Sprite ist.

Bei diesem ersten Beispiel bewegen wir ein Sprite durch XMOV von der Bildschirmmitte nach unten links (und lassen es umlaufen), ohne zuerst SP1 durch PUT zu behandeln. Bitte beachten Sie, daß eine Kopie von SP1 verbleibt.

```
5 ' BEISPIEL VON XMOV
10 !SCLS
20 !COL,40:!ROW,100
30 !SP1,2 :!SP2,2  :!SP3,2:!SP4,2
40 !HGT,-2:!LEN,1
50 !XMOV,218,1
```

Zeile 10 Löscht den Bildschirm.
Zeile 20 Richtet die Ausgangspositionen ein.
Zeile 30 Setzt die Spritezahlen ein (keine Bewegung).
Zeile 40 Setzt die Richtung ein.
Zeile 50 Bewegt Sprite 2 durch XMOV 218 mal mit Synchronisierung durch Bild-Horizontalrucklauf.

Wir wiederholen jetzt das Beispiel, initjalisieren aber jetzt XMOV durch Einsatz von PTXR, um Sprite SP1 (der übrigens in SPN enthalten ist) durch PUT vor Bewegung in die Mitte zu bringen.

```
5 ' BEISPIEL VON XMOV II
10 !SCLS
20 !COL,40:!ROW,100
30 !SPN,2 :!PTXR
40 !SP1,2 :!SP2,2:!SP3,2:!SP4,2
50 !HGT,-2:!LEN,1
60 !XMOV,218,1
```

Bevor wir zu unserem nächsten Beispiel übergehen, erwähnen wir am besten kurt den Bildschirm-'Umlauf', d[illegible]n allen linearen Bewegungsbefehlen benutzt wird. Der Bildschirm ist 80 Bytes breit und 200 Bildpunkte hoch, jedoch kann man es so sehen, daß sich die Sprites in einem Raum bewegen, der 256 Bytes (3 Bytes und ein Bit Bildschirme) breit und 256 Bildpunkte hoch ist.

Wenn der Weg eines Sprite den Rand des 'virtuellen Bildschrims' überquert, läuft es um, bis es schließlich auf dem Bildschirm selbst ankommt und erneut sichtbar ist. Dieser 'Umlauf' findet in jeder Richtung statt.

Wenn wir mit der Bewegung eines Sprite durch 'XMOV' aus einem virtuellen Bereich beginnen, müssen wir ihn nicht mit PXTR 'starten'. Das nächste Beispiel verdeutlicht diesen Punkt.

```
5 ' BEISPIEL VON XMOV III
10 !SCLS
20 !COL,-8:!ROW,100
30 !SP1,2 :!SP2,2  :!SP3,2:!SP4,2
40 !HGT,0 :!LEN,1
50 !XMOV,218,1
```

Kommen wir jetzt einmal zur Kollisionserfassung. In allen bisher besprochenen Beispielen haben wir entweder die Bewegung ohne Parameter vorgenommen oder mit 2 Parametern (Maschinencode-Schleifen). Sehen wir uns jetzt einmal ein Beispiel an, das die Kollisionserfassung verdeutlicht.

```
5 ' BEISPIEL EINER KOLLISIONSERFASSUNG
6 ' OHNE MASCHINENCODESCHLEIFE
10 X%=0
20 !SCLS
30 !SPN,1 :FOR I=1 TO 40:!COL,RND*77:!ROW,RND*184:!PTIF:NEXT I
40 !COL,-8:!ROW,100
50 !SP1,2 :!SP2,2  :!SP3,2:!SP4,2
60 !HGT,0 :!LEN,1
70 FOR I=1 TO 512
80   !XMOV,@X%
90   IF X%>3 THEN SOUND 1,40,5:X%=0
100 NEXT I
```

Zeile 10 Erklärt X%, das bei der Kollisionserfassung zu verwenden ist.
Zeile 20 Löscht den Bildschirm.
Zeile 30 Bringt gewisse Daten auf den Bildschirm.
Zeile 40 Bringt die Ausgangspositionen des Sprite direkt links vor den Bildschirm, d. h. -8.
Zeile 50 Richtet die Spritezahlen ein.
Zeile 60 Setzt die Richtung ein.
Zeile 70 Beginn der Hauptschleife.
Zeile 80 XMOV mit X% als Kollisionssuchzähler.
Zeile 90 Prüft, ob X% anders ist und falls ja, ertönt ein kurzer Piepton, worauf X% auf 0 rückgestellt wird.
Zeile 100 Schleife zurück zu 70.

Die Kollisionserfassung erfolgt durch überfahren der Adresse einer BASIC-Variablen (daher das so wichtige '@' vor der Variablenbezeichnung), so daß die Bewegungsroutine die Variable erhöhen kann, wenn die Suche erfolgt. Durch XMOV,@X% würde die Bewegung einmal erfolgen, und wenn eine Kollision auftritt, würde sie um X% erhöht. Durch XMOV,@X%,500,1 würde die Bewegung 500 mal erfolgen (mit Synchronisation durch Bild-Horizontalrücklauf), und die Inkrementierung würde jedesmal, wenn eine Kollision festgestellt wird, X% betragen (in diesem Fall maximal 500 mal). Es ist sehr wichtig nicht zu vergessen, das '@' vor X% oder dem System absolut eine Kollision sein kann. Die für die Kollisionserfassung benutzte BASIC-Variable muß eine ganzzahlige Variable sein, d.h. ihr muß '%' folgen oder sie muß als ganzzählig in einer DEFINT-Anweisung erklärt worden sein. Wenn die benutzte Variable noch nicht erklärt worden ist, erfolgt eine Fehlermeldung 'improper argument'. Um diese zu löschen, nur eine Anweisung eingeben wie X%=0, bevor der Befehl ausgeführt wird.

FMOV entspricht übrigens XMOV in jeder Hinsicht, außer daß die Bewegung vor die Bildschirmdaten erfolgt und nicht durch exclusive-OR. Zur Verwendung die Operation 'vor' sind jedoch einige Punkte zu klären.

1. Alle durch diese Bewegungsart benutzten Sprites (und der FSWP, der die Bewegung initialisiert) sind zu überdecken, bevor die Bewegung beginnen kann. Wenn eines dieser Sprites in einer Bewegung 'hinter' benutzt wurde, ist es erneut mit dem Befehl RMSK zu überdecken. Die Ausführung des Befehls RMSK kann keine Schäden verursachen, in Zweifelsfällen nehmen Sie also eine erneute Maskierung vor!

2. FMOV setzt voraus, daß Sprite SP1 bereits unter Verwendung des Befehls FSWP auf den Bildschirm gebracht wurde (außer wenn die Bewegung von einer Position außerhalb des Bildschirms beginnt). Wenn SP1 nicht unter Verwendung von FSWP auf dem Bildschirm gebracht wurde, wird SP1 unwiderruflich verstümmelt. Aus diesem Grund ist es eine gute Idee, eine Kopie aller überdeckten Sprites in anderen Sprites zu halten, so daß Sie verstümmelte Sprites unter Verwendung der Befehle 'PM' oder 'GM' rekonstruieren können.

3. Nachdem Sie die Bewegung eines Sprite mit FMOV beendet haben, ist es durch Ausführen eines weiteren FSWP vom Bildschirm zu entfernen oder 'außerhalb des Bildschirms' zu bewegen. Wenn dies nicht geschieht, kann dies auch zur Verstümmelung eines Sprite führen.

Beim nächsten Beispiel beginnen wir mit vier nicht abgedeckten Sprites, die die vier Bilder einer bewegten Folge bilden sollen, die dann, bereit für die Bewegung, maskiert werden. Wenn wir zufällig eines dieser abgedeckten Sprites verstümmeln, können wir das Verfahren wiederholen. Das folgende kurze Programm zeigt den genauen Gebrauch des Befehls FMOV.

```
5 ' BEISPIEL VON FMOV
10 !SCLS
20 FOR I=21 TO 24 :!SPN,I:!MASK:NEXT I
30 FOR I=1  TO 50 :!SPN,1:!COL,RND*80:!ROW,RND*200:!PTBL:NEXT I
40 !SPN,21:!SP1,21:!SP2,22:!SP3,23:!SP4,24
50 !COL,40:!ROW,0 :!FSWP
60 !HGT,1 :!LEN,0
70 FOR I=1 TO 480
80 !FMOV,1,1
90 FOR W=1 TO 50:NEXT W
100 NEXT I
```

Zeile 10 Löscht den Bildschirm.
Zeile 20 Überdeckt Sprites 21 bis 24.
Zeile 30 Bringt die Daten auf dem Bildschirm, um zu zeigen, daß sie sich 'vor' bewegen.
Zeile 40 Setzt die Spritezahlen für die Bewegung ein.
Zeile 50 Setzt die Ausgangspositionen und die FSWP ein.
Zeile 60 Legt Richtung und Anzahl der Bildpunkte in dieser Richtung fest, über die sie sich bewegen.
Zeile 70 Hauptschleife
Zeile 80 FMOV einmal mit Synchronisation durch Bild-Horizontalrücklauf.
Zeile 90 Einrichten der Verzögerungsschleife, so daß Sie die Bewegung sehen können.
Zeile 100 Zurück zu 70.

BMOV

Der Befehl BMOV wird in genau der gleichen Weise verwendet wie der Befehl FMOV; nur mit einigen beachtenswerten änderungen.

1. Während FMOV für den ersten Rahmen, der auf den Bildschirm gebracht wird, einen FSWP Befehl benötigt, und anschließend am Ende der Bewegungsfolge, zum Entfernen des letzten Rahmens, einen anderen FSWP braucht, müssen für BMOV zwei ausgeprägte Befehle verwendet werden. Der erste Rahmen wird mit BPUT auf den Bildschirm gebracht und der letzte Rahmen wird mit BGET entfernt. BPUT ist nicht erforderlich, wenn sich der erste Rahmen 'off-screen' (außerhalb des Bildschirms) befindet, und BGET wird nicht benötigt, wenn der letzte Rahmen off-screen' ist.

ummaskiert werden, bevor sie von FSWP und FMOV benutzt werden können.

Joystick-/Tastatur-MO Befehle

Diese Befehle unterstützen alle vier Bewegungsarten, verwenden 9 Grafikvariablen und werden mit bis zu 3 optionalen Parametern ausgeführt. Sie sind den linearen Bewegungsbefehlen in fast jeder Hinsicht ähnlich, deren Bewegung wird jedoch durch eine vom Benutzer wählbaren Joystick- bzw. Tastatur gesteuert.

Diese Bildschirme verwenden dieselben 8 Grafikvariablen wie deren Vorgänger, haben jedoch eine zusätzliche Variable, um die erforderliche Joystick- oder Tastaturreihe zu wählen. Die zusätzlich verwendete Variable ist KEY. Zuordnung von Werten und Abfrage erfolgen wie gewöhnlich. Joystick 0 wird gewählt, indem KEY auf einen Wert im Bereich von 72 bis 79 eingestellt wird, und Joystick 1 wird gewählt, indem KEY auf einen Wert im Bereich von 48 bis 55 eingestellt wird. Wenn Sie keine Joysticks angeschlossen haben (oder diese nicht verwenden), können Sie Ihr Sprite auch mit Tasten herumbewegen.

Tabelle 2 zeigt, welche Tastenkombination mit den einzelnen KEY-Werten korrespondieren. Die Tasten in den runden Klammern korrespondieren mit den Tasten des numerischen Tastenfelds (falls angeschlossen). UP, DOWN, LEFT und RIGHT korrespondieren mit den Cursortasten.

VALUE IN KEY	UP KEY	DOWN KEY	LEFT KEY	RIGHT KEY
0	UP	RIGHT	DOWN	(9)
8	LEFT	COPY	(7)	(8)
16	CLR	[	ENTER	]
24	↑	-	@	P
32	0	9	O	I
40	8	7	U	Y
48	6	5	R	T
56	4	3	E	W
64	1	2	ESC	Q

TABLE 2

Der Trickbewegungskreislauf wird nur dann fortgesetzt, wenn eine Taste gedrückt wird (oder der Joystick aktiviert wird). Der Kreislauf wird immer um einen Rahmen fortgesetzt, unabhängig von der HGT- und LEN-Erhöhung, der Anzahl der gedrückten Tasten oder der Bewegungsrichtung. Nachfolgend sind einige Beispiele, die illustrieren, was mit diesen Befehlen erzielt werden kann. Das erste Beispiel demonstriert die Bewegung mit Joystick und es kann ausgelassen werden, falls Sie keinen Joystick besitzen.

```
5 ' BEISPIEL ZUR BEWEGUNG EINES SPRITES
6 ' MITTELS JOYSTICK 0
10 !SCLS
20 !COL,40:!ROW,100
30 !SPN,17:!SP1,17 :!SP2,18:!SP3,19:!SP4,20
40 !PTXR
50 !HGT,1 :!LEN,1
60 !KEY,72
70 !WMVJ,1,1
80   FOR I=1 TO 50:NEXT I
90 GOTO 60
```

Zeile 10 Löscht den Bildschirm.
Zeile 20 Stellt die COL- und ROW-Position für Sprite 17 ein.
Zeile 30 Stellt SPN und SP1 bis SP4 ein (die 4 Sprites im Trickbewegungsablauf).
Zeile 40 Sprite 17 wird mit Exklusiv-ODER auf den Bildschirm gebracht.
Zeile 50 Stellt HGT und LEN auf 1 ein.

Zeile 70 Bewegung und Trickbelebung der Sprites m[illegible]WMVJ.
Zeile 80 Ist eine Verzögerungsschleife.
Zeile 90 Stellt mit Zeile 60 eine Schleife her.

Das nächste Beispiel, das dem obigen ähnlich ist, zeigt, wie Bewegung und Trickeffekte, durch d[illegible] Tastatur gesteuert werden können (s. Tabelle 2).

KEY ist auf 40 eingestellt. Dies bedeutet, daß mit Taste '8' die Trickbewegung der Sprites nach obe[illegible] erfolgt, mit Taste '7' nach unten, mit Taste 'U' nach links und mit Taste 'Y' nach rechts.

```
5 ' BEISPIEL ZUR BEWEGUNG EINES SPRITES
6 ' MITTELS TASTATUR
10 !SCLS
20 !COL,40:!ROW,100
30 !SPN,17:!SP1,17 :!SP2,18:!SP3,19:!SP4,20
40 !PTXR
50 !HGT,1 :!LEN,1
60 !KEY,40
70 !WMVJ,1,1
80  FOR I=1 TO 50:NEXT I
90 GOTO 60
```

Sie werden vielleicht feststellen, daß die Bewegung nach rechts mit der 'Y'-Taste und die Bewegun[illegible] nach links mit der 'U'-Taste etwas kompliziert ist, da sich die 'Y'-Taste links von der 'U'-Tast[illegible] befindet. Sie können allerdings die in Tabelle 2 vorgegeben Richtungen umkehren, indem Sie in HG[illegible] und LEN negative Vorzeichen setzen. Das untere Beispiel ist gleich wie das obige, HGT und LE[illegible] sind jetzt jedoch auf -1 eingestellt, so daß die 'Y'-Taste die Linksbewegung ausführt, die 'U'-Taste d[illegible] Rechtsbewegung, die Taste '7' die Aufwärtsbewegung und die Taste '8' die Abwärtsbewegung[illegible]

```
5 ' BEISPIEL ZUR BEWEGUNG EINES SPRITES
6 ' MITTELS TASTATUR II
10 !SCLS
20 !COL,40:!ROW,100
30 !SPN,17:!SP1,17 :!SP2,18:!SP3,19:!SP4,20
40 !PTXR
50 !HGT,-2 :!LEN,-1
60 !KEY,40
70 !WMVJ,1,1
80  FOR I=1 TO 50:NEXT I
90 GOTO 60
```

Die 'Sprung'-MOVE-Befehle

Es ist wahrscheinlich nicht sofort verständlich, warum diese besondere MOVE-Gruppe dem Lase[illegible] BASIC eingegliedert wurde. Wie Sie sehen werden, finden diese beim Entwerfen von Spielen und vo[illegible] allem bei Plattformspielen viele Anwendungsmöglichkeiten. Wie die beiden vorangegangene[illegible] Gruppen, unterstützen auch diese Befehle alle damit verbundenen Bewegungsarten und benutzen[illegible] Grafikvariablen. Sie verwenden dieselben 9 Variablen, die auch bei den Tastatur[illegible] Joystickbewegungen verwendet werden. Bei dieser Anwendung wird jedoch KEY dazu benutzt, di[illegible] Nummer eines Sondersprites zu halten, das die Sprungfenster-Parameter enthält.

Sprungfenster

Ein Sprungfenster wird durch die vier Variablen COL, ROW, HGT und LEN definiert. Das Fenste[illegible] wird eingestellt und in einem Sprite gespeichert, bevor die Sprungbefehle ausgeführt werden. De[illegible] Befehl für diese Operation ist BWST (bounce window SET). Die linke, obere Ecke des som[illegible] erzeugten Fensters wird durch COL und ROW definiert und seine Größe wird durch HGT und LE[illegible] bestimmt. Sprites werden 'innerhalb' dieses Fensters nicht springen und deren COL- un[illegible] ROW-Werte sind auf die Bereiche COL bis COL+LEN und ROW bis ROW+HGT beschränkt. HG[illegible] und LEN bestimmen daher eher den freien Bewegungsraum des Sprites, als daß die ein[illegible] physikalische Grenze darstellen.

Noch ein erwähnenswerter Punkt wäre, daß die Kollisionserfassung der beiden ersten Bewe[illegible]gsarten durch eine Sprung[illegible]ssungsoption ersetzt wird, die jedoch genauso aufgerufen wird, d.h. 1 oder 3 nachfolgende Parameter verwendet. Zum Schluß wäre noch zu erwähnen, daß die Rahmenfolge jedesmal umgekehrt wird, wenn ein Sprung vorkommt. Ein simultaner Sprung an zwei aufrechten Seiten wird zwei Umkehrungen zur Folge haben, wodurch die Rahmenfolge unverändert fortgesetzt wird.

Wenn die Bewegung außerhalb des Fensters beginnt, wird sich das Sprite solange geradlinig bewegen, bis es versucht durch einen Teil des Fensters zu gehen, wobei es 'gefangen' wird und innerhalb der Fenstergrenzen weiterspringt.

Anwendungsmöglichkeiten für Plattformspiele

Trickeffekte sind bei linearen Bewegungen im Rahmenablauf nicht möglich, es sei denn HGT bzw. LEN sind ungleich null. Das heißt, um Trickeffekte zu erzielen müssen sie sich bewegen. Mit den Sprungbefehlen kann ein Sprite um einen Bildpunkt nach oben und unten hüpfen. Wenn aufeinanderfolgende Rahmen um einen Bildpunkt versetzt sind, können, wie aus dem nachfolgenden Beispiel ersichtlich, stationäre Trickeffekte erzeugt werden.

```
5 ' BEISPIEL VON STATIONARER TRICKBELEBUNG
6 ' MITTELS WBNC
10 !SCLS
20 !SPN,18:!NPX,1:!SSVN :!SPN,20:!SSVN
30 !SPN,15:!DSPR :!HGT,1:!LEN,4 :!CSPR
40 !SPN,15:!HGT,2:!LEN,1:!COL,39:!ROW,100:!BWST
50 !KEY,15:!COL,39:!ROW,100:!HGT,1:!LEN,0:!SP1,19:!SP2,20:!SP3,17:!SP4,18
60 !WBNC,1,1
70 FOR I=1 TO 100:NEXT I
80 GOTO 60
```

Zeile 10 Löscht den Bildschirm.
Zeile 20 Rollt Sprites 18 und 20 um 1 Bildpunkt nach oben (wenn Sie dieses Programm wieder ablaufen lassen, werden diese Sprites noch einmal gerollt).
Zeile 30 Erzeugt ein springendes Datensprite.
Zeile 40 Stellt das Sprungsprite ein.
Zeile 50 Stellt die Position des Sprites und die Werte von SP1 bis SP4 ein.
Zeile 60 Führt einen Einzelsprung aus.
Zeile 70 Dies ist eine Verzögerungsschleife.
Zeile 80 Bildet mit Zeile 60 eine Schleife.

Wenn wir uns einen typischen Bildschirm eines beliebten Plattformspiels näher betrachten, werden wir feststellen, daß ein Großteil der Bewegungen durch vertikale oder horizontale Sprungbewegungen erzeugt werden kann. Später können Sie das Laser-BASIC-Demo laufen lassen. Ein Demobildschirm zeigt ein plattformartiges Spiel, bei dem 7 Sprungfenster eingestellt wurden. Sprites wie rollende Augäpfel, ein Klosett und ein Lift scheinen ziemlich gleichmäßig zu springen.

Wenn Sie dieses Beispiel erarbeitet haben, könnten Sie dieses Demo abbrechen und das Programm erneuern, indem Sie "NEW" eintippen. Laser BASIC und Sprites werden im Speicher verbleiben. Wenn Sie also das folgende Programm eintippen, werden alle 7 Sprites am Bildschirm herumspringen.

```
5 ' BEISPIEL VON SPRUNGFENSTERN
10 FOR I=102 TO 108
20  !SET,I-102
30  !SPN,I:!ASTV
40 NEXT I
50 A$="WBNCAWBNCBWBNCDWBNCEWBNCFWBNCG#"
60 !ISET,@A$
70 !IRUN,4
```

Zeile 20 Stellt in der Variable SET den Wert I-102 ein.
Zeile 30 Benutzt ASTV, um die auf I-102 gesetzte Variable mit den im Datensprite gespeicherten Werten einzustellen.
Zeile 40 Bildet mit 10 eine Schleife.
Zeile 50 Dies ist die Kette mit den 7 WBNC-Befehlen, die unter Interrupt laufen.
Zeile 60 Kompiliert die Kette A$ in die Hintergrundtabelle.
Zeile 70 Läßt das Hintergrundsprungprogramm bei jedem 4. Interrupt laufen.

Hier endet das Kapitel über MOVE-BEFEHLE.

BILD

Der BILD-Befehl dient zur kompakten Speicherung von Bildschirmkulissen. Obwohl er speziell für Plattformspiele geschaffen ist, sollte er sich für nahezu jedes Spielformat nützlich erweisen. Sowohl Datenkompression als auch schnelle Produktion von Kulissen sind dadurch möglich. Dies ist im Prinzip nicht einfach, da die Information in einer "Bit-Matrix" gespeichert wird, die in einem Sprite enthalten ist. Jedes eingestellte Bit entspricht einem gePUTetem Sprite und jedes Sprite, das nicht eingestellt wurde, entspricht einem leeren Bereich mit den Abmessungen jenes Sprites, das sonst gePUTet wird. Der Befehl hat bloß fünf Parameter.

COL wird zur Bestimmung der Spalte verwendet, in der der Aufbau beginnt. COL kann einen negativen Wert enthalten. Diese Funktion kann zur Bewegung durch die 'Tafel' verwendet werden.

ROW wird zur Bestimmung der Zeile verwendet, in der der Aufbau beginnt. ROW kann auch einen negativen Wert enthalten.

KEY wird durch Bestimmung der Operationsart benutzt, mit der BILD die Daten auf den Bildschirm bringt (PUT). Es werden vier Operationsarten unterstützt, die hier zusammengefaßt sind:

KEY-Wert	Operation
0	
1	Blocküberschreibung
2	Exklusiv-ODER
3	PUT 'vor' den am Bildschirm vorhandenen, aktuellen Daten
	PUT 'hinter' den am Bildschirm vorhandenen, aktuellen Daten

SPN enthält die Nummer des Sprites mit der Bit-Tafel. Die Abmessungen dieses Sprites sind in keiner Weise begrenzt und es können daher sehr große Kulissen gespeichert werden, deren Höhen und Breiten mehreren Bildschirmen entsprechen.

SP1 enthält die Nummer des Sprites, das zum Aufbau der Kulisse verwendet wird. Eine '1' in der Bit-Tafel wird das Sprite auf den Bildschirm bringen (PUT) und eine '0' hinterläßt eine Lücke mit den Abmessungen des Sprites, das sonst auf den Bildschirm gebracht wird (PUT). Beachten Sie, daß dabei nur eine Lücke geschaffen wird und kein Leerbereich entsteht. Man kann gemeinsam mit einem leeren Sprite INVV verwenden, um Leerbereiche zu löschen.

Unter Verwendung des Sprites 31 als Baublock, werden wir in unserem nächsten Beispiel ein 'P' erzeugen. Das Datensprite 30 benutzt lediglich 8 Bytes zur Erzeugung einer 8x8 großen Matrix. Sprite 31 ist 10 Bytes breit (1/8 der Bildschirmbreite) und 25 Bildpunkte hoch (1/8 der Bildschirmhöhe) – d.h. der ganze Bildschirm wird ausgenutzt.

```
5 ' BEISPIEL VON BILD
10 !SPN,30:!HGT,8:!LEN,1:!CSPR
20 X%=0: Y%=0
30 !ISPR,@Y%,@Y%,@Y%,@X%
40 FOR I=X% TO X%+7:READ A:POKE I,A:NEXT I
50 !SCLS
80 DATA 252,102,102,124,96,96,240,0
```

Zeile 10 Erzeugt ein Datensprite für BILD.
Zeile 20 Leitet X% und Y% ein.
Zeile 30 Verwendet ISPR zur Auffindung der Datenspriteadresse.
Zeile 40 Liest in der Zeile 80 eine Datenanweisung und speichert (POKE) in das BILD-Sprite.
Zeile 50 Löscht den Bildschirm.
Zeile 60 Bringt den BILD-Befehl zur Ausführung, der die Daten aus dem Datensprite liest und Sprite 31 auf den Bildschirm bringt.
Zeile 70 ist eine festgehaltene Schleife.
Zeile 80 ist die Datenanweisung (Achten Sie darauf, daß nach Erzeugung des BILD-Datensprites die zahlen 10, 20, 30, 40 und 80 nicht mehr erforderlich sind).

In unserem zweiten Beispiel werden wir Sprite 30 wieder als unser Datensprite benutzen, um ein 'P' zu erzeugen. Wir werden diesmal jedoch Sprite 30 umkehren und Sprite 32 (ein leeres Sprite) verwenden, um zu garantieren, daß die Lücken keinen 'Müll' enthalten. Vor dem Aufbau (BILD) hätten wir natürlich auch SCLS oder CLSV verwenden können. Da wir aber nicht immer den ganzen Bildschirm ausfüllen und Bereiche des Bildschirms häufig nicht benutzen möchten, ist dies nicht notwendig.

```
5 ' BEISPIEL VON BILD II
10 X%=0:Y%=0
20 !SPN,32:!CLSS
30 !SPN,30:!ISPR,@Y%,@Y%,@Y%,@X%
40 FOR I=X% TO X%+7:READ A:POKE I,A:NEXT I
50 !SPN,30:!INVS
60 !COL,0:!ROW,0:!HGT,200:!LEN,80:!KEY,0:!SPN,30:!SP1,32:!BILD
70 !SPN,30:!INVS
80 !COL,0:!ROW,0:!HGT,200:!LEN,80:!KEY,0:!SPN,30:!SP1,31:!BILD
90 GOTO 90
100 DATA 252,102,102,124,96,96,240,0
```

Zeile 10 leitet Variablen X% und Y% ein.
Zeile 20 ersetzt Bausprite 32 durch INK 0 mittels CLSS.
Zeile 30 ermittelt die Adresse von Sprite 30.
Zeile 40 überführt (POKE) die in der Datenanweisung auf Zeile 100 enthaltenen Daten in das Datensprite 30.
Zeile 50 kehrt das Datensprite um, so daß der Bereich um 'P' jene Daten darstellt, die von BILD gelesen werden.
Zeile 60 verwendet BILD, um alle Bereiche (Sprite 32) auf den Bildschirm zu bringen.
Zeile 70 kehrt Datensprite um (Datenrücksetzung).
Zeile 80 benutzt BILD, um das 'P' auf den Bildschirm zu bringen.
Zeile 90 stellt eine festgehaltene Schleife dar.
Zeile 100 ist jene Datenanweisung, die Daten von 'P' enthält.

In unserem dritten Beispiel werden wir ein 'Q' erzeugen. Diesmal werden wir jedoch anstelle eines zweiten, leeren Sprites Sprite 31 benutzen, indem wir es löschen und danach wieder zurückbringen (GET).

```
5 ' BEISPIEL VON BILD III
10 X%=0:Y%=0
20 !SPN,30:!ISPR,@Y%,@Y%,@Y%,@X%
30 FOR I=X% TO X%+7:READ A:POKE I,A:NEXT I
40 !COL,0 :!ROW,0 :!HGT,200:!LEN,80:!KEY,0 :!SPN,30:!SP1,31:!BILD
50 !SPN,31:!CLSS :!SPN,30 :!INVS
60 !COL,0 :!ROW,0 :!HGT,200:!LEN,80:!KEY,0 :!SPN,30:!SP1,31:!BILD
70 !SPN,31:!COL,10:!ROW,25 :!HGT,25:!LEN,10:!GTBL
80 GOTO 80
90 DATA 56,108,198,198,218,204,118,0
```

Zeile 20 ermittelt die Adresse von Sprite 30.
Zeile 30 überführt (POKE) die in der Datenanweisung a[illegible]eile 90 enthaltenen Daten in das Datensprite 30.
Zeile 40 bringt mittels BILD das 'Q' auf den Bildschirm.
Zeile 50 ersetzt mittels CLSS das Ziegelsprite (Sprite 31) durch INK 0 und kehrt das Datensprite um.
Zeile 60 stellt mit BILD unter Verwendung des gelöschten Ziegelsprites den Bereich um das 'Q' her.
Zeile 70 holt (GET) den Ziegel vom Bildschirm in das Sprite 31 zurück.
Zeile 80 stellt eine festgehaltene Schleife dar.
Zeile 90 ist jene Datenanweisung, die die Daten von 'Q' enthält.

Kollisionserfassung und Musterregistrierung

Wir haben bereits gesehen, wie die Kollisionserfassung mit GETs, PUTs und MOVEs funktioniert. Einer der Nachteile besteht darin, daß die Ausführungsgeschwindigkeit etwas verlangsamt wird. Um das Paket flexibler zu machen, haben wir einen weiteren Befehl eingegliedert, der eine Bildschirmfläche oder eine Sprite nach Bildpunktdaten abtastet (SCAN) und, sollten Daten entdeckt werden (Bildpunkte ungleich INK 0), eine ganzzahlige BASIC-Variable erhöht. Diese Gruppe umfaßt drei Befehle:

SCNV,@V1 Tastet das durch COL, ROW, HGT und LEN definierte Fenster solange ab, bis Daten aufgespärt werden. Nach Entdeckung von Daten wird die ganzzahlige Variable V1 erhöht und eine weitere Ausführung findet nicht mehr statt. V1 muß ganzzahlig sein.

SCNS,@V1 Tastet das durch SPN dekfinierte Sprite ab. Auch hier wird keine weitere Ausführung stattfinden, nachdem Daten entdeckt wurden, und die ganzzahlige Variable V1 wird erhöht.

SCNP,@V1 Tastet das durch SPN, ROW, COL, HGT und LEN definierte Fenster ab. Ausführung wird abgebrochen, nachdem Daten gefunden wurden, und die ganzzahlige Variable V1 wird erhöht.

Diese Befehle können alleine verwendet werden, sie werden aber gewöhnlich in Verbindung mit logischen GETs und PUTs sowie Blindsprites verwendet, um Kollisionen mit bestimmten Gegenständen usw. zu erfassen.

Bei unserem ersten Beispiel setzen wir ein hüpfendes Sprite in ein leeres Fenster und betrachten die linke, obere Ecke, um zu sehen, ob das Sprite durchgeht.

```
5 ' BEISPIEL VON SCNV
10  !SCLS
20  !COL,9 :!ROW,48:!HGT,120:!LEN,66:!INVV
30  !COL,10:!ROW,50:!HGT,115:!LEN,64:!INVV
40  X%=0
50  !SPN,15:!DSPR  :!HGT,1  :!LEN,4 :!CSPR
60  !COL,10:!ROW,50:!HGT,100:!LEN,60
70  !BWST
80  !SET,10:!COL,10:!ROW,50:!HGT,16:!LEN,8
90  !SET,2 :!KEY,15:!SP1,8 :!SP2,8 :!SP3,8:!SP4,8:!HGT,2:!LEN,1
100 !SPN,8 :!COL,30:!ROW,50:!PTXR
110 !SET,2 :!XBNC,1,1
120 !SET,10:!SCNV,@X%
130 IF X%>6 THEN GOSUB 150
140 GOTO 110
150 SOUND 1,30,25
160 LOCATE 11,24:PRINT "SPRITE IST IN DER ECKE":FOR A=1 TO 1000:NEXT A
170 LOCATE 11,24:PRINT "                    "
180 X%=0
190 RETURN
```


Zeile 10 löscht den Bildschirm.
Zeile 20 kehrt ein Fe[illegible]er um, das um mindestens 2 Bildpunkte größer ist als das Fenster, das wir verwen[illegible] werden.
Zeile 30 definiert ein kleineres Fenster und kehrt es um. Es wird also ein roter Rand entstehen.
Zeile 40 Die Kollisonserfassungsvariable X% wird definiert.
Zeile 50 Aus Sprite 15 wird ein Sprungfenster-Datensprite gebildet.
Zeile 60 Die Fensterabmessungen werden in die entsprechenden Variablen geladen.
Zeile 70 Die obigen Daten werden in das Sprite 15 überführt.
Zeile 80 Dem SET wird eine Nummer und Information zugeordnet. Set 10 wird verwendet, um das Fenster zu halten, das wir abtasten.
Zeile 90 Set 2 enthält die Informationen für das Sprungsprite. Der Key-Wert wurde auf 15 gesetzt.
Zeile 100 Da wir 'XBNC' verwenden, müssen wir für Sprite 8 PTXR durchführen.
Zeile 110 Der eigentliche Sprungbefehl mit Rahmenrücklauf-Synchronisierung.
Zeile 120 Abtasten des Fensters in Set 10.
Zeile 130 Kontrolliert, ob X% erhöht wurde.
Zeile 140 Bildet mit Zeile 110 eine Schleife. Dieses Programm muß mit der Pausentaste angehalten werden.
Zeilen 150, 160 und 170 Nach Erfassung einer Kollision erklingt kurzseitig ein Ton und eine entsprechende Nachricht wird angezeigt und kurz darauf wieder gelöscht.
Zeilen 180 und 190 X% wird auf 0 zurückgesetzt und es erfolgt ein Rücksprung zur Hauptschleife.

Musterregistrierung

Die Datenerfassung ist manchmal nicht ausreichend und der Gegenstand muß daher indentifiziert werden. Dies verursacht natürlich ein Problem, das auf verschiedene Weise gelöscht werden kann. Wenn zwei identische Bilder mit einer XOR-Verknüpfung verbunden werden, wird das Ergebnis Null betragen, d.h. es sind alle auf INK 0 eingestellt. In unserem ersten Beispiel werden wir den ganzen Bildschirm nach Großbuchstaben 'A' abtasten (Farbe muß ebenfalls abgestimmt werden). Wenn der Buchstabe gefunden wird, wird er in einen Kleinbuchstaben umgewandelt. Achten Sie darauf, daß für den Vergleich ein Blindsprite verwendet wird.

```
5 ' BEISPIEL VON MUSTERERKENNUNG
10  X%=0
20  !SCLS
30  !SPN,12: FOR I=1 TO 50:!COL,RND*70:!ROW,RND*174 :!PTIF:NEXT I
40  FOR I=1 TO 20:LOCATE RND*38+1,RND*23+1:PRINT "A":NEXT I
50  !HGT,8:!LEN,2
60  FOR R=0 TO 200 STEP 8
70   FOR C=0 TO 80 STEP 2
80     !COL,C :!ROW,R
90     !SPN,33:!PTXR:!SPN,34:!GTBL:!SPN,33:!PTXR
100    !SPN,34:!SCNS,@X%
110    IF X%=0 THEN GOSUB 150
120    X%=0
130 NEXT C:NEXT R
140 END
150 !SPN,35:SOUND 1,40,10:!PTBL:X%=0:RETURN
```

Zeilen 10 bis 50 Die Erfassungsvariable wird angegeben und der Bildschirm wird gelöscht. Danach werden sowohl Daten als auch 'A'-Zeichen in beliebiger Folge über den Bildschirm verteilt. Anschließend werden HGT und LEN auf jene Werte gebracht, die zum Abtasten verwendet werden, d.h. 1 Zeichen.
Zeilen 55 und 60. Die Zeilen werden erhöht, nachdem eine ganze Zeile abgetastet wurde.
Zeilen 80 und 90 COL und ROW werden eingestellt und Sprite 33 wird danach mit dem Zeichen, das wir abtasten wollen, mittels Exklusiv-ODER verknüpft. Wir werden eine Leerstelle erhalten, wenn das durch eine XOR-Verknupfung auf den Bildschirm gebrachte Zeichen ein 'A' ist. Anschließend wird Sprite 34 mit dieser Information gefullt.
Zeile 100 Sprite 34 wird abgetastet. Wenn Sprite 34 leer ist, dann stellen die Daten 'A' dar.
Zeile 110 Wenn keine Daten festgestellt wurden, dann GOSUB Zeile 150.
Zeile 120 X% wird rückgesetzt.
Zeile 130 Die Schleife kehrt zur Zeile 60 zurück.

und die Subroutine kehrt zum Hauptprogramm zurück.

Es ist nicht unbedingt notwendig, das ganze Zeichen z rgleichen, da der abgetastete Gegenstar manchmal hervorstechende Merkmale besitzt, die wir nutzen können, um die ganze Abtastung z vermeiden. Wir werden jetzt versuchen, das vorhergehende Beispiel zu wiederholen und einen T des 'A'-Zeichens als den zu identifizierenden Gegenstand zu verwenden. Dieses Beispiel ist u vieles einfacher, da 'A' immer auf Zeichengrenzen gesetzt wird.

```
5 ' BEISPIEL VON MUSTERERKENNUNG II
10  X%=0
20  !SPN,34:!DSPR:!HGT,3:!LEN,1:!CSPR
30  !SCLS
40  !SPN,12: FOR I=1 TO 50:!COL,RND*70:!ROW,RND*174:!PTIF:NEXT I
50  FOR I=1 TO 20:LOCATE RND*38+1,RND*23+1:PRINT "A":NEXT I
60  !HGT,8:!LEN,2
70  FOR R=0 TO 192 STEP 8
80    FOR C=0 TO 78 STEP 2
90    !COL,C :!ROW,R
100   !SPN,33:!SCL,0:!SRW,0:!HGT,3:!LEN,2:!PWXR:!SPN,34:!GWBL:!SPN,33:!PWXR
110   !SPN,34:!SCNS,@x%
120   IF X%=0 THEN GOSUB 160
130   X%=0
140 NEXT C:NEXT R
150 END
160 !SPN,35:SOUND 1,40,10:!PTBL:X%=0:RETURN
```

Zeile 10 X% wird angegeben.
Zeile 20 Sprite 34 wird gelöscht und mit einer Höhe von 3 Bildpunkten und einer Breite von 1 Byte neu aufgebaut.
Zeile 30 Der Bildschirm wird gelöscht und Sprites sowie Buchstaben 'A' werden in beliebiger Folge am Bildschirm angezeigt.
Zeilen 100 und 110. Statt das ganze Zeichen in Sprites 33 und 34 zu geben, werden nur die oberen 3 Bildpunkte verwendet, wodurch das Abtasten erheblich beschleunigt wird.

Das Problem kompliziert sich ferner, wenn der Gegenstand, den wir suchen, sich 'vor' fremden Daten befindet. In diesem Fall müssen wir alle fremden Daten durch eine logische UND-Verknüpfung mit dem Testgegenstand isolieren und anschließend eine XOR-Verknüpfung benutzen, um festzustellen, ob der Gegenstand vorhanden ist. Im nachfolgenden Beispiel wird dies illustriert.

```
5 ' BEISPIEL VON MUSTERERKENNUNG III
10  !SCLS
20  !SPN,34:!DSPR:!HGT,8:!LEN,2:!CSPR:!CLSS
30  FOR N=1 TO 400 STEP 4:PLOT 1,N:DRAW 650,N,2:NEXT N
40  FOR I=1 TO 20:!ROW,INT(RND*24)*8:!COL,INT(RND*40)*2:!SPN,33:!PTIF:NEXT I
50  FOR C=0 TO 78 STEP 2
60    FOR R=0 TO 192 STEP 8
70    X%=0
80    !COL,C :!ROW,R :!INVV :!INVV
90    !SP1,34:!SP2,33:!SCL,0:!SRW,0:!GMBL:!SPN,34:!GTND:!GMXR:!SCNS,@X%
100   IF X%=0 THEN GOSUB 130
110 NEXT R:NEXT C
120 END
130 !SPN,35:SOUND 1,40,10:!PTBL:RETURN
```

Zeile 30 zeichnet jene Linien auf den Bildschirm.
Zeile 40 bringt in beliebiger Folge 20 'A'-Zeichen auf den Bildschirm.
Zeilen 50 und 60 sind die FOR-NEXT-Schleifen, die den ganzen Bildschirm abtasten.
Zeile 70 Setzt X% auf null.
Zeile 80 Stellt COL und ROW ein und kehrt den ganzen, abgetasteten Bildschirm um (INVERTS). (Daraus können Sie erkennen, welcher Bereich abgetastet wird.)
Zeile 90 überführt die Daten von COL und ROW mittels GMBL in den Speicher. Danach wird für diese Daten ein GTND und ein GTXR ausgeführt. Sprite 34 wird nach Daten abgetastet.
Zeile 100 kontrolliert, ob Daten vorhanden sind. Wenn keine vorhanden sind, wurde das 'A' registriert und die Subroutine der Zeile 30 wird aufgerufen.
Zeile 110 springt zurück.
Zeile 120 beendet (END) das Programm.
Zeile 130 ist eine Subroutine, die einen Piepton (BEEP) erzeugt und einen Kleinbuchstaben 'a' auf den Bildschirm bringt, um die Registrierung von 'A' anzuzeigen.

Dieser Test kann manchmal versagen, wenn ein Gegenstand alle Daten, die wir zu testen hatten, und noch mehr enthält. Im folgenden Beispiel wird dieser Nachteil illustriert.

```
5 ' BEISPIEL VON SCNS
10  !SCLS  :!SPN,34:!CLSS
20  !SPN,12: FOR I=1 TO 50:!COL,RND*70:!ROW,RND*174:!PTIF:NEXT I
30  FOR I=1 TO 10:!ROW,INT(RND*24)*8:!COL,INT(RND*40)*2:!SPN,33:!PTIF:NEXT I
40  FOR N=1 TO 20: LOCATE (RND*39)+1,(RND*20)+1:PRINT CHR$(143):NEXT N
50  FOR C=0 TO 78  STEP 2
60   FOR R=0 TO 192 STEP 8
70    X%=0
80    !COL,C :!ROW,R :!INVV :!INVV
90    !SP1,34:!SP2,33:!SCL,0:!SRW,0:!GMBL:!SPN,34:!GTND:!GMXR:!SCNS,@X%
100   IF X%=0 THEN GOSUB 130
110 NEXT R:NEXT C
120 END
130 !SPN,35:SOUND 1,40,10:!PTBL:RETURN
```

Zeile 10 löscht den Bildschirm.
Zeile 20 bringt 50 'Autosprites' in beliebiger Folge auf den Bildschirm.
Zeile 30 bringt 10 'A'-Zeichen auf den Bildschirm.
Zeile 40 bringt 20 gelbe Blöcke in beliebiger Folge auf den Bildschirm. Dieser Block enthält alle erforderlichen Daten, die wir zu testen haben. Die Abtastroutine wird daher veranlasst, einen Kleinbuchstaben 'a' zu setzen.
Zeilen 50 und 60 sind die FOR-NEXT-Schleifen, die die ROW- und COL-Werte errechnen.
Zeile 70 stellt X% auf 0.
Zeile 80 stellt ROW und COL ein und kehrt den Bildschirmteil, der gerade untersucht wird, um
Zeile 90 sucht nach einem 'A'.
Zeile 100 Falls ein 'A' oder, wie in diesem Beispiel, ein gelber Ziegel vorhanden ist, wird die Subroutine in Zeile 130 aufgerufen.
Zeile 110 springt zurück.
Zeile 120 beendet (END) das Programm.
Zeile 130 ist jene Subroutine, die feststellt, ob 'A' registriert wurde.

Wenn der zu testende Gegenstand 'hinter' Bildschirmdaten liegt, können wir dies nicht ohne Verwendung einer speziell dafür bestimmten Methode durchführen.

Die bisher betrachteten Musterregistriermethoden können als sogenannte 'exakte' Methoden bezeichnet werden. Das heißt, wenn zwei Gegenstände eine XOR-Verknüpfung erfahren und das Ergebnis null beträgt, dann sind sie ganz sicher identisch. Wir werden uns jetzt einige 'Annäherungsmethoden' betrachten, die relativ schnell und einfach durchführbar sind, aber keine eindeutigen Ergebnisse bieten und daher mit einer gewissen Vorsicht behandelt werden müssen. Hier werden drei neue Befehle eingeführt

...sche Summe aller Daten. Das Result... verbleibt in der BASIC-Ganzzahlvariable ...1.

SUMS,@V1 Dieser Befehl produziert die Summe aller Daten in jenem Sprite, dessen Nummer ... SPN gehalten wird, und wird das Ergebnis in der BASIC-Ganzzahlvariable V... hinterlassen.

SUMP,@V1 Dieser Befehl wird in dem durch SPN, ROW, COL, HGT und LEN definierte... Spritefenster die Summe aller Daten produzieren. Das Resultat verbleibt in de... ganzzahligen BASIC-Variable V1.

Bei Verwendung dieser Befehle sollten Sie sich folgende Punkte überlegen:

1. Das Resultat wird in einer 16-Bit-Variable gespeichert. Sollte es also 65536 überschreiten, wird die Zählung wieder von 0 beginnen. In anderen Worten: Das Resultat ist MOD (65536). Dies ergibt bei einer wirklichen Summe von 65538 ein Ergebnis von 2. Es ist daher möglich, daß zwei unterschiedliche Gegenstände nicht unterscheidbar sind.

2. Zusätzlich zur oben beschriebenen 'Zweideutigkeit' macht sich eine weitere Beschränkung dadurch bemerkbar, daß ein neuerliches Ordnen der Daten im Gegenstand die Summe nicht verändert. Zwei Gegenstände können daher vollkommen unterschiedlich sein und dennoch dieselbe Summe ergeben. Folgende Transformationen können an einem Gegenstand ausgeübt werden, ohne dessen Datensumme zu verändern.

   Rollen (mit Umlauf) in vertikaler Richtung, Rollen (mit Umlauf) um ein oder zwei ganze Bytes nach links oder rechts und vertikale Spiegelung.

Es ist jedoch relativ einfach, Gegenstände mit unterschiedlichen Summen zu entwerfen, und die Möglichkeit, daß zwei verschiedene Gegenstände dieselbe 16-Bit-Summe haben, ist sehr gering. Wichtig ist jedoch, daß das Feststellen einer 'Zweideutigkeit' sehr einfach ist. Dies sollte standardgemäß durchgeführt werden.

Die folgenden Beispiele demonstrieren die Verwendung und Nachteile dieser Befehle:

```
5 ' BEISPIEL VON SUMS UND SUMV
10  !SCLS
20  S%=INT(RND*4)+17
30  !COL,0:!ROW,0:!SPN,S%:!PTBL
40  T%=0
50  !HGT,25:!LEN,8:!SUMV,@T%
60  FOR N=17 TO 20
70    !SPN,N:!SUMS,@S%
80      IF S%=T% THEN !COL,10:!PTBL
90  NEXT N
100 LOCATE 10,10
```

Zeile 10 löscht den Bildschirm.
Zeile 20 holt ein beliebiges Sprite von 4.
Zeile 30 bringt dieses Sprite auf die linke, obere Bildschirmfläche.
Zeile 40 gibt die Variable T% an
Zeile 50 tastet ein Fenster mittels SUMV um das Sprite herum ab und speichert das Ergebnis in T%.
Zeilen 60 bis 90 vergleichen die Summenwerte von Sprites 17 bis 20 mit dem in Zeile 50 erhaltenen Wert. Wenn es ähnlich ist, wird es rechts neben dem Originalsprite gesetzt.
Zeile 100 bewegt den Cursor nach unten

Das untere Beispiel gleicht dem obigen Beispiel mit der einen Ausnahme, daß das Sprite am Bildschirm von dem Abtasten geFIPVt wird. SUMV kann es immer noch als ein nichtgeFIPVtes Spritebild erkennen.


```
5 ' BEISPIEL VON SUMS UND SUMV II
10  !SCLS
20  S%=INT(RN...4)+17
30  !COL,0:!ROW,0:!SPN,S%:!PTBL
40  T%=0
50  !HGT,25:!LEN,8:!FIPV:!SUMV,@T%
60  FOR N=17 TO 20
70    !SPN,N:!SUMS,@S%
80      IF S%=T% THEN !COL,10:!PTBL
90  NEXT N
100 LOCATE 10,10
```

Es ist öfters erforderlich festzustellen, ob ein Gegenstand zu einer Reihe von möglichen Gegenständen gehört. Wir können dies natürlich erzielen, indem das durch die SUM-Befehle bewirkte Ergebnis in V1 mit einem vorbestimmten Kandidatenfeld verglichen wird. Da in diesem Fall jedoch relativ langsames BASIC erforderlich ist, haben wir die SUM-Befehle erweitert, um dieses Problem mit Maschinencode-Geschwindigkeiten zu behandeln. Dies wird nicht nur die Geschwindigkeit Ihrer Programme erhöhen, sondern sie auch viel kompakter gestalten. Der Syntax ist wie folgt:

```
SUMV,e1,e2....en,@V1
SUMS,e1,e2....en,@V1
SUMP,e1,e2....en,@V1
```

Die Ausführung des Befehls ist dem vorhin beschriebenen SUM mit einem Parameter sehr ähnlich. Das Ergebnis wird hier jedoch nicht in V1 gesetzt, sondern es wird mit der vor @V1 befindlichen Ausdrucksliste verglichen, um ein gleichwertiges Resultat zu finden. Wenn kein passender Vergleichswert gefunden wird, wird V1 null zugeordnet. Wenn ein passender Vergleichswert gefunden wird, dann wird die Position des ersten Ausdrucks in der Liste, der mit dem Ergebnis übereinstimmt, V1 zugeordnet. Die folgenden Beispiele illustrieren die Verwendung dieser Befehle.

```
5 ' BEISPIEL VON SUMS UND SUMV III
10  !SCLS
20  S17%=0:!SPN,17:!SUMS,@S17%
30  S18%=0:!SPN,18:!SUMS,@S18%
40  S19%=0:!SPN,19:!SUMS,@S19%
50  S20%=0:!SPN,20:!SUMS,@S20%
60  S%=INT(RND*4)+17
70  !COL,0:!ROW,0:!SPN,S%:!PTBL
80  T%=0
90  !HGT,25:!LEN,8:!SUMV,S17%,S18%,S19%,S20%,@T%
100 !COL,10:!SPN,16+T%:!PTBL
110 LOCATE 10,10
```

Zeile 10 löscht den Bildschirm.
Zeilen 20 bis 50 speichern die Summen der Spritedaten in entsprechenden Variablen.
Zeile 60 holt von vier Möglichkeiten ein beliebiges Sprite.
Zeile 70 bringt das Sprite auf den Bildschirm.
Zeile 80 gibt die Variable T% an.
Zeile 90 tastet das Fenster unter Verwendung von SUMV ab und speichert von den 4 Werten den passenden Wert in T% (T% wird ein 0 ergeben, wenn kein passender Wert gefunden wird).
Zeile 100 bringt das passende Sprite (16+T%) auf den Bildschirm.

Hiermit ist der Abschnitt über Kollisionserfassung und Musterregistrierung abgeschlossen. Für diese Funktionen sind ziemlich fortgeschrittene Programmiertechniken erforderlich und entsprechende Übung ist notwendig, um damit vertraut zu werden. Wir haben einige Verwendungsmöglichkeiten vorgestellt, möchten aber darauf hinweisen, daß nach entsprechender Übung hoher entwickelte Programme erzeugt werden können.

Bewegung mit hoher Auflösung

...kleinste Bewegungsschritt eines Sprites in horizontaler Richtung 1 Byte beträgt ... Einschränkungen der 'Byteauflösung'. Die Bildpunkte im 4-Farbenmodus und 2 Bildpunkte im ...-Farbenmodus). Dies läßt sich verbessern, indem Sprites in 'Zwischenpositionen' entworfen werden und nacheinander zwischen Erhöhungen von COL gesetzt werden. Im folgenden Beispiel werden wir in Sprite 18 eine Kopie von Sprite 17 aufbauen, es in eine Zwischenposition rollen und sequentiell anzeigen (PUT). COL wird bei jedem PUT erhöht.

```
5 ' BEISPIEL VON HIRESBEWEGUNG
10 !SCLS
20 !SPN,17:!COL,0:!ROW,0:!PTBL
30 !SPN,18:!DSPR:!HGT,20:!LEN,8:!CSPR:!GTBL:!SSR1
40 !SCLS
50 FOR I=0 TO 80
60 !COL,I
70 !SPN,17:!PTBL
80 !COL,I:!SPN,18:!PTBL
90 NEXT I
```

Zeile 10 Der Bildschirm wird gelöscht.
Zeile 20 Sprite 17 wird auf den Bildschirm gebracht.
Zeile 30 Sprite 18 wird eine Kopie von Sprite 17 (durch Verwendung von GTBL). Danach wird Sprite 18 um einen Bildpunkt nach rechts gerollt.
Zeile 40 Der Bildschirm wird wieder gelöscht.
Zeilen 50 bis 90 Sprites 17 und 18 werden abwechselnd nach jeder Schleife auf den Bildschirm gebracht.

Leider läßt sich diese Technik nicht mit den MOVE-Befehlen verwenden, da die Erhöhung in die X-Richtung konstant sein muß. Dasselbe Ergebnis läßt sich jedoch auch erzielen, wenn Laser-BASIC auf einen Modus von hoher Auflösung eingestellt wird. Der Befehl hierfür ist CLHI (CLLO bringt Laser-Basic auf den normalen Auflösungsmodus zurück).

Im Hochauflösungsmodus wird eine Bildschirmbreite von 160 Spalten erzeugt. Der Wert in COL wird durch zwei geteilt und falls ein Rest verbleibt (d.h. COL hält einen 'ungeraden' Wert statt einen 'geraden'), wird zur Spritenummer eine 1 dazugezählt. Wir konnen also Sprite X mit einer Auflösung von einem 1/2 Byte am Bildschirm entlang bewegen (2 Bildpunkte im 4-Farbenmodus, 1 Bildpunkt im 16-Farbmodus), indem wir Sprite X+1 dasselbe Bild verleihen wie Sprite X, es aber um 1/2 Byte nach rechts verschieben. Es gibt hierfür einen Befehl – HRSP. Die einzige von HRSP verwendete Variable ist SPN. HRSP erzeugt ein zweites Sprite, dessen Spritenummer um eine Nummer größer ist als in SPN (es tritt ein Fehler auf, wenn diese Spritenummer bereits verteilt wurde), kopiert das Sprite SPN in das Sprite SPN+1 und rollt Sprite SPN+1 um ein 1/2 nach rechts. In den folgenden Beispielen zeigen wir, wie HRSP, CLHI und CLLO mit den MOVE-Befehlen verwendet werden.

```
5 ' BEISPIEL VON HRSP
10 !SCLS
20 !CLHI
30 !SPN,18:!DSPR
40 !SPN,17:!HRSP
50 !SPN,17:!SP1,17:!SP2,17:!SP3,17:!SP4,17:!HGT,0:!LEN,1:!COL,-20:!ROW
60 !XMOV,230,1
70 GOTO 60
```

Zeile 10 löscht den Bildschirm
Zeile 20 stellt die Software auf HIRES MODE (Hochauflösungsmodus) ein.
Zeile 30 löscht das alte Sprite 18
Zeile 40 fuhrt an Sprite 17 HRSP aus
Zeile 50 Stellt die Parameter fur XMOV ein.
Zeile 60 bringt Sprite 17 mittels Exklusiv-ODER auf den Bildschirm.
Zeile 80 bewegt das Sprite mittels XMOV.
Zeile 80 springt auf Zeile 70 zuruck

Bitte beachten Sie, daß bei Verwendung von CLHI nur die 'GT'-, 'PT'-, 'GW'-, 'PW'- und 'MOVE'-Befehle betroffen sind. Bildschirmfenster, die von den anderen Befehlen verwendet werden, behandeln den Bildschirm weiterhin als 80 Spalten breit. Die folgenden Beispiele werden dies illustrieren.

```
5 ' BEISPIEL VON CLHI
10 !SCLS
20 !CLHI
30 !SET,1 :!SP1,17 :!SP2,17 :!SP3,17:!SP4,17:!HGT,0:!LEN,1
40 !SPN,17:!COL,-20:!ROW,180:!PTXR
50 FOR X=1 TO 160
60  !SET,1:!XMOV,1,1
70  !SET,2:!COL,RND*160:!ROW,RND*135:!HGT,RND*60:!LEN,RND*30:!IK1,RND*4:!STCV
80 NEXT X
90 !CLLO
```

Zeile 10 löscht den Bildschirm.
Zeile 20 stellt die Hardware auf HIRES MODE ein.
Zeile 30 bildet die Parameter für XMOV.
Zeile 40 bringt Sprite 17 mit Exklusiv-ODER auf den Bildschirm.
Zeile 50 Ist eine Schleife.
Zeile 60 bewegt den Sprite mit XMOV den Bildschirm entlang.
Zeile 70 erzeugt mit STCV ein beliebiges Fenster mit einem beliebigen INK-Wert.
Zeile 80 pringt auf Zeile 50 zurück.

Hintergrundsausführung von Laser-BASIC-Befehlen

Eine der stärksten Funktionen von Locomotive-BASIC besteht darin, daß Subroutinen bei Verwendung von EVERY- und AFTER-Befehle unter Interrupt laufen können. Obwohl sich dies besonders für die Entwicklung von Spielen eignet, liegt hier eine Beschränkung vor. BASIC führt seine Subroutine nicht sofort nach dem Empfang des passenden Interrupt aus, sondern beendet vorher die Ausführung des laufenden BASIC-Befehls. Dies verursacht eine gewisse Unsicherheit im Hinblick auf die Position des Punktes, der den Bildschirm 50-mal pro Sekunde abtastet und das Bild aufbaut, das Sie am Bildschirm sehen. Diese leichte Unsicherheit kann Flickern verursachen. Um dieses Problem zu umgehen, ist Laser-BASIC mit einem eigenen Interrupt-Mechanismus ausgestattet, der sofort zur Ausführung gelangt, wenn das Interrupt empfangen wird. Diese Ausführungsart wird als Hintergrundausführung bezeichnet, da solche Routinen unabhängig von der Maschine weiterlaufen. Sie können sogar Ihr neues Programm eingeben, während im Hintergrund noch Routinen ablaufen. Hintergrundprogramme werden durch GSPR, MSPR und PSPR automatisch beendet. Sie MÜSSEN aber beendet werden, bevor Locomotive-BASIC ein Band bzw. eine Diskette anspricht. Zur Hintergrundausführung gehören drei Befehle – ISET, IRUN und IEND.

ISET,@A$

Dieser Befehl weist Laser-BASIC darauf hin, welche erweiterten Befehle im Hintergrund laufen sollen und welcher Variablensatz (SET) von den einzelnen Befehlen verwendet werden soll. Die Information wird in Form einer Zeichenkette weitergegeben:

"Befehl,Satz,Befehl,Satz....#"

Es gibt also keine Leerzeichen oder sonstige Abgrenzungszeichen in der Zeichenkette. Die Sätze werden durch die Buchstaben 'A' bis 'P' dargestellt. Sämtliche Zeichen müssen in Großbuchstaben eingegeben werden. Wenn wir z.B. ein durch SET 0 definiertes Fenster rollen und ein Sprite innerhalb der Parameter in SET 10 bewegen mochten, wurden wir folgendes verwenden

A$="WVR1AXMOVK#":|ISET,@A$

Laser-BASIC weiß jetzt Bescheid, was ausgefuhrt werden soll. Wir haben jedoch das Programm immer noch nicht zum Laufen gebracht. Um dies zu ermoglichen, mussen wir den IRUN-Befehl verwenden.

Dieser Befehl gleicht dem EVERY-Befehl von Locomotiv BASIC. In diesem Fall wird aber lediglich ein Parameter benötigt – ein BASIC-Ausdruck der die Ausführungsfrequenz einstellt. Der Ausdruck kann einen beliebigen Wert von 0 bis 65535 besitzen und bestimmt die Anzahl der Interrupts, die zwischen den aufeinanderfolgenden Ausführungen erlaubt sind. Ein 0-Wert wird bei jedem Interrupt eine Ausführung erzwingen und der Wert 1 bei jedem zweiten Interrupt, usw. Einige Punkte müssen hierzu beachtet werden.

1. Laser-BASIC wird bei jedem Rahmenrücklauf ein Interrupt anerkennen (bei jedem Fünfzigstel einer Sekunde). Die maximale Ausführungsgeschwindigkeit ist also 50-mal pro Sekunde (in Einklang mit IRUN,0).

2. Sollten Sie eine Ausführungsgeschwindigkeit von 0 wählen (IRUN,0) und die Hintergrundroutine benötigt mehr als eine 50stel Sekunde für die Ausführung, dann wird die Kontrolle von der Hintergrundroutine nicht zurückkehren. Dies ist recht häufig der gewünschte Effekt und Programmaussprung ist nur durch Drücken der ESC-Taste möglich.

3. Wenn Sie eine Ausführungsgeschwindigkeit von 1 oder mehr wählen, dann wird immer eine Pause von einer 50stel Sekunde oder mehr zwischen aufeinanderfolgenden Ausführungen erlaubt, egal wie lange die Ausführung der Hintergrundroutine dauert.

4. Sollte im Vorder- oder Hintergrundprogramm ein Laser-BASIC-Fehler (durch zwei Sternchen angedeutet) auftreten, dann wird das Hintergrundprogramm abgebrochen. Wenn die ESC-Taste während der Ausführung des Hintergrundprogramms gedruckt wird, dann wird das Hintergrundprogramm ebenfalls abgebrochen.

IEND

Dieser Befehl bricht ein Hintergrundprogramm ab. ISET oder IRUN dürfen nicht ausgeführt werden, während ein Hintergrundprogramm läuft.

Hintergrundbefehle

Alle Laser-BASIC-Befehle können nicht im Hintergrund ausgeführt werden. Befehle, die folgende Parameter benötigen, können nicht im Hintergrund ausgeführt werden Diejenigen Befehle, die wahlweise folgende Parameter haben, können nur ohne deren wahlweisen Parametern ausgeführt werden. Das Letztere verursacht gewöhnlich keine Schwierigkeiten. Kollisionserfassung erfolgt durch Verfolgungssprites. Diese Sprites werden in einem späteren Abschnitt näher beschrieben. Befehle, die unter Interrupt ausgeführt werden können, sind im Abschnitt 'CLASS OPTIONS' näher beschrieben

TON

Laser-BASIC hat nur einen Befehl für Toneffekte und Musik: dieser Befehl heißt PLAY. Wie wir aber sehen werden, gibt es noch 20 zusätzliche Anweisungen, die mit dem PLAY-Befehl benutzt werden können und die den vom Betriebssystem der Maschine gebotenen Funktionen entsprechen. Toneffekte werden mehr oder weniger genauso behandelt wie Verfolgungssprites; 'Melodien' werden in Sprites gespeichert. Es gibt insgesamt 20 Steuercodes. Jeder Code kann von einem oder mehreren Datensprites gefolgt werden. Das Spielen mit Ton lernt man am einfachsten durch das in diesem Paket enthaltene Tonerzeugerprogramm kennen.

Das Tonerzeugerprogramm

LOAD und RUN des Tonprogramms (Laden und Laufen):

Laser-BASIC laden und danach 'SNDGEN laden. Verwenden Sie dabei:

Band: Das Band zum, Anfang des Tonerzeugers zurückspielen und RUN" eintippen (s.Bandkarte).

Diskette: Diskette einlegen und RUN"SNDGEN eintippen.

Auf dem Bildschirm w ein Menü erscheinen. Wir können jetzt eine Wahlfunktion nach der anderen behandeln.

Anmerkung: Bevor Sie RUN (Lauf) beginnen, müssen Sie feststellen, ob die Tastatur auf Großbuchstaben eingestellt ist. (Wenn nicht, müssen Sie CAPS SHIFT drücken).

Wahlfunktion 1 – ENTER SOUND PROGRAM (Tonprogramm eingeben)

Um diese Wahlfunktion zu wählen, müssen Sie zuerst "1" eintippen und danach ENTER. Sie werden zuerst die Nachricht "ENTER TARGET SPRITE" (Zielsprite eingeben) erhalten. Dieses sollte eine Nummer von 1 bis 255 haben (sonst würde ein Bereichsfehler auftreten). Ist das Sprite bereits vorhanden, dann haben Sie die Wahl, es zu benutzen. Wenn Sie es benutzen wollen (drücken Sie hierfür ganz einfach die "Y"-Taste), können Sie mit der Programmeingabe sofort beginnen. Wenn Sie es nicht benutzen wollen (Drücken der 'N'-Taste), dann wird das Programm auf "ENTER TARGET SPRITE" zurückkehren. Ist das Sprite noch nicht vorhanden, dann haben Sie die Möglichkeit es durch "CREATE IT" (Y/N) zu erzeugen. Wenn Sie "N" drücken, werden Sie zur Nachricht "ENTER TARGET SPRITE" zurückkehren Wenn Sie aber "Y" drücken, dann werden Sie dazu aufgefordert, die Spriteabmessungen HGT und LEN einzugeben. Sie können ohne weiteres großzügig sein – wenn Sie ein Sprite erzeugen, das größer ist als Sie unbedingt erfordern, können Sie es zu einen späteren Zeitpunkt immer noch verkleinern (crunch). Wenn das Sprite aber zu klein ist, dann ist alles verloren. Wenn Sie ein Sprite erzeugen, das mehr Speicherplatz benötigt als vielleicht vorhanden ist, dann wird die Nachricht "INSUFFICIENT MEMORY" (Speicherraum nicht ausreichend) erscheinen. Sie werden dann aufgefordert HGT und LEN neu einzugeben. Der Speicherraum für das Programm beträgt dann HGTxLEN-4. Der 'Crunch-Vorgang' (Verkleinerung) wird auch einen gewissen Speicherraum beanspruchen – seien Sie also nicht zu großzügig!

Nachdem das Zielsprite einmal aufgebaut wurde, können Sie beginnen das Programm einzugeben. Die Nachricht "ENTER INSTRUCTION" (Anweisung eingeben) wird erscheinen. Sie müssen die Anweisung zuerst eintippen und dann die ENTER-Taste drücken. Es wird dann eine Liste mit gültigen Anweisungen und Informationen über deren Anforderungen folgen. Bevor wir auf die einzelnen Anweisungen eingehen, müssen wir darauf hinweisen, daß die Mnemoniks für die einzelnen Anweisungen etwas zu lang sind, und daß Sie diese später abkürzen können, wenn Sie damit mehr vertraut sind. Die Mnemoniks werden in den Datenanweisungen in den Zeilen 1000 bis 1040 gelistet. Die Reihenfolge der Anweisungen darf nicht geändert werden. Sie können diese allerdings editieren und eine Kundenkopie davon anfertigen. Der LIST-Melodie-Befehl wird diese neuen Anweisungen ebenfalls registrieren. Und nun zu den Anweisungen selbst.

"TON"

Fast alle Tonprogramme enthalten die 'SOUND'-(Ton)- bzw. die 'WAIT-SOUND' (Tonverzögerung)-Anweisung. Diese wird gewöhnlich als erste Anweisung des Programms eingegeben, so daß die 'CALL'-Anweisung (Aufrufanweisung) benutzt werden kann (um Programmplatz zu sparen). Wenn die erste Anweisung keine 'SOUND'- oder 'WAIT-SOUND'-Anweisung ist, und keine der 'CALL'-Anweisungen ausgeführt wird, dann wird beim Programmablauf "ILLEGAL TRACKER CODE" (ungültiger Verfolgercode) erzeugt. In der Praxis wird aber die 'SOUND'-Anweisung selten verwendet (s. 'WAIT-SOUND'), da die Ausführung bei der nächsten Anweisung fortgesetzt wird, falls Sie der Warteschlange keinen 'SOUND/ (Ton) hinzufügen. Falls dabei eine 'Melodie' eingegeben wird, ist dies nicht besonders wünschenswert, da die Note verlorengeht. Die 'SOUND - und 'WAIT-SOUND'-Anweisungen werden folgende Meldungen erzeugen:

'CHANNEL STATUS" (Kanalstatus) wird am Anfang angezeigt und gefolgt durch.

CHANNEL A (Y/N)" (Kanal A -J/N-) Wenn Sie wunschen, daß der Ton an Kanal A ausgegeben wird, wahlen Sie "Y", ansonsten "N".

'CHANNEL B (Y/N") (Kanal B -J N-) Sollte der Ton an Kanal B und anstatt Kanal A ausgegeben werden, dann tippen Sie "Y", ansonsten "N".

ansonsten "N".

"RENDEZVOUS WITH A" (Rendezvous mit A ... sollte der Ton mit einem Ton an Kanal A zusammentreffen, dann tippen Sie "Y", ansonsten "N".

"RENDEZVOUS WITH B" Sollte der Ton mit einem Ton an Kanal B zusammentreffen, dann tippen Sie "Y", ansonsten "N".

"RENDEZVOUS WITH C" Sollte der Ton mit einem Ton an Kanal C zusammentreffen, dann tippen Sie "Y", ansonsten "N".

"HOLD Y/N" (Anhalten J/N) Sollte der Ton am Anfang der Warteschlange angehalten werden, bis er freigegeben wird, dann tippen Sie "Y", ansonsten "N".

"FLUSH (Y/N)" (Ausräumen J/N) Sollte der Ton sämtliche, davorstehende Töne in der Warteschlange ausräumen, dann tippen Sie "Y", ansonsten "N".

"ENTER AMPLITUDE ENVELOPE NUMBER" (Nummer der Amplitudenhüllkurve eingeben) Sie sollten eine Nummer zwischen 0 und 15 eingeben, welche die Amplitudenhüllkurve auswählt, die vom Ton benutzt wird. Die Lautstärke wird während der ganzen Tonausgabe konstant bleiben, wenn Sie 0 eintippen. Wählen Sie jedoch eine Hüllkurvennummer zwischen 1 und 15 dann müssen Sie sie mit "AMP-EMV" aufbauen, bevor der Ton ausgeführt wird. (Sie kann auch von BASIC aufgebaut werden. Vergessen Sie aber nicht, daß sie von "RUN" zerstört wird).

"ENTER TONE ENVELOPE NUMBER" (Nummer der Klanghüllkurve eingeben) Dies ist eine Nummer zwischen 1 und 15, die die zu verwendende Klanghüllkurve bestimmt. Die Hüllkurve muß auch mit "TONE-ENV" aufgebaut werden, bevor die Ausführung beginnt. (oder den Aufbau von BASIC beginnen und "RUN" vermeiden).

"ENTER TONE PERIOD" (Klangdauer eingeben) Dies ist eine Nummer zwischen 0 und 4095, welche die Tonhöhe (PITCH) der Note einstellt. (Sie kann durch die Tonhülle geändert werden). Ein hoher Wert ergibt eine tiefe Note und ein niedriger Wert ergibt eine hohe Note. Die Klangperioden, die echten Werten entsprechen, werden in dem BASIC-Begleithandbuch Ihrer Maschine näher beschrieben.

"ENTER NOISE PERIOD" (Schallperiode eingeben) Dies ist eine Nummer zwischen 0 und 31. Ein Nullwert wird im allgemeinen für 'Melodien' verwendet und entspricht 'schallos'

"ENTER INITIAL VOLUME" (Anfangslautstärke eingeben) Dies ist ein Wert zwischen 0 und 15 und entspricht der Lautstarke, mit der die Note zu spielen beginnt (kann durch Amplitudenhüllkurve verstellt werden). Wenn eine Amplitudenhüllkurve verwendet wird, ist dieser Wert gewöhnlich immer null.

"ENTER DURATION (Lange eingeben) Dies ist eine beliebige Nummer zwischen -32768 und 32767. Wird ein positiver Wert eingegeben, dann wird sie als absolute Lange der Note in 100stel Sekunden betrachtet. Wenn ein negativer Wert eingegeben, so wird sie darstellen, wie oft eine Amplitudenhullkurve wiederholt werden soll.

Diese Anweisung is der "SOUND"-Anweisung in jeder Hinsicht ähnlich. Wenn die Warteschlange jedoch belegt ist, ... wird der Programmzähler auf die laufende Anweisung zeigen (die nicht ausgeführt werden kann, da die Warteschlange belegt ist) und die Kontrolle wird zum PLAY-Befehl zurückkehren. Die Parameter werden wie bei der "SOUND"-Anweisung eingegeben. "WAIT-SOUND" wird bei Melodien immer verwendet.

"RESET" (Zurücksetzen)

Diese Anweisung hat keine Parameter und erzeugt daher auch keine Aufforderungen. "RESET" bewirkt die Löschung von allen Tonwarteschlangen und bricht alle zur Zeit ausgeführten Töne ab. Sie ist gewöhnlich die erste Anweisung, die von Melodien ausgeführt wird.

"RELEASE" (Auslösen)

Damit können einzelne Töne ausgelöst werden. Es gibt drei Aufforderungen:

"CHANNEL A (Y/N)" (Kanal A -J/N-) Falls Sie den in Kanal A gespeicherten Ton auslösen wollen müssen Sie "Y" tippen, ansonsten "N".

"CHANNEL B (Y/N)" (Kanal B -J/N-) Falls Sie den in Kanal B gespeicherten Ton und/statt den in Kanal A gespeicherten Ton auslösen wollen müssen Sie "Y" tippen, ansonsten "N".

"CHANNEL C" (Y/N) (Kanal C (J/N)) Falls Sie den in Kanal C gespeicherten Ton und/statt den in Kanal A und/oder Kanal B gespeicherten Ton auslosen wollen müssen Sie "Y" tippen, ansonsten "N".

"HOLD" (Anhalten)

Hold hat keine Parameter und somit auch keine Aufforderungen. Damit werden alle Töne angehalten ohne die Warteschlange zu leeren, die daher durch 'CONTINUE' neu gestartet werden können.

"CONTINUE" (fortsetzen)

Auch diese Anweisung hat keine Parameter und erzeugt keine Meldungen. Durch diese Anweisung werden sämtliche durch "HOLD" festgehaltene Töne wieder freigegeben.

"AMP-ENV"

Diese Anweisung hat eine unterschiedliche Anzahl von Parametern (abhängig von der Anzahl der Hüllkurvenabschnitte). Hullkurven können Hardware oder Software sein (s. BASIC-Handbuch). Folgende Meldungen werden erzeugt:

"ENTER AMPLITUDE ENVELOPE NUMBER" (Nummer der Amplitudenhüllkurve eingeben) Eine Nummer zwischen 1 und 15, welche die zu definierende Hüllkurve bestimmt.

"NUMBER OF SECTIONS" (Zahl der Abschnitte) Eine Nummer zwischen 1 und 15, welche die Zahl der einzugebenden Abschnitte bestimmt. Jeder Abschnitt wird wie folgt angezeigt:

"SOFTWARE ENVELOPE (Y/N)" (Softwarehullkurve (J/N)) Wenn "Y" eingetippt wird, werden die nachsten drei Meldungen erscheinen. Wenn "N" eingetippt wird, werden stattdessen die nachsten zwei Meldungen erscheinen.

"ENTER STEP COUNT" (Abstufungszahl eingeben) Dies ist eine Nummer zwischen 1 und 127 (s. BASIC Handbuch)

und -128 (s. BASIC-[illegible]dbuch)

"ENTER PAUSE TIME" (Pausenlänge eingeben) Dies ist eine Nummer zwischen 0 un[illegible] 255 (s. BASIC-Handbuch).

oder wenn eine Hardwarehüllkurve gewählt wird:

"ENTER ENVELOPE SHAPE" (Hüllkurvenform eingeben) Dies ist eine Nummer zwischen 8 und 15 und bestimmt eine von acht möglichen Hardwarehüllkurven (s. BASIC-Handbuch).

"ENTER ENVELOPE PERIOD" (Hüllkurvenform eingeben) Dies ist eine Nummer zwischen 0 und 65535 (s. BASIC-Handbuch).

"TONE-ENV"

Diese Anweisung ist der "AMP-ENV"-Anweisung ähnlich, da sie ebenfalls eine unterschiedliche Anzahl von Parametern hat (abhängig von der Anzahl der Hüllkurvenabschnitte). Es werden folgende Meldungen erzeugt:

"ENTER TONE ENVELOPE NUMBER" (Nummer der Amplitudenhüllkurve eingeben) Eine Nummer zwischen 1 und 15, welche die zu definierende Hüllkurve bestimmt

"NUMBER OF SECTIONS" (Zahl der Abschnitte) Dies sollte eine Nummer zwischen −5 und 5 sein (ausschließl. 0), welche die Zahl der einzugebenden Abschnitte bestimmt. Ein negativer Wert entspricht einer Wiederholungshüllkurve. Jeder Abschnitt wird wie folgt angezeigt:

"ENTER STEP COUNT" (Abstufungszahl eingeben) Eine Nummer zwischen 0 und 239 (s. BASIC-Handbuch).

"ENTER STEP SIZE" (Abstufungsgröße eingeben) Dies ist eine Nummer zwischen 127 und -128 (s. BASIC-Handbuch)

"ENTER PAUSE TIME" (Pausenlänge eingeben) Dies ist eine Nummer zwischen 0 und 255 (s. BASIC-Handbuch).

"RE-RUN" (wiederholter Ablauf)

Hierbei kehrt das Programm zur ersten Anweisung zurück (gewöhnlich eine 'SOUND' (Ton)- oder 'WAIT-SOUND' (Tonverzögerung)-Anweisung). Bei Melodien wird diese Anweisung selten verwendet, da die ersten Anweisungen gewöhnlich nur einmal ausgeführt werden. Ein Toneffekt wird üblicherweise mit einem 'STOP' gefolgt von einem 'RE-RUN' beendet. 'RE-RUN' hat keine Parameter.

"JUMP" (Springen)

Diese Anweisung ist flexibler als 'RE-RUN', da die Kontrolle auf jeden beliebigen Programmteil übertragen werden kann. Es wird nur eine Meldung ausgegeben:

"ENTER PC VALUE" (PC-Wert eingeben) Eine Nummer zwischen 1 und 65[illegible] en PC-Wert zu errechnen, ist es oft notwendig, einen Hilfswert, wie z.B. 1, einzugeben. Nach Eingabe des ganzen Programms 'LIST' (Laufen lassen) (Wahlfunktion 5) eintippen und den PC-Wert beachten. Danach kann der echte Wert eingegeben werden. Hierfür verwendet man 'DOKE VALUE' (Wahlfunktion 15) oder 'OVERWRITE AT PC' (Wahlfunktion 2). Melodien werden gewöhnlich durch einen Sprung zum ersten 'CALL' (Aufruf) des Programms abgebrochen, um die Melodie zu wiederholen.

"RE-LIM" (Neue Begrenzung bilden)

Der PLAY-Befehl (Spielen) wird meistens unter Interrupt ausgeführt (s. ISET, IRUN, IEND). Eine Reihe von anderen Befehlen werden recht häufig unter Interrupt ausgeführt. Sollte es sich dabei um Grafik-Befehle handeln, dann werden dies viel häufiger benutzt als der PLAY-Befehl. Der PLAY-Befehl hat eine Begrenzung, um die Prozessorzeit zu verkürzen, wodurch Ausführungen mit bestimmten Interrupts möglich sind. Wenn die Grenze auf 0 eingestellt ist, erfolgt bei jedem Aufruf eine Ausführung, bei 1 erfolgt bei jedem zweiten Aufruf eine Ausführung, bei 2 erfolgt bei jedem dritten Aufruf eine Ausführung, usw. Mit 'RE-LIM' können Sie diese Grenze vom Programm aus ändern und auch den internen Zähler des Programms zurücksetzen. Wenn Sie 'count' (Zahlwert) und 'limit' (Grenze) gleichsetzen, dann wird das Programm beim nächsten Aufruf ausgeführt. Ansonsten wird es auf 'count' gleich 'limit' warten, bevor es zur Ausführung gelangt. Nach jeder Ausführung wird 'count' automatisch auf 0 zurückgesetzt. Diese Anweisung hat daher zwei Parameter, die mit einer Aufforderung eingegeben werden:

"ENTER COUNT,LIMIT" Beide Nummern sollten zwischen 0 und 255 liegen. Sie müssen mit einem Komma getrennt eingetippt werden, bevor ENTER gedrückt wird.

"CALL-CHANNEL" (Kanalaufruf)

Diese Anweisung ermöglicht am Programmanfang die änderung des Kanalstatus im 'SOUND-bzw. 'WAIT-SOUND-Datenblock, ohne 'SOUND' oder WAIT-SOUND zur Ausführung zu bringen. Gewöhnlich wird sie zur änderung der Rendezvous-Anforderungen verwendet, bevor der Wartes-chlange ein bestimmter Ton angereiht wird. Zur Erzeugung einer Melodie, die mehr als einen Kanal benötigt, und daher auch mehr als ein Programm (und mehr als ein Sprite!), und um all Kanäle gleichzeitig zu starten, setzt man gewöhnlich in jedem Programm den ersten Ton so ein, daß er sich mit den beiden anderen trifft. Die drei Harmonien laufen aber selten gemeinsam ab, nachdem der erste Ton ausgeführt wurde. Die Rendezvous werden meistens nach dem ersten 'call-TONE-PERIOD' (Aufruf Tonperiode) oder 'CALL-TONE-PERIOD' (Aufruf Tonlänge) zurückgesetzt, damit die drei Kanäle unabhängig voneinander spielen können. Diese Technik wird in den Beispielmelodien dieses Pakets demonstriert. Obwohl diese Anweisung nur ein Datenbyte hat, ist jedes einzelne Bit sehr wichtig. Es gibt hier insgesamt 8 Aufforderungen, eine für jedes Bit. Diese Aufforderungen wurden bereits bei den 'SOUND -Anweisungen ausführlich beschrieben und sind nachfolgend zusammengefaßt.

CHANNEL A (Y/N)
CHANNEL B (Y/N)
CHANNEL C (Y/N)
RENDEZVOUS WITH A (Y/N)
RENDEZVOUS WITH B (Y/N)
RENDEZVOUS WITH C (Y/N)
HOLD (Y/N)
FLUSH (Y/N)

"CALL-AMP-ENV" (Aufruf Amplitudenhüllkurve)

Anweisung zur Ausführung zu bringen. Es wird nur eine Aufforderung ausgegeben, welche für die Hüllkurve bestimmt ist – siehe 'SOUND'/'WAIT-SOUND'.

"CALL-TONE-ENV" (Aufruf Tonhüllkurve)

Abgesehen von der änderung der Tonhüllkurvennummer, ist diese Anweisung identisch mit 'CALL-AMP-ENV' – siehe 'SOUND'/'WAIT-SOUND'.

"CALL-TONE-PERIOD" (Aufruf Tonperiode)

Diese Anweisung ermöglicht am Programmanfang die Änderung der Tonperiode (welche die Tonhöhe steuert) im 'SOUND'- bzw. 'WAIT-SOUND' - Datenblock. Diese Anweisung wird 'SOUND' oder 'WAIT-SOUND' zur Ausführung bringen. Sollte die erste Anweisung 'WAIT-SOUND' und nicht 'SOUND' sein, und sollte die Warteschlange besetzt sein, dann verbleibt der Programmzähler bei CALL-TONE-PERIOD und die Kontrolle kehrt vom PLAY-Befehl zurück. Diese Anweisung und 'CALL-TONE-DURATION' werden in der Praxis benutzt, um der Warteschlange Töne anzureihen, während 'SOUND' und 'WAIT-SOUND' nur selten direkt ausgeführt werden. Es wird lediglich ein Parameter gefordert und zwar für "TONE-PERIOD", die eine Nummer von 0 bis 4095 haben kann.

"CALL-NOISE-PERIOD" (Aufruf Schallperiode)

Diese Anweisung ermöglicht dem Benutzer am Programmanfang die Schallperiode im 'SOUND' – bzw. 'WAIT-SOUND'-Datenblock zu ändern, ohne daß 'SOUND' bzw. 'WAIT-SOUND' ausgeführt werden. Es wird nur eine Aufforderung ausgegeben – siehe 'SOUND'/'WAIT-SOUND'.

"CALL-INITIAL-AMP" (Aufruf Anfangsamplitude)

Diese Anweisung ermöglicht dem Benutzer am Programmanfang die Anfangsmplitude im 'SOUND'- bzw. 'WAIT-SOUND'- Datenblock zu ändern, ohne daß 'SOUND' bzw. 'WAIT-SOUND' zur Ausführung gelangen. Es wird nur eine Aufforderung ausgegeben, welche für die Anfangsamplitude bestimmt ist – siehe 'SOUND'/'WAIT-SOUND'.

"CALL-DURATION" (Aufruf Länge)

Diese Anweisung ermöglicht dem Benutzer am Programmanfang die Länge im 'SOUND'-bzw., 'WAIT-SOUND'-Datenblock zu ändern, ohne daß 'SOUND' bzw. 'WAIT-SOUND' ausgeführt werden – siehe 'SOUND'/WAIT-SOUND'.

"CALL-TONE-DURATION" (Aufruf Tonlänge)

Diese Anweisung ermöglicht dem Benutzer die Tonperiode sowie die Tonlänge in einer Anweisung zu ändern, wodurch sich ein Byte (und Programmzeit) einsparen läßt. Wenn beide Werte gleichzeitig geändert werden, was sehr oft vorkommt. Wie bei 'CALL-TONE-PERIOD' werden die neuen Werte in den Datenblock gesetzt und 'SOUND' oder 'WAIT-SOUND' werden anschließend ausgeführt. Im letzteren Fall wird bei einer belegten Warteschlange der PLAY-Befehl verlassen, wobei der Programmzähler weiterhin bei der Anweisung 'CALL-TONE-DURATION' bleibt. Es gibt zwei Aufforderungen, die erste für die Länge und die zweite für die Tonperiode – siehe 'SOUND'/'WAIT-SOUND'.

"STOP"

Hiermit wird die Ausführung des Programms beendet und der 'PLAY'-Befehl wird verlassen. Der Programmzähler wird die nächste Anweisung nach 'STOP' anzeigen. Es gibt hier keine Parameter.

"DONE" (Fertig)

Dies erzeugt im Tonprogramm keinen Code, sondern beendet ganz einfach die Programmeingabe und kehrt zum Menü zurück

Diese ist der 'DONE'-Anweisung ähnlich. Es wird hier jedoch ... den 'LIST'-Befehl darauf hinweist, daß das Programmende erreicht wurde. Dieser Programmteil sollte bei der Ausführung des Programms nicht erreicht werden, da das System sonst mit "**ILLEGAL TRACKER CODE**" (Ungültiger Verfolgungscode) zum Stillstand kommt.

Allgemeines:

a) Falls das für die Melodieaufnahme von Ihnen geschaffene Sprite belegt ist, wird die Meldung "END OF SPACE" (Kein Platz mehr) ausgegeben. Versuchen Sie nicht, weitere Daten einzugeben. Die Daten werden bereits addiert sein.

b) Falls die von Ihnen getippten Daten nicht in das Sprite passen, dann wird die Meldung "NO ROOM" (Kein Platz) ausgegeben. Sollte dies der Fall sein, dann sind die Daten nicht addiert.

c) Wenn Sie irgendwelche Anweisungen falsch eingeben (ENTER), dann wird die Meldung "ILLEGAL INSTRUCTION" (ungültige Anweisung) erscheinen. Sie brauchen dann nur die Anweisung richtig einzugeben, da noch keine Daten in das Sprite geschrieben wurden.

d) Wenn Sie zufällig einen Parameter eingeben, der nicht aus dem gültigen Bereich stammt, dann wird die Meldung "OUT OF RANGE" (Bereichsabweichung) erscheinen und Sie werden aufgefordert, die Daten neu einzugeben.

e) Wenn Sie andere Aufgaben unter Interrupt laufen lassen, werden Sie vielleicht feststellen, daß Ihre Melodie oder Ihr Toneffekt langsamer laufen. Man kann die für die Regelung notwendige Länge nur durch Versuch feststellen. Tonhöhen (durch die Tonperioden bestimmt) bleiben unverändert.

Wahlfunktion 2 – OVERWRITE AT PC (Überschreiben bei PC)

Wenn Sie Ihr Programm auflisten ('LIST'), werden Sie manchmal feststellen, daß Sie das Programm falsch eingegeben haben oder eine Anweisung ausgelassen haben (siehe Wahlfunktion 4 – INSERT AT PC (Einsetzen bei PC)). In diesem Fall können Sie Ihren Fehler mit dieser Wahlfunktion überschreiben. Beachten Sie, daß der Aussprung aus dieser Wahlfunktion mit einem 'DONE' und nicht mit einem 'EXIT' erfolgt, was in Ihrem Programm eine Markierung hinterlassen wird (es sei denn, die letzte Anweisung wird überschrieben). Mit dieser Funktion lassen sich auch Anweisungen am Ende des zuvor eingegebenen Programms hinzufugen. Diese Wahlfunktion wird folgende Aufforderungen erzeugen:

"ENTER TARGET SPRITE" Dies ist eine Nummer zwischen 1 und 255 und muß ein vordefiniertes Sprite sein. Ist das Sprite nicht vorhanden, dann erscheint die Meldung "NO SUCH SPRITE" (Sprite nicht vorhanden) und die Kontrolle wird zum Menü zurückkehren. Ist das Sprite vorhanden, erscheint folgende Aufforderung:

"ENTER PC VALUE" Dies sollte ein Wert zwischen 1 und 65535 sein. "PC-Wert eingeben. Ist der Wert höher, als die Spriteabmessungenzulassen, dann wird eine Meldung "NO ROOM" (kein Platz) erzeugt, sobald Sie versuchen, eine Anweisung einzugeben.

Befindet sich der PC-Wert innerhalb des Bereichs, dann werden die Anweisungen in der gleichen Reihenfolge wie bei Wahlfunktion 1 eingeben.Siehe 'EXIT' und 'DONE'. Zur Feststellung des PC-Wertes in einem bestimmten Programmteil siehe Wahlfunktion 5 – LIST.

Wahlfunktion 3 – DELETE AT PC (Loschen bei PC)

Diese Wahlfunktion ermoglicht das Loschen eines ganzen Programmteils und das Speichern des restlichen Programms im geloschten Speicherteil. Das ganze Programm wird daher um den geloschten Teil verkurzt. Diese Wahlfunktion wird drei Aufforderungen erzeugen

– OVERWRITE AT PC (Überschreiben bei PC).

"ENTER PC VALUE" — Diese Aufforderung sollte ebenfalls wie bei der vorhin beschriebenen Wahlfunktion 2 – OVERWRITE AT PC – behandelt werden.

"HOW MANY BYTES" — Dies sollte eine Nummer zwischen 1 und 65535 sein und ist die Anzahl der Bytes, die vom PC-Wert (einschließlich des PC-Wertes) gelöscht werden soll.

Die Ausführung dieser Wahlfunktion kann einige Sekunden in Anspruch nehmen. Der entsprechende PC-Wert wird durch

Wahlfunktion 5 ermittelt – LIST.

Wahlfunktion 4 INSERT AT PC (Einsetzen bei PC)

Mit dieser Wahlfunktion kann der Benutzer für ein Programmteil Platz schaffen, indem das restliche Programm (einschließlich Anweisungen beim ausgewählten PC) im Speicher nach oben bewegt wird, um Platz zu machen. Kontrollieren Sie, ob genügend Platz vorhanden ist, da sonst das Ende Ihres Programms verloren gehen kann. Diese Wahlfunktion erzeugt die gleichen drei Aufforderungen wie die Wahlfunktion 3 – DELETE AT PC. Diese sollten in der gleichen Weise behandelt werden. Der zugeteilte Raum wird zunächst mit Nullzeichen gefüllt.

Wahlfunktion 5 – LIST TUNE (Melodie auflisten)

Diese Wahlfunktion ermöglicht das Auflisten (LIST) eines vorher eingegebenen Tonprogramms am Bildschirm bzw. am Drucker. Am Ende dieses Abschnittes ist eine Beispielauflistung vorgegeben. Diese Wahlfunktion erzeugt vier Aufforderungen:

"PRINTER (Y/N)" — (Drucker -J/N-) Wenn"Y" eingetippt wird, dann erfolgt die Ausgabe nur zum Drucker, wird "N" eingetippt, dann wird die Ausgabe zum Bildschirm übertragen:

"ENTER TARGET SPRITE" — Die Nummer des Sprites, welches das zu listende Programm enthält (siehe Wahlfunktion 2).

"ENTER PC VALUE" — Jener PC-Wert, bei dem die Auflistung beginnt (siehe Wahlfunktion 2).

"NUMBER OF BYTES" — Die Länge des aufzulistenden Codes. Ist dieser länger als das eigentliche Sprite, dann wird die Auflistung abgebrochen. Wird eine Markierung festgestellt (siehe 'DONE'), dann wird die Auflistung ebenfalls abgebrochen. Wird eine Anweisung bei einem ungültigen Code entdecket, dann wird 'JLLEGAL INSTRUCTION' (ungultige Anweisung) ausgegeben und die Auflistung wird beim nächsten Byte fortgesetzt. Das Auflisten kann wie bei normalen BASIC-Auflistungen gestoppt und mit "ESC" wieder fortgesetzt werden. Auflistungen werden im folgenden Format erzeugt:

Spalte 1

Dies ist die absolute Adresse (in Hexadezimalform) der laufenden Anweisung. Sie wird sich nur ändern, wenn die Sprites gelöscht, verschoben oder gemischt werden. Denken Sie aber immer daren, daß sie sich ändern kann. Machen Sie jedesmal eine Kontrolle, wenn Sie sich unsicher sind.

Dies ist das eigentliche Byte, das in der laufenden Adresse enthalten ist (in Dezimalform).

Spalte 3

Dies ist der Programmzählwert (in Dezimalform), welcher bei den meisten Wahlfunktionen die Position im Programm angibt.

Spalte 4

Enthält entweder die laufende Anweisung oder die laufenden Daten, die zur laufenden Anweisung gehören.

Wahlfunktion 6 – SAVE TUNES (Melodien speichern)

Diese Wahlfunktion wurde entwickelt, um laufende Sprites auf Band bzw. auf Diskette zu speichern. Es wird nur eine Aufforderung ausgegeben:

"ENTER FILENAME TO SAVE UNDER" — (Dateiname zum Speicherneingeben) Der Dateiname darf höchstens 8 Zeichen lang sein. Die letzten drei Zeichen müssen "SPR" sein, um eine Sprite anzugeben.

Wahlfunktion 7 – LOAD TUNES (Melodien laden)

Diese Wahlfunktion wurde geschaffen, um das Laden einer bereits gespeicherten Spritesdatei von Band bzw. Diskette zu ermöglichen. Wenn bei dieser Wahlfunktion (oder bei einer anderen) ein Fehler auftritt, dann brauchen Sie nur "RUN" einzutippen. Es wird nur eine Aufforderung erzeugt:

"ENTER FILE TO LOAD" — (Die zu speichernde Datei eingeben) Der Dateiname muß auch hier dem Spritedateiformat entsprechen.

ANMERKUNG:

Zum Laden der Demonstrationsmelodien muß der Dateiname "MUSICSPR" verwendet werden. (Diese sind direkt nach dem Tonerzeugerprogramm gespeichert).

Wahlfunktion 8 – MERGE TUNES (Melodien mischen)

Diese Wahlfunktion wurde geschaffen, um eine auf Band bzw. Diskette gespeicherte Spritesdatei zu mischen. Auch hier gibt es nur eine Aufforderung, die das zu mischende Sprite verlangt.

Wahlfunktion 9 – PLAY TUNES (Melodien spielen)

Mit dieser Wahlfunktion kann der Benutzer seine "Arbeit" anhören. Da es hier 3 Kanäle gibt, köennen bis zu drei Programme gleichzeitig laufen. Die Rendezvous-Anforderungen machen es öfters notwendig, daß alle drei Melodien gemeinsam gespielt werden. Diese Wahlfunktion wird die Melodien unter Interrupt spielen. Es gibt mehrere Aufforderungen:

"HOW MANY TUNES" — (Wieviele Melodien) Dies ist eine Nummer zwischen 1 und 3, die bestimmt, wieviele Melodien gemeinsam gespielt werden. Bei den 3 Beispielprogrammen wird die Melodie in drei Sprites gehalten, eines fur jeden Kanal. Wegen der Rendezvous-Anforderungen wird keine alleine spielen. Fur jedes Sprite gibt es 3 Aufforderungen

"ENTER SPRITE NUMBER" — (Spritenummer eingeben) Dies ist eine Nummer zwischen 1 und 255, welche jenes Sprite angibt, das das zu spielende Programm enthält.

Sprite beginnen soll.

"ENTER LIMIT" (Grenze eingeben) ... mit wird die Frequenz der gewünschte Ausführung gesteuert.

Die BASIC-Zeilen, die die Melodie beginnen, könnten auch für Ihr Programm recht nützlich sein. Si sind auf 2960 bis 2990.

Spielen der Beispielmelodien

Es gibt 3 Beispielmelodien, die unter dem Dateinamen "MUSICSPR" Direkt nach dem Tonerzeuger programm auf Band bzw. Diskette aufgenommen wurden. Laden Sie die Beispielmelodien unte Anwendung der Wahlfunktion 7.

Melodie 1 Dies ist die Begleitmusik des "CHASE"-Bildschirm vom Laser-BASIC-Demo. Sie spie auf allen drei Kanälen und die Daten werden in Sprite 90, 91 und 92 gespeichert. Alle drei werden mit einem "PC to start at" von 11 und einem "Limit" von 1 eingeleitet

Melodie 2 Dies ist jene Musik, die den "Plattform-Spiel-Bildschirm" des Laser-BASIC-Demos begleitet. Auch sie wird auf allen drei Kanälen abgespiet, verwendet aber Sprite 60, 6 und 62. Wiederum werden alle drei mit einem "PC to start at" von 11 und einem "Limit" von 5 eingeleitet.

Melodie 3 Dies ist die Begleitmusik des "Glöckner"-Bildschirms vom Laser-BASIC-Demo. Auch hier werden alle drei Kanäle benutzt und Daten sind in Sprite 93, 94 und 95. Wie bei der Melodien 1 und 2 ist der "PC to start at" ein Wert von 11 und das "Limit" ist 5 oder weniger.

Wahlfunktion 10 – STOP TUNE (Melodie stoppen)

Hierdurch wird die unter Interrupt laufende Melodie bzw. der Toneffekt beendet und das Programm kehrt zum Menü zurück. Es werden keine Aufforderungen ausgegeben. Melodien können auch durch "ESC" angehalten werden.

Wahlfunktion 11 – ALTER SPRITE MAX (Spritehöchstwert ändern).

Der Spritehöchstwert wird bei 'RUN' (Lauf) vom Tonerzeugerprogramm nicht geändert. Die zu diesem Zeitpunkt im Speicher enthaltenen Sprites bleiben erhalten. Wenn Sie jedoch der Spritehöchstwert ändern wollen, dann konnen Sie Wahlfunktion 11 fur diesen Zweck verwenden. Es wird nur eine Aufforderung ausgegeben, die den neuen Spritehöchstwert verlangt Denken Sie daran, daß alle Sprites mit einer höheren Nummer als der Spritehöchstwert verloren gehen.

Wahlfunktion 12 – CRUNCH TUNE (Melodie komprimieren)

Diese Wahlfunktion stellt eine sehr nützliche Möglichkeit dar. Sie muß aber mit Vorsicht verwendet werden. Wie vorhin beschrieben, halten wir es für sinnvoller, wenn Sprites etwas größer als erforderlich entworfen werden, da die gewünschte Größe immer schwer abzuschatzen ist und sich Fehler nicht vermeiden lassen. Daten können aus einem zu kleinen Sprite nur nach Aussprung aus dem Programm von Hand ubertragen werden, d h Sie mussen mit ISPR die Anfangsadresse finden ein neues Sprite erzeugen und mittels PEEKs und POKEs die Ubertragung ausfuhren – dies ist nicht zu empfehlen. Das Tonerzeugerprogramm hinterläßt genugend Speicherplatz für die Sprites . Es ist nur selten der Fall, daß für Sprites nicht genugend Platz vorhanden ist, es sei denn Sie mischen eine Datei mit Grafiksprites. Achten Sie darauf, daß durch das Komprimieren ein neues Sprite mit der gewünschten Größe erzeugt wird, bevor das alte Sprite geloscht wird Es muß ebenfalls mindestens eine Spritenummer frei sein Wenn der Speicherraum fur die Ubertragung unzureichend ist, dann wird "INSUFFICIENT ROOM" (Nicht genugend Platz) gemeldet. Wenn alle Spritenummern verteilt wurden, wird "NO FREE SPRITE" (Kein freies Sprite) angezeigt. Eine Ubertragung findet nicht statt, wenn die erste freie Spritenummer großer als der maximale Spritewert ist. Diese Operation erzeugt mehrere Aufforderungen:

"ENTER PC VALU" Dies ist der Wert des Programmzählers beim letzten vom Tonprogramm benutzten Byte. Vorsicht: die letzte Anweisung kann 2 oder mehr Bytes lang sein. Falls Sie sich nicht sicher sind, können Sie vorsichtshalber ein paar Bytes hinzurechnen. Bei jedem Beispielprogramm ist die letzte Anweisung eine 'JUMP'-Anweisung (3-Byte-Anweisung). Wir benötigen den Programmzählerwert (Spalte 3 der Auflistung) bei der 'JUMP'-Anweisung selbst + 2. Hiermit wird das letzte Byte angezeigt.

Nach erfolgreichem 'Komprimiervorgang' wird die Höhe und die neue Breite angezeigt. Durch Drücken einer beliebigen Taste können Sie zum Menü zurückkehren. Die Ausführung dieser Wahlfunktion kann einige Zeit in Anspruch nehmen.

Wahlfunktion 13 – CLEAR ALL TUNES (Alle Melodien löschen)

Diese Wahlfunktion ermöglicht das Löschen aller im Speicher vorhandenen Sprites sowie das Neusetzen der maximalen Spritenummer. Es gibt zwei Aufforderungen:

"CLEAR SPRITES (Y/N)" (Sprites löschen -J/N-)Diese Aufforderung wurde geschaffen, falls Sie die Wahlfunktion 13 aus Versehen gewählt haben (dies ist für manche ungünstig). Wenn dies der Fall ist, brauchen Sie nur "N" eintippen, um zum Menu zurückzukehren. Tippen Sie "Y", um das Löschen fortzusetzen. Danach wird folgende Aufforderung ausgegeben:

"ENTER NEW SMAX" (Neuen Spritehöchstwert eingeben) Hier sollten Sie den neuen Spritehöchstwert eingeben, der eine Nummer zwischen 1 und 255 ist.

Wahlfunktion 14 – POKE BYTE

Diese Wahlfunktion ist mit Vorsicht zu benutzen. Sie ist im Prinzip eine Alternative zur Wahlfunktion 2 – OVERWRITE AT PC: Die Wahlfunktion 'LIST' kann zum Auffinden der absoluten Adresse (Spalte 1), die Sie "POKE"en wollen, benutzt werden. Kontrollieren Sie, ob sich diese Adresse nicht geändert hat, seit sie zum letzten Mal überprüft wurde. Es gibt hier zwei Aufforderungen:

"ENTER ADRESS TO POKE" (Adresse zum POKE eingeben) Dies ist eine Nummer (hexadezimal) zwischen 1 und 7000. Denken Sie daran, daß die in Spalte 1 der Auflistung gegebene Adresse eine HEX-Adresse ist und somit ein "&"-Zeichen vorgesetzt werden muß, wenn Sie als HEX-Nummer eingegeben werden soll.

"ENTER BYTE" (Byte eingeben) Dies ist der Wert, den Sie in die Adresse POKEn wollen. Es wäre sinnvoll, das Programm jedesmal nach der Ausführung dieser Wahlfunktion aufzulisten, um sicher zu gehen, daß Sie keinen Fehler gemacht haben. Das Byte muß zwischen 0 und 255 liegen.

Wahlfunktion 15 – DOKE VALUE

Diese komisch klingende Wahlfunktion ist das 16-Bit-Gegenstuck von 'POKE BYTE'. Es gibt hier zwei Aufforderungen:

"ENTER ADDRESS TO DOKE" (Adresse zum DOKE eingeben) Dies ist wiederum die Adresse, die geandert werden muß. Das unbedeutendste Byte kommt in diese Position und das bedeutendste Byte kommt in die nachste Position.

... gegebene Adresse geDOKET wird. Er kann zwischen -32768 und 65535 liegen.

Wahlfunktion 16 – DELETE TUNE (Melodie löschen)

Mit dieser Wahlfunktion kann ganz einfach ein einzelnes Sprite gelöscht werden. Es werden nur zwei Aufforderungen ausgegeben:

"SPRITE TO DELETE"	(Das zu löschende Sprite eingeben) Dies sollte die Nummer des zu löschenden Sprites sein, die zwischen 1 und 255 liegen muß, oder der maximale Spritewert.
"DELETE (Y/N)"	(Löschen -J/N-) Falls Sie die falsche Spritenummer eingegeben haben, können Sie hiermit abbrechen. Tippen Sie einfach "N", um abzubrechen, und "Y", um das Löschen fortzusetzen.

Verwendung von Ton in Ihren Programmen

Nachdem Sie Ihre Melodien/Toneffekte erzeugt haben, ist es relativ einfach, diese in Ihr BASIC-Programm einzubauen. Für die Verwendung von Ton gibt es nämlich nur einen Befehl. Dieser hat jedoch zwei unterschiedliche Operationen, die von der Anzahl der gebotenen Parameter abhängig sind.

PLAY,e1,e2

Hat der PLAY-Befehl 2 Parameter, dann geschieht folgendes: Beim Sprite, dessen Nummer sich in der Variable KEY befindet, wird der Programmzähler auf e1 gesetzt, der Zählwert auf 0 und die Grenze auf e2. Im Tonprogramm werden keine Anweisungen ausgeführt.

PLAY

Wenn der PLAY-Befehl keine Parameter hat, dann wird das Programm in dem Sprite, dessen Nummer sich in KEY befindet, vom laufenden Programmzähler aus aufgeführt.

Beispiel

Angenommen, wir haben eine Melodie, die in 3 Sprites mit den Nummern 50, 51 und 58 enthalten ist. Angenommen, jede wird mit einem Anfangs-PC von 11 (d.h. am Anfang des Programms befindet sich eine 'SOUND'- bzw. 'WAIT-SOUND'-Anweisung) und einer Grenze von 5 ausgeführt. Um Sie unter Interrupt mittels SETs 1, 2 und 4 zum Laufen zu bringen würden wir folgendes verwenden:

```
10 |SET,1:|KEY,50:|PLAY,11,5:A$="PLAYB"
20 |SET,2:|KEY,51:|PLAY,11,5:A$=A$+"PLAYC"
30 |SET,4:|KEY,58:|PLAY,11,5:A$=A$+"PLAYE#"
40 |ISET,@A$:|IRUN,5
```

Allgemeines zur Verwendung von Toneffekten

1. Der 'RUN'-Befehl verursacht eine Rücksetzung ('RESET') des Tonchips. Wenn Sie also eine Melodie unter Interrupt spielen, dann wird ein Programm unberechenbare Resultate erbringen.

2. Wenn Sie eine Melodie spielen, dann sollte die erste Anweisung praktisch immer 'WAIT-SOUND' sein.

3. Wenn Sie eine Melodie spielen, die mehr als einen Kanal beansprucht, dann ist es besser, wenn die einzelnen Melodieteile in separaten Sprites gaschrieben werden. (Dies verhindert ununterbrochene CALL-CHANNELs (Kanalaufrufe))

4. Wenn Sie eine Melodie in mehr als einem Sprite ausgeführt wird (im Gegensatz zur ersten Anweisung im Programm) ein 'RESET' sein. Die ... wird gewöhnlich von sämtlichen Hüllkurvendefinitionen und anschließend einer 'CALL-CHANNEL'-Anweisung (in jedem Sprite) gefolgt, um sicher zu gehen, daß die Melodien gleichzeitig starten.

5. Wenn es sich um 3 Sprites handelt, dann sollte die erste Note in Kanal A mit B und C zusammentreffen, die erste Note in Kanal B mit A und C, und die erste Note in Kanal C mit A und B. Nach Ausführung der ersten 'SOUND'- bzw. 'WAIT-SOUND'-Anweisung ändern sich gewöhnlich die Rendezvous-Anforderungen.

6. Fängt eine Melodie nicht auf allen drei Kanälen gleichzeitig an, dann wird gewöhnlich eine Blindhüllkurve mit Lautstärke 0 verwendet, um eine Verzögerung hervorzurufen, bis alle drei am Anfang zusammentreffen.

7. Wenn eine Melodie dauernd gespielt wird, dann lautet die letzte Anweisung gewöhnlich 'JUMP' (Spring) zurück und 'CALL-CHANNEL', die die Rendezvous mit den anderen Harmonien herbeiführt. Die 'RESET' (Rücksetz-Anweisung) muß nicht noch einmal ausgeführt werden. Das gleiche gilt für die Hüllkurvendefinitionen.

Eine typische 3-Kanal-Melodie kann folgendes Format haben:%

Sprite 1

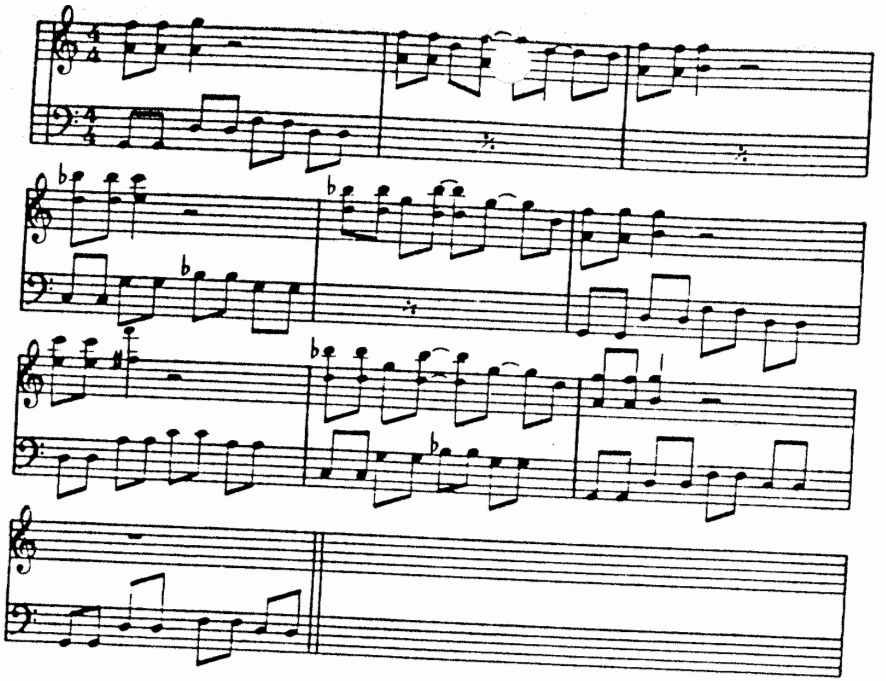
PC	Anweisung	
1	WAIT-SOUND	
11	RESET	
12	AMP-ENV	
30	TONE-ENV	
38	CALL-CHANNEL	(Rendezvous mit B und C einstellen)
40	CALL-TONE-DURATION	(erste Note spielen)
45	CALL-CHANNEL	(Rendezvous-Anforderungen löschen)
47	CALL-TONE-DURATION	(die restliche Melodie)
.		
.		
.		
N	JUMP	('JUMP' zurück auf 'CALL-CHANNEL in 38)

Dieses würde mit PLAY,11,LIMIT aufgebaut werden.

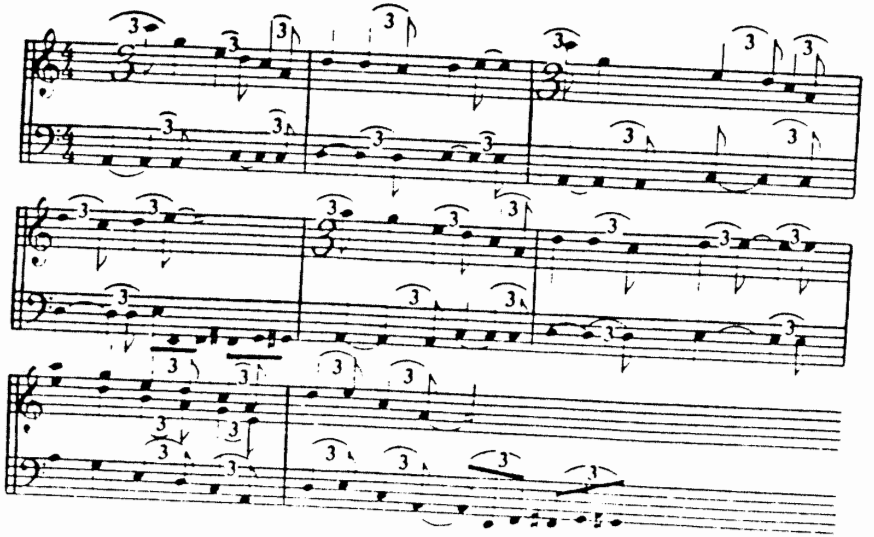
Sprites 2 und 3

Diese würden dieselbe Form annehmen wie Sprite 1, mit der Ausnahme, daß sie kein 'RESET' enthalten. Auch sie würden so aufgebaut werden, daß die Ausführung bei einem PC-Wert von 11 beginnt.

Blindamplitudenhüllkurven mit einer Lautstärke von 0 zur Kontrolle von verzögerten Anfängen würden vor einem 'CALL-CHANNEL' (dargestellt bei einem PC von 38 in Sprite 1) gesetzt werden und gewöhnlich in Sprite 1 definiert. Es gibt eigentlich keinen Grund, warum nicht sämtliche Hüllkurvendefinitionen in Sprite 1 aufgebaut werden könnte.



DER GLÖCKNER IN MUSIK




```
CHANNEL A

6C55 1     1    WAIT-SOUND
                ISSUE ON A
6C57 1     3    AMPLITUDE ENVELOPE= 1
6C58 0     4    TONE ENVELOPE     = 0
6C59 126   5    TONE PERIOD       = 638
6C5B 0     7    NOISE PERIOD      = 0
6C5C 15    8    INITIAL VOLUME    = 15
6C5D 23    9    DURATION          = 23
6C5F 2     11   RESET
6C60 6     12   AMP-ENV
6C61 1     13   AMPLITUDE ENVELOPE= 1
6C62 5     14   NO OF SECTIONS    = 5
6C63 1     15   STEP COUNT        = 1
6C64 0     16   STEP SIZE         = 0
6C65 1     17   PAUSE TIME        = 1
6C66 1     18   STEP COUNT        = 1
6C67 0     19   STEP SIZE         = 0
6C68 10    20   PAUSE TIME        = 10
6C69 1     21   STEP COUNT        = 1
6C6A 241   22   STEP SIZE         = 241
6C6B 1     23   PAUSE TIME        = 1
6C6C 12    24   STEP COUNT        = 12
6C6D 1     25   STEP SIZE         = 1
6C6E 1     26   PAUSE TIME        = 1
6C6F 12    27   STEP COUNT        = 12
6C70 255   28   STEP SIZE         = 255
6C71 1     29   PAUSE TIME        = 1
6C72 6     30   AMP-ENV
6C73 2     31   AMPLITUDE ENVELOPE= 2
6C74 5     32   NO OF SECTIONS    = 5
6C75 1     33   STEP COUNT        = 1
6C76 15    34   STEP SIZE         = 15
6C77 1     35   PAUSE TIME        = 1
6C78 1     36   STEP COUNT        = 1
6C79 0     37   STEP SIZE         = 0
6C7A 10    38   PAUSE TIME        = 10
6C7B 1     39   STEP COUNT        = 1
6C7C 241   40   STEP SIZE         = 241
6C7D 1     41   PAUSE TIME        = 1
6C7E 15    42   STEP COUNT        = 15
6C7F 1     43   STEP SIZE         = 1
6C80 1     44   PAUSE TIME        = 1
6C81 15    45   STEP COUNT        = 15
6C82 255   46   STEP SIZE         = 255
6C84 1     47   PAUSE TIME        = 1
6C84 11    48   CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6CB6 14    50   CALL-TONE-PERIOD
6CB7 126   51   TONE PERIOD       = 638
6CB9 14    53   CALL-TONE-PERIOD
6CBA 126   54   TONE PERIOD       = 638
6CBC 14    56   CALL-TONE-PERIOD
6CBD 83    57   TONE PERIOD       = 851
6CBF 11    59   CALL-CHANNEL
                ISSUE ON A
6C91 14    61   CALL-TONE-PERIOD
6C92 83    62   TONE PERIOD       = 851
6C94 14    64   CALL-TONE-PERIOD
6C95 204   65   TONE PERIOD       = 716
6C97 14    67   CALL-TONE-PERIOD
6C98 204   68   TONE PERIOD       = 716
6C9A 14    70   CALL-TONE-PERIOD
6C9B 83    71   TONE PERIOD       = 851
6C9D 14    73   CALL-TONE-PERIOD
6C9E 83    74   TONE PERIOD       = 851
6CA0 11    76   CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6CA2 14    78   CALL-TONE-PERIOD
6CA3 126   79   TONE PERIOD       = 638
6CA5 14    81   CALL-TONE-PERIOD
6CA6 126   82   TONE PERIOD       = 638
6CA8 14    84   CALL-TONE-PERIOD
6CA9 83    85   TONE PERIOD       = 851
6CAB 14    87   CALL-TONE-PERIOD
6CAC 83    88   TONE PERIOD       = 851
6CAE 11    90   CALL-CHANNEL
                ISSUE ON A
6CB0 14    92   CALL-TONE-PERIOD
6CB1 204   93   TONE PERIOD       = 716
6CB3 11    95   CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6CB5 14    97   CALL-TONE-PERIOD
6CB6 204   98   TONE PERIOD       = 716
6CB8 11    100  CALL-CHANNEL
                ISSUE ON A
6CBA 14    102  CALL-TONE-PERIOD
6CBB 83    103  TONE PERIOD       = 851
6CBD 11    105  CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6CBF 14    107  CALL-TONE-PERIOD
6CC0 83    108  TONE PERIOD       = 851
6CC2 14    110  CALL-TONE-PERIOD
6CC3 126   111  TONE PERIOD       = 638
6CC5 14    113  CALL-TONE-PERIOD
6CC6 126   114  TONE PERIOD       = 638
6CC8 14    116  CALL-TONE-PERIOD
6CC9 83    117  TONE PERIOD       = 851
6CCB 11    119  CALL-CHANNEL
                ISSUE ON A
6CCD 14    121  CALL-TONE-PERIOD
6CCE 83    122  TONE PERIOD       = 851
6CD0 14    124  CALL-TONE-PERIOD
6CD1 204   125  TONE PERIOD       = 716
6CD3 14    127  CALL-TONE-PERIOD
6CD4 204   128  TONE PERIOD       = 716
6CD6 14    130  CALL-TONE-PERIOD
6CD7 83    131  TONE PERIOD       = 851
6CD9 14    133  CALL-TONE-PERIOD
6CDA 83    134  TONE PERIOD       = 851
6CDC 11    136  CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6CDE 14    138  CALL-TONE-PERIOD
6CDF 188   139  TONE PERIOD       = 956
6CE1 14    141  CALL-TONE-PERIOD
6CE2 188   142  TONE PERIOD       = 956
6CE4 14    144  CALL-TONE-PERIOD
6CE5 126   145  TONE PERIOD       = 638
6CE7 11    147  CALL-CHANNEL
                ISSUE ON A
6CE9 14    149  CALL-TONE-PERIOD
6CEA 126   150  TONE PERIOD       = 638
6CEC 14    152  CALL-TONE-PERIOD
6CED 24    153  TONE PERIOD       = 536
6CEF 14    155  CALL-TONE-PERIOD
6CF0 24    156  TONE PERIOD       = 536
6CF2 14    158  CALL-TONE-PERIOD
6CF3 126   159  TONE PERIOD       = 638
6CF5 14    161  CALL-TONE-PERIOD
6CF6 126   162  TONE PERIOD       = 638
6CF8 11    164  CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6CFA 14    166  CALL-TONE-PERIOD
6CFB 188   167  TONE PERIOD       = 956
6CFD 14    169  CALL-TONE-PERIOD
```

```
6CFE 188  170   TONE PERIOD          = 956
6D00 14   172   CALL-TONE-PERIOD
6D01 126  173   TONE PERIOD          = 638
6D03 14   175   CALL-TONE-PERIOD
6D04 126  176   TONE PERIOD          = 638
6D06 11   178   CALL-CHANNEL
                ISSUE ON A
6D08 14   180   CALL-TONE-PERIOD
6D09 24   181   TONE PERIOD          = 536
6D0B 11   183   CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6D0D 14   185   CALL-TONE-PERIOD
6D0E 24   186   TONE PERIOD          = 536
6D10 11   188   CALL-CHANNEL
                ISSUE ON A
6D12 14   190   CALL-TONE-PERIOD
6D13 126  191   TONE PERIOD          = 638
6D15 11   193   CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6D17 14   195   CALL-TONE-PERIOD
6D18 126  196   TONE PERIOD          = 638
6D1A 14   198   CALL-TONE-PERIOD
6D1B 126  199   TONE PERIOD          = 638
6D1D 14   201   CALL-TONE-PERIOD
6D1E 126  202   TONE PERIOD          = 638
6D20 14   204   CALL-TONE-PERIOD
6D21 83   205   TONE PERIOD          = 851
6D23 11   207   CALL-CHANNEL
                ISSUE ON A
6D25 14   209   CALL-TONE-PERIOD
6D26 83   210   TONE PERIOD          = 851
6D28 14   212   CALL-TONE-PERIOD
6D29 204  213   TONE PERIOD          = 716
6D2B 14   215   CALL-TONE-PERIOD
6D2C 204  216   TONE PERIOD          = 716
6D2E 14   218   CALL-TONE-PERIOD
6D2F 83   219   TONE PERIOD          = 851
6D31 14   221   CALL-TONE-PERIOD
6D32 83   222   TONE PERIOD          = 851
6D34 11   224   CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6D36 14   226   CALL-TONE-PERIOD
6D37 83   227   TONE PERIOD          = 851
6D39 14   229   CALL-TONE-PERIOD
6D3A 83   230   TONE PERIOD          = 851
6D3C 14   232   CALL-TONE-PERIOD
6D3D 56   233   TONE PERIOD          = 568
6D3F 11   235   CALL-CHANNEL
                ISSUE ON A
6D41 14   237   CALL-TONE-PERIOD
6D42 56   238   TONE PERIOD          = 568
6D44 14   240   CALL-TONE-PERIOD
6D45 222  241   TONE PERIOD          = 478
6D47 14   243   CALL-TONE-PERIOD
6D48 222  244   TONE PERIOD          = 478
6D4A 14   246   CALL-TONE-PERIOD
6D4B 56   247   TONE PERIOD          = 568
6D4D 14   249   CALL-TONE-PERIOD
6D4E 56   250   TONE PERIOD          = 568
6D50 11   252   CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6D52 14   254   CALL-TONE-PERIOD
6D53 188  255   TONE PERIOD          = 956
6D55 14   257   CALL-TONE-PERIOD
6D56 188  258   TONE PERIOD          = 956
6D58 14   260   CALL-TONE-PERIOD
6D59 126  261   TONE PERIOD          = 638
```

```
6D5B 14   263   CALL-TONE-PERIOD
6D5C 126  264   TONE PERIOD          = 638
6D5E 11   266   CALL-CHANNEL
                ISSUE ON A
6D60 14   268   CALL-TONE-PERIOD
6D61 24   269   TONE PERIOD          = 536
6D63 11   271   CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6D65 14   273   CALL-TONE-PERIOD
6D66 24   274   TONE PERIOD          = 536
6D68 11   276   CALL-CHANNEL
                ISSUE ON A
6D6A 14   278   CALL-TONE-PERIOD
6D6B 126  279   TONE PERIOD          = 638
6D6D 11   281   CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
6D6F 14   283   CALL-TONE-PERIOD
6D70 126  284   TONE PERIOD          = 638
6D72 14   286   CALL-TONE-PERIOD
6D73 126  287   TONE PERIOD          = 638
6D75 14   289   CALL-TONE-PERIOD
6D76 126  290   TONE PERIOD          = 638
6D78 14   292   CALL-TONE-PERIOD
6D79 83   293   TONE PERIOD          = 851
6D7B 11   295   CALL-CHANNEL
                ISSUE ON A
6D7D 14   297   CALL-TONE-PERIOD
6D7E 83   298   TONE PERIOD          = 851
6D80 14   300   CALL-TONE-PERIOD
6D81 204  301   TONE PERIOD          = 716
6D83 14   303   CALL-TONE-PERIOD
6D84 204  304   TONE PERIOD          = 716
6D86 14   306   CALL-TONE-PERIOD
6D87 83   307   TONE PERIOD          = 851
6D89 14   309   CALL-TONE-PERIOD
6D8A 83   310   TONE PERIOD          = 851
6D8C 14   312   CALL-TONE-PERIOD
6D8D 126  313   TONE PERIOD          = 638
6D8F 14   315   CALL-TONE-PERIOD
6D90 126  316   TONE PERIOD          = 638
6D92 14   318   CALL-TONE-PERIOD
6D93 83   319   TONE PERIOD          = 851
6D95 14   321   CALL-TONE-PERIOD
6D96 83   322   TONE PERIOD          = 851
6D98 14   324   CALL-TONE-PERIOD
6D99 204  325   TONE PERIOD          = 716
6D9B 14   327   CALL-TONE-PERIOD
6D9C 204  328   TONE PERIOD          = 716
6D9E 14   330   CALL-TONE-PERIOD
6D9F 83   331   TONE PERIOD          = 851
6DA1 14   333   CALL-TONE-PERIOD
6DA2 83   334   TONE PERIOD          = 851
6DA4 9    336   JUMP
6DA5 48   337   PC ADDRESS           = 48
```

CHANNEL B

```
6E44 1    1     WAIT-SOUND
                ISSUE ON B
                RENDEZVOUS WITH A
                RENDEZVOUS WITH C
6E46 2    3     AMPLITUDE ENVELOPE= 2
6E47 0    4     TONE ENVELOPE       = 0
6E48 213  5     TONE PERIOD         = 213
6E4A 0    7     NOISE PERIOD        = 0
6E4B 0    8     INITIAL VOLUME      = 0
6E4C 23   9     DURATION            = 23
6E4E 11   11    CALL-CHANNEL
                ISSUE ON B
```



```
                RENDEZVOUS WITH A
                RENDEZVOUS WITH C
6E50 14   13    CALL-TONE-PERIOD
6E51 28   14    TONE PERIOD         = 284
6E53 14   16    CALL-TONE-PERIOD
6E54 28   17    TONE PERIOD         = 284
6E56 14   19    CALL-TONE-PERIOD
6E57 253  20    TONE PERIOD         = 253
6E59 14   22    CALL-TONE-PERIOD
6E5A 28   23    TONE PERIOD         = 284
6E5C 14   25    CALL-TONE-PERIOD
6E5D 28   26    TONE PERIOD         = 284
6E5F 14   28    CALL-TONE-PERIOD
6E60 213  29    TONE PERIOD         = 213
6E62 14   31    CALL-TONE-PERIOD
6E63 28   32    TONE PERIOD         = 284
6E65 14   34    CALL-TONE-PERIOD
6E66 213  35    TONE PERIOD         = 213
6E68 14   37    CALL-TONE-PERIOD
6E69 213  38    TONE PERIOD         = 213
6E6B 14   40    CALL-TONE-PERIOD
6E6C 28   41    TONE PERIOD         = 284
6E6E 14   43    CALL-TONE-PERIOD
6E6F 28   44    TONE PERIOD         = 284
6E71 14   46    CALL-TONE-PERIOD
6E72 253  47    TONE PERIOD         = 253
6E74 14   49    CALL-TONE-PERIOD
6E75 213  50    TONE PERIOD         = 213
6E77 14   52    CALL-TONE-PERIOD
6E78 213  53    TONE PERIOD         = 213
6E7A 14   55    CALL-TONE-PERIOD
6E7B 190  56    TONE PERIOD         = 190
6E7D 14   58    CALL-TONE-PERIOD
6E7E 213  59    TONE PERIOD         = 213
6F80 14   61    CALL-TONE-PERIOD
6E81 213  62    TONE PERIOD         = 213
6E83 14   64    CALL-TONE-PERIOD
6E84 159  65    TONE PERIOD         = 159
6E86 14   67    CALL-TONE-PERIOD
6E87 213  68    TONE PERIOD         = 213
6E89 14   70    CALL-TONE-PERIOD
6E8A 159  71    TONE PERIOD         = 159
6E8C 14   73    CALL-TONE-PERIOD
6E8D 213  74    TONE PERIOD         = 213
6E8F 14   76    CALL-TONE-PERIOD
6E90 28   77    TONE PERIOD         = 284
6E92 14   79    CALL-TONE-PERIOD
6E93 28   80    TONE PERIOD         = 284
6E95 14   82    CALL-TONE-PERIOD
6E96 253  83    TONE PERIOD         = 253
6E98 14   85    CALL-TONE-PERIOD
6E99 190  86    TONE PERIOD         = 190
6E9B 14   88    CALL-TONE-PERIOD
6E9C 190  89    TONE PERIOD         = 190
6E9E 14   91    CALL-TONE-PERIOD
6E9F 169  92    TONE PERIOD         = 169
6EA1 14   94    CALL-TONE-PERIOD
6EA2 213  95    TONE PERIOD         = 213
6EA4 14   97    CALL-TONE-PERIOD
6EA5 213  98    TONE PERIOD         = 213
6EA7 14   100   CALL-TONE-PERIOD
6EA8 159  101   TONE PERIOD         = 159
6EAA 14   103   CALL-TONE-PERIOD
6EAB 213  104   TONE PERIOD         = 213
6EAD 14   106   CALL-TONE-PERIOD
6EAE 159  107   TONE PERIOD         = 159
6EB0 14   109   CALL-TONE-PERIOD
6EB1 213  110   TONE PERIOD         = 213
6EB3 14   112   CALL-TONE-PERIOD
6EB4 28   113   TONE PERIOD         = 284
6EB6 14   115   CALL-TONE-PERIOD
6EB7 28   116   TONE PERIOD         = 284
6EB9 14   118   CALL-TONE-PERIOD
6EBA 253  119   TONE PERIOD         = 253
6EBC 9    121   JUMP
6EBD 11   122   PC ADDRESS          = 11
```

CHANNEL C

```
6DAC 1    1     WAIT-SOUND
                ISSUE ON C
                RENDEZVOUS WITH A
                RENDEZVOUS WITH B
6DAE 2    3     AMPLITUDE ENVELOPE= 2
6DAF 0    4     TONE ENVELOPE       = 0
6DB0 134  5     TONE PERIOD         = 134
6DB2 0    7     NOISE PERIOD        = 0
6DB3 0    8     INITIAL VOLUME      = 0
6DB4 23   9     DURATION            = 23
6DB6 11   11    CALL-CHANNEL
                ISSUE ON C
                RENDEZVOUS WITH A
                RENDEZVOUS WITH B
6DB8 14   13    CALL-TONE-PERIOD
6DB9 179  14    TONE PERIOD         = 179
6DBB 14   16    CALL-TONE-PERIOD
6DBC 179  17    TONE PERIOD         = 179
6DBE 15   19    CALL-NOISE-PERIOD
6DBF 1    20    NOISE PERIOD        = 1
6DC0 14   21    CALL-TONE-PERIOD
6DC1 159  22    TONE PERIOD         = 159
6DC3 15   24    CALL-NOISE-PERIOD
6DC4 0    25    NOISE PERIOD        = 0
6DC5 14   26    CALL-TONE-PERIOD
6DC6 179  27    TONE PERIOD         = 179
6DC8 14   29    CALL-TONE-PERIOD
6DC9 179  30    TONE PERIOD         = 179
6DCB 14   32    CALL-TONE-PERIOD
6DCC 213  33    TONE PERIOD         = 213
6DCE 14   35    CALL-TONE-PERIOD
6DCF 179  36    TONE PERIOD         = 179
6DD1 14   38    CALL-TONE-PERIOD
6DD2 213  39    TONE PERIOD         = 213
6DD4 14   41    CALL-TONE-PERIOD
6DD5 213  42    TONE PERIOD         = 213
6DD7 14   44    CALL-TONE-PERIOD
6DD8 179  45    TONE PERIOD         = 179
6DDA 14   47    CALL-TONE-PERIOD
6DDB 179  48    TONE PERIOD         = 179
6DDD 15   50    CALL-NOISE-PERIOD
6DDE 1    51    NOISE PERIOD        = 1
6DDF 14   52    CALL-TONE-PERIOD
6DE0 159  53    TONE PERIOD         = 159
6DE2 15   55    CALL-NOISE-PERIOD
6DE3 0    56    NOISE PERIOD        = 0
6DE4 14   57    CALL-TONE-PERIOD
6DE5 134  58    TONE PERIOD         = 134
6DE7 14   60    CALL-TONE-PERIOD
6DE8 134  61    TONE PERIOD         = 134
6DEA 15   63    CALL-NOISE-PERIOD
6DEB 1    64    NOISE PERIOD        = 1
6DEC 14   65    CALL-TONE-PERIOD
6DED 119  66    TONE PERIOD         = 119
6DEF 15   68    CALL-NOISE-PERIOD
6DF0 0    69    NOISE PERIOD        = 0
6DF1 14   70    CALL-TONE-PERIOD
6DF2 134  71    TONE PERIOD         = 134
6DF4 14   73    CALL-TONE-PERIOD
6DF5 134  74    TONE PERIOD         = 134
6DF7 14   76    CALL-TONE-PERIOD
6DF8 159  77    TONE PERIOD         = 159
6DFA 14   79    CALL-TONE-PERIOD
6DFB 134  80    TONE PERIOD         = 134
6DFD 14   82    CALL-TONE-PERIOD
6DFE 159  83    TONE PERIOD         = 159
6E00 14   85    CALL-TONE-PERIOD
6E01 213  86    TONE PERIOD         = 213
6E03 14   88    CALL-TONE-PERIOD
6E04 179  89    TONE PERIOD         = 179
6E06 14   91    CALL-TONE-PERIOD
6E07 179  92    TONE PERIOD         = 179
6E09 15   94    CALL-NOISE-PERIOD
```

```
6E0A 1     95   NOISE PERIOD          = 1
6E0B 14    96   CALL-TONE-PERIOD
6E0C 159   97   TONE PERIOD           = 159
6E0E 15    99   CALL-NOISE-PERIOD
6E0F 0     100  NOISE PERIOD          = 0
6E10 14    101  CALL-TONE-PERIOD
6E11 119   102  TONE PERIOD           = 119
6E13 14    104  CALL-TONE-PERIOD
6E14 119   105  TONE PERIOD           = 119
6E16 15    107  CALL-NOISE-PERIOD
6E17 1     108  NOISE PERIOD          = 1
6E18 14    109  CALL-TONE-PERIOD
6E19 213   110  TONE PERIOD           = 213
6E1B 15    112  CALL-NOISE-PERIOD
6E1C 0     113  NOISE PERIOD          = 0
6E1D 14    114  CALL-TONE-PERIOD
6E1E 134   115  TONE PERIOD           = 134
6E20 14    117  CALL-TONE-PERIOD
6E21 134   118  TONE PERIOD           = 134

6E23 14    120  CALL-TONE-PERIOD
6E24 15?   121  TONE PERIOD           = 159
6E26 14    123  CALL-TONE-PERIOD
6E27 134   124  TONE PERIOD           = 134
6E29 14    126  CALL-TONE-PERIOD
6E2A 159   127  TONE PERIOD           = 159
6E2C 14    129  CALL-TONE-PERIOD
6E2D 213   130  TONE PERIOD           = 213
6E2F 14    132  CALL-TONE-PERIOD
6E30 179   133  TONE PERIOD           = 179
6E32 14    135  CALL-TONE-PERIOD
6E33 179   136  TONE PERIOD           = 179
6E35 15    138  CALL-NOISE-PERIOD
6E36 1     139  NOISE PERIOD          = 1
6E37 14    140  CALL-TONE-PERIOD
6E38 159   141  TONE PERIOD           = 159
6E3A 15    143  CALL-NOISE-PERIOD
6E3B 0     144  NOISE PERIOD          = 0
6E3C 9     145  JUMP
6E3D 11    146  PC ADDRESS            = 11
```

DAS GLÖCKNERSPIEL

CHANNEL A

```
68DF 1     1    WAIT-SOUND
                ISSUE ON A
68E1 1     3    AMPLITUDE ENVELOPE=  1
68E2 1     4    TONE ENVELOPE        =  1
68E3 90    5    TONE PERIOD          =  602
68E5 0     7    NOISE PERIOD         =  0
68E6 0     8    INITIAL VOLUME       =  0
68E7 20    9    DURATION             =  20
68E9 2     11   RESET
68EA 6     12   AMP-ENV
68EB 1     13   AMPLITUDE ENVELOPE=  1
68EC 5     14   NO OF SECTIONS       =  5
68ED 2     15   STEP COUNT           =  2
68EE 5     16   STEP SIZE            =  5
68EF 1     17   PAUSE TIME           =  1
68F0 1     18   STEP COUNT           =  1
68F1 5     19   STEP SIZE            =  5
68F2 1     20   PAUSE TIME           =  1
68F3 1     21   STEP COUNT           =  1
68F4 0     22   STEP SIZE            =  0
68F5 5     23   PAUSE TIME           =  5
68F6 10    24   STEP COUNT           =  10
68F7 255   25   STEP SIZE            =  255
68F8 1     26   PAUSE TIME           =  1
68F9 5     27   STEP COUNT           =  5
68FA 255   28   STEP SIZE            =  255
68FB 10    29   PAUSE TIME           =  10
68FC 6     30   AMP-ENV
68FD 2     31   AMPLITUDE ENVELOPE=  2
68FE 5     32   NO OF SECTIONS       =  5
68FF 2     33   STEP COUNT           =  2
6900 5     34   STEP SIZE            =  5
6901 1     35   PAUSE TIME           =  1
6902 1     36   STEP COUNT           =  1
6903 5     37   STEP SIZE            =  5
6904 1     38   PAUSE TIME           =  1
6905 1     39   STEP COUNT           =  1
6906 0     40   STEP SIZE            =  0
6907 5     41   PAUSE TIME           =  5
6908 4     42   STEP COUNT           =  4
6909 255   43   STEP SIZE            =  255
690A 1     44   PAUSE TIME           =  1
690B 5     45   STEP COUNT           =  5
690C 254   46   STEP SIZE            =  254

690D 5     47   PAUSE TIME           =  5
690E 6     48   AMP-ENV
690F 4     49   AMPLITUDE ENVELOPE=  4
6910 1     50   NO OF SECTIONS       =  1
6911 1     51   STEP COUNT           =  1
6912 0     52   STEP SIZE            =  0
6913 100   53   PAUSE TIME           =  100
6914 7     54   TONE-ENV
6915 255   55   TONE ENVELOPE        =  255
6916 3     56   NO OF SECTIONS       =  3
6917 2     57   STEP COUNT           =  2
6918 2     58   STEP SIZE            =  2
6919 2     59   PAUSE TIME           =  2
691A 4     60   STEP COUNT           =  4
691B 254   61   STEP SIZE            =  254
691C 2     62   PAUSE TIME           =  2
691D 2     63   STEP COUNT           =  2
691E 2     64   STEP SIZE            =  2
691F 2     65   PAUSE TIME           =  2
6920 6     66   AMP-ENV
6921 3     67   AMPLITUDE ENVELOPE=  3
6922 2     68   NO OF SECTIONS       =  2
                HARDWARE ENVELOPE
6923 138   69   ENVELOPE SHAPE       =  138
6924 142   70   ENVELOPE PERIOD      =  1422
6926 1     72   STEP COUNT           =  1
6927 0     73   STEP SIZE            =  0
6928 20    74   PAUSE TIME           =  20
6929 11    75   CALL-CHANNEL
                ISSUE ON A
                RENDEZVOUS WITH B
                RENDEZVOUS WITH C
692B 18    77   CALL-TONE-DURATION
692C 100   78   DURATION             =  100
692E 56    80   TONE PERIOD          =  568
6930 11    82   CALL-CHANNEL
                ISSUE ON A
6932 18    84   CALL-TONE-DURATION
6933 20    85   DURATION             =  20
6935 56    87   TONE PERIOD          =  568
6937 18    89   CALL-TONE-DURATION
6938 100   90   DURATION             =  100
693A 188   92   TONE PERIOD          =  956
693C 18    94   CALL-TONE-DURATION
693D 20    95   DURATION             =  20
693F 188   97   TONE PERIOD          =  956
6941 18    99   CALL-TONE-DURATION
```



```
6942 100   1    DURATION             =  100
6944 83    102  TONE PERIOD          =  851
6946 18    104  CALL-TONE-DURATION
6947 20    105  DURATION             =  20
6949 83    107  TONE PERIOD          =  851
694B 18    109  CALL-TONE-DURATION
694C 60    110  DURATION             =  60
694E 123   112  TONE PERIOD          =  379
6950 18    114  CALL-TONE-DURATION
6951 20    115  DURATION             =  20
6953 246   117  TONE PERIOD          =  758
6955 14    119  CALL-TONE-PERIOD
6956 164   120  TONE PERIOD          =  676
6958 14    122  CALL-TONE-PERIOD
6959 90    123  TONE PERIOD          =  602
695B 18    125  CALL-TONE-DURATION
695C 100   126  DURATION             =  100
695E 56    128  TONE PERIOD          =  568
6960 18    130  CALL-TONE-DURATION
6961 20    131  DURATION             =  20
6963 56    133  TONE PERIOD          =  568
6965 18    135  CALL-TONE-DURATION
6966 100   136  DURATION             =  100
6968 188   138  TONE PERIOD          =  956
696A 18    140  CALL-TONE-DURATION
696B 20    141  DURATION             =  20
696D 188   143  TONE PERIOD          =  956
696F 18    145  CALL-TONE-DURATION
6970 100   146  DURATION             =  100
6972 83    148  TONE PERIOD          =  851
6974 18    150  CALL-TONE-DURATION
6975 20    151  DURATION             =  20
6977 83    153  TONE PERIOD          =  851
6979 14    155  CALL-TONE-PERIOD
697A 123   156  TONE PERIOD          =  379
697C 14    158  CALL-TONE-PERIOD
697D 246   159  TONE PERIOD          =  758
697F 14    161  CALL-TONE-PERIOD
6980 204   162  TONE PERIOD          =  716
6982 14    164  CALL-TONE-PERIOD
6983 164   165  TONE PERIOD          =  676
6985 14    167  CALL-TONE-PERIOD
6986 126   168  TONE PERIOD          =  638
6988 14    170  CALL-TONE-PERIOD
6989 90    171  TONE PERIOD          =  602
698B 18    173  CALL-TONE-DURATION
698C 100   174  DURATION             =  100
698E 56    176  TONE PERIOD          =  568
6990 18    178  CALL-TONE-DURATION
6991 20    179  DURATION             =  20
6993 56    181  TONE PERIOD          =  568
6995 18    183  CALL-TONE-DURATION
6996 100   184  DURATION             =  100
6998 188   186  TONE PERIOD          =  956
699A 18    188  CALL-TONE-DURATION
699B 20    189  DURATION             =  20
699D 188   191  TONE PERIOD          =  956
699F 18    193  CALL-TONE-DURATION
69A0 100   194  DURATION             =  100
69A2 83    196  TONE PERIOD          =  851
69A4 18    198  CALL-TONE-DURATION
69A5 20    199  DURATION             =  20
69A7 83    201  TONE PERIOD          =  851
69A9 18    203  CALL-TONE-DURATION
69AA 100   204  DURATION             =  100
69AC 123   206  TONE PERIOD          =  379
69AE 18    208  CALL-TONE-DURATION
69AF 20    209  DURATION             =  20
69B1 123   211  TONE PERIOD          =  379
69B3 18    213  CALL-TONE-DURATION
69B4 60    214  DURATION             =  60
69B6 56    216  TONE PERIOD          =  568
69B8 14    218  CALL-TONE PERIOD
69B9 126   219  TONE PERIOD          =  638
69BB 18    221  CALL-TONE-DURATION
69BC 40    222  DURATION             =  40

69BE 123   224  TONE PERIOD          =  379
69C0 18    226  CALL-TONE-DURATION
69C1 20    227  DURATION             =  20
69C3 83    229  TONE PERIOD          =  851
69C5 18    231  CALL-TONE-DURATION
69C6 40    232  DURATION             =  40
69C8 188   234  TONE PERIOD          =  956
69CA 18    236  CALL-TONE-DURATION
69CB 20    237  DURATION             =  20
69CD 56    239  TONE PERIOD          =  568
69CF 18    241  CALL-TONE-DURATION
69D0 40    242  DURATION             =  40
69D2 83    244  TONE PERIOD          =  851
69D4 18    246  CALL-TONE-DURATION
69D5 20    247  DURATION             =  20
69D7 123   249  TONE PERIOD          =  379
69D9 18    251  CALL-TONE-DURATION
69DA 40    252  DURATION             =  40
69DC 188   254  TONE PERIOD          =  956
69DE 18    256  CALL-TONE-DURATION
69DF 60    257  DURATION             =  60
69E1 56    259  TONE PERIOD          =  568
69E3 18    261  CALL-TONE-DURATION
69E4 80    262  DURATION             =  80
69E6 123   264  TONE PERIOD          =  379
69E8 9     266  JUMP
69E9 75    267  PC ADDRESS           =  75
```

CHANNEL B

```
67F2 1     1    WAIT-SOUND
                ISSUE ON B
67F4 2     3    AMPLITUDE ENVELOPE=  2
67F5 1     4    TONE ENVELOPE        =  1
67F6 142   5    TONE PERIOD          =  142
67F8 0     7    NOISE PERIOD         =  0
67F9 0     8    INITIAL VOLUME       =  0
67FA 60    9    DURATION             =  60
67FC 11    11   CALL-CHANNEL
                ISSUE ON B
                RENDEZVOUS WITH A
                RENDEZVOUS WITH C
67FE 12    13   CALL-AMP-ENV
67FF 4     14   AMPLITUDE ENVELOPE=  4
6800 18    15   CALL-TONE-DURATION
6801 40    16   DURATION             =  40
6803 119   18   TONE PERIOD          =  119
6805 11    20   CALL-CHANNEL
                ISSUE ON B
6807 12    22   CALL-AMP-ENV
6808 2     23   AMPLITUDE ENVELOPE=  2
6809 18    24   CALL-TONE-DURATION
680A 20    25   DURATION             =  20
680C 119   27   TONE PERIOD          =  119
680E 18    29   CALL-TONE-DURATION
680F 60    30   DURATION             =  60
6811 142   32   TONE PERIOD          =  142
6813 18    34   CALL-TONE-DURATION
6814 40    35   DURATION             =  40
6816 159   37   TONE PERIOD          =  159
6818 18    39   CALL-TONE-DURATION
6819 20    40   DURATION             =  20
681B 190   42   TONE PERIOD          =  190
681D 18    44   CALL-TONE-DURATION
681E 40    45   DURATION             =  40
6820 213   47   TONE PERIOD          =  213
6822 18    49   CALL-TONE-DURATION
6823 20    50   DURATION             =  20
6825 239   52   TONE PERIOD          =  239
6827 18    54   CALL-TONE-DURATION
6828 60    55   DURATION             =  60
682A 213   57   TONE PERIOD          =  213
682C 18    59   CALL-TONE-DURATION
```

```
682F 213  62   TONE PERIOD          = 213
6831 18   64   CALL-TONE-DURATION
6832 20   65   DURATION            = 20
6834 239  67   TONE PERIOD         = 239
6836 18   69   CALL-TONE-DURATION
6837 40   70   DURATION            = 40
6839 213  72   TONE PERIOD         = 213
683B 18   74   CALL-TONE-DURATION
683C 120  75   DURATION            = 120
683E 190  77   TONE PERIOD         = 190
6840 18   79   CALL-TONE-DURATION
6841 20   80   DURATION            = 20
6843 119  82   TONE PERIOD         = 119
6845 18   84   CALL-TONE-DURATION
6846 60   85   DURATION            = 60
6848 142  87   TONE PERIOD         = 142
684A 18   89   CALL-TONE-DURATION
684B 40   90   DURATION            = 40
684D 159  92   TONE PERIOD         = 159
684F 18   94   CALL-TONE-DURATION
6850 20   95   DURATION            = 20
6852 190  97   TONE PERIOD         = 190
6854 18   99   CALL-TONE-DURATION
6855 40   100  DURATION            = 40
6857 213  102  TONE PERIOD         = 213
6859 18   104  CALL-TONE-DURATION
685A 20   105  DURATION            = 20
685C 239  107  TONE PERIOD         = 239
685E 18   109  CALL-TONE-DURATION
685F 40   110  DURATION            = 40
6861 213  112  TONE PERIOD         = 213
6863 18   114  CALL-TONE-DURATION
6864 20   115  DURATION            = 20
6866 239  117  TONE PERIOD         = 239
6868 18   119  CALL-TONE-DURATION
6869 40   120  DURATION            = 40
686B 213  122  TONE PERIOD         = 213
686D 18   124  CALL-TONE-DURATION
686E 180  125  DURATION            = 180
6870 190  127  TONE PERIOD         = 190
6872 18   129  CALL-TONE-DURATION
6873 20   130  DURATION            = 20
6875 80   132  TONE PERIOD         = 80
6877 18   134  CALL-TONE-DURATION
6878 60   135  DURATION            = 60
687A 95   137  TONE PERIOD         = 95
687C 18   139  CALL-TONE-DURATION
687D 40   140  DURATION            = 40
687F 106  142  TONE PERIOD         = 106
6881 18   144  CALL-TONE-DURATION
6882 20   145  DURATION            = 20
6884 127  147  TONE PERIOD         = 127
6886 18   149  CALL-TONE-DURATION
6887 40   150  DURATION            = 40
6889 142  152  TONE PERIOD         = 142
688B 18   154  CALL-TONE-DURATION
688C 20   155  DURATION            = 20
688E 159  157  TONE PERIOD         = 159
6890 18   159  CALL-TONE-DURATION
6891 60   160  DURATION            = 60
6893 142  162  TONE PERIOD         = 142
6895 14   164  CALL-TONE-PERIOD
6896 142  165  TONE PERIOD         = 142
6898 18   167  CALL-TONE-DURATION
6899 40   168  DURATION            = 40
689B 142  170  TONE PERIOD         = 142
689D 18   172  CALL-TONE-DURATION
689E 80   173  DURATION            = 80
68A0 127  175  TONE PERIOD         = 127
68A2 18   177  CALL-TONE-DURATION
68A3 60   178  DURATION            = 60
68A5 142  180  TONE PERIOD         = 142
68A7 14   182  CALL-TONE-PERIOD
68A8 159  183  TONE PERIOD         = 159
68AA 18   185  CALL-TONE-DURATION
68AB 40   186  DURATION            = 40
68AD 190  188  TONE PERIOD         = 190
68AF 18   190  CALL-TONE-DURATION
68B0 20   191  DURATION            = 20
68B2 213  193  TONE PERIOD         = 213
68B4 18   195  CALL-TONE-DURATION
68B5 40   196  DURATION            = 40
68B7 239  198  TONE PERIOD         = 239
68B9 18   200  CALL-TONE-DURATION
68BA 20   201  DURATION            = 20
68BC 28   203  TONE PERIOD         = 284
68BE 18   205  CALL-TONE-DURATION
68BF 40   206  DURATION            = 40
68C1 213  208  TONE PERIOD         = 213
68C3 18   210  CALL-TONE-DURATION
68C4 20   211  DURATION            = 20
68C6 190  213  TONE PERIOD         = 190
68C8 18   215  CALL-TONE-DURATION
68C9 40   216  DURATION            = 40
68CB 237  218  TONE PERIOD         = 237
68CD 18   220  CALL-TONE-DURATION
68CE 60   221  DURATION            = 60
68D0 28   223  TONE PERIOD         = 284
68D2 18   225  CALL-TONE-DURATION
68D3 80   226  DURATION            = 80
68D5 150  228  TONE PERIOD         = 150
68D7 9    230  JUMP
68D8 11   231  PC ADDRESS          = 11

CHANNEL C

6F0F 1    1    WAIT-SOUND
               ISSUE ON C
6F11 3    3    AMPLITUDE ENVELOPE= 3
6F12 3    4    TONE ENVELOPE      = 3
6F13 71   5    TONE PERIOD        = 71
6F15 0    7    NOISE PERIOD       = 0
6F16 0    8    INITIAL VOLUME     = 0
6F17 60   9    DURATION           = 60
6F19 12   11   CALL-AMP-ENV
6F1A 4    12   AMPLITUDE ENVELOPE= 4
6F1B 11   13   CALL-CHANNEL
               ISSUE ON C
               RENDEZVOUS WITH A
               RENDEZVOUS WITH B
6F1D 18   15   CALL-TONE-DURATION
6F1E 40   16   DURATION           = 40
6F20 119  18   TONE PERIOD        = 119
6F22 12   20   CALL-AMP-ENV
6F23 3    21   AMPLITUDE ENVELOPE= 3
6F24 11   22   CALL-CHANNEL
               ISSUE ON C
6F26 18   24   CALL-TONE-DURATION
6F27 20   25   DURATION           = 20
6F29 60   27   TONE PERIOD        = 60
6F2B 18   29   CALL-TONE-DURATION
6F2C 60   30   DURATION           = 60
6F2E 71   32   TONE PERIOD        = 71
6F30 18   34   CALL-TONE-DURATION
6F31 40   35   DURATION           = 40
6F33 80   37   TONE PERIOD        = 80
6F35 18   39   CALL-TONE-DURATION
6F36 20   40   DURATION           = 20
6F38 95   42   TONE PERIOD        = 95
6F3A 18   44   CALL-TONE-DURATION
6F3B 40   45   DURATION           = 40
6F3D 106  47   TONE PERIOD        = 106
6F3F 18   49   CALL-TONE-DURATION
6F40 20   50   DURATION           = 20
6F42 119  52   TONE PERIOD        = 119
6F44 18   54   CALL-TONE-DURATION
6F45 60   55   DURATION           = 60
6F47 106  57   TONE PERIOD        = 106
```



```
6F49 18   59   CALL-TONE-DURATION
6F4A 40   60   DURATION           = 40
6F4C 106  62   TONE PERIOD        = 106
6F4E 18   64   CALL-TONE-DURATION
6F4F 20   65   DURATION           = 20
6F51 119  67   TONE PERIOD        = 119
6F53 18   69   CALL-TONE-DURATION
6F54 40   70   DURATION           = 40
6F56 106  72   TONE PERIOD        = 106
6F58 18   74   CALL-TONE-DURATION
6F59 120  75   DURATION           = 120
6F5B 95   77   TONE PERIOD        = 95
6F5D 18   79   CALL-TONE-DURATION
6F5E 20   80   DURATION           = 20
6F60 60   82   TONE PERIOD        = 60
6F62 18   84   CALL-TONE-DURATION
6F63 60   85   DURATION           = 60
6F65 71   87   TONE PERIOD        = 71
6F67 18   89   CALL-TONE-DURATION
6F68 40   90   DURATION           = 40
6F6A 80   92   TONE PERIOD        = 80
6F6C 18   94   CALL-TONE-DURATION
6F6D 20   95   DURATION           = 20
6F6F 95   97   TONE PERIOD        = 95
6F71 18   99   CALL-TONE-DURATION
6F72 40   100  DURATION           = 40
6F74 106  102  TONE PERIOD        = 106
6F76 18   104  CALL-TONE-DURATION
6F77 20   105  DURATION           = 20
6F79 119  107  TONE PERIOD        = 119
6F7B 18   109  CALL-TONE-DURATION
6F7C 40   110  DURATION           = 40
6F7E 106  112  TONE PERIOD        = 106
6F80 18   114  CALL-TONE-DURATION
6F81 20   115  DURATION           = 20
6F83 119  117  TONE PERIOD        = 119
6F85 18   119  CALL-TONE-DURATION
6F86 40   120  DURATION           = 40
6F88 106  122  TONE PERIOD        = 106
6F8A 18   124  CALL-TONE-DURATION
6F8B 180  125  DURATION           = 180
6F8D 95   127  TONE PERIOD        = 95
6F8F 18   129  CALL-TONE-DURATION
6F90 20   130  DURATION           = 20
6F92 60   132  TONE PERIOD        = 60
6F94 18   134  CALL-TONE-DURATION
6F95 60   135  DURATION           = 60
6F97 71   137  TONE PERIOD        = 71
6F99 18   139  CALL-TONE-DURATION
6F9A 40   140  DURATION           = 40
6F9C 80   142  TONE PERIOD        = 80
6F9E 18   144  CALL-TONE-DURATION
6F9F 20   145  DURATION           = 20
6FA1 95   147  TONE PERIOD        = 95
6FA3 18   149  CALL-TONE-DURATION
6FA4 40   150  DURATION           = 40
6FA6 106  152  TONE PERIOD        = 106
6FA8 18   154  CALL-TONE-DURATION
6FA9 20   155  DURATION           = 20
6FAB 119  157  TONE PERIOD        = 119
6FAD 18   159  CALL-TONE-DURATION
6FAE 60   160  DURATION           = 60
6FB0 106  162  TONE PERIOD        = 106
6FB2 18   164  CALL-TONE-DURATION
6FB3 40   165  DURATION           = 40
6FB5 106  167  TONE PERIOD        = 106
6FB7 18   169  CALL-TONE-DURATION
6FB8 20   170  DURATION           = 20
6FBA 119  172  TONE PERIOD        = 119
6FBC 18   174  CALL-TONE-DURATION
6FBD 40   175  DURATION           = 40
6FBF 106  177  TONE PERIOD        = 106
6FC1 18   179  CALL-TONE-DURATION
6FC2 80   180  DURATION           = 80
6FC4 95   182  TONE PERIOD        = 95
6FC6 18   184  CALL-TONE-DURATION
6FC7 60   185  DURATION           = 60
6FC9 71   187  TONE PERIOD        = 71
6FCB 14   189  CALL-TONE-PERIOD
6FCC 80   190  TONE PERIOD        = 80
6FCE 18   192  CALL-TONE-DURATION
6FCF 40   193  DURATION           = 40
6FD1 95   195  TONE PERIOD        = 95
6FD3 18   197  CALL-TONE-DURATION
6FD4 20   198  DURATION           = 20
6FD6 106  200  TONE PERIOD        = 106
6FD8 18   202  CALL-TONE-DURATION
6FD9 40   203  DURATION           = 40
6FDB 119  205  TONE PERIOD        = 119
6FDD 18   207  CALL-TONE-DURATION
6FDE 20   208  DURATION           = 20
6FE0 142  210  TONE PERIOD        = 142
6FE2 18   212  CALL-TONE-DURATION
6FE3 40   213  DURATION           = 40
6FE5 106  215  TONE PERIOD        = 106
6FE7 18   217  CALL-TONE-DURATION
6FE8 20   218  DURATION           = 20
6FEA 95   220  TONE PERIOD        = 95
6FEC 18   222  CALL-TONE-DURATION
6FED 40   223  DURATION           = 40
6FEF 119  225  TONE PERIOD        = 119
6FF1 18   227  CALL-TONE-DURATION
6FF2 60   228  DURATION           = 60
6FF4 142  230  TONE PERIOD        = 142
6FF6 18   232  CALL-TONE-DURATION
6FF7 80   233  DURATION           = 80
6FF9 119  235  TONE PERIOD        = 119
6FFB 9    237  JUMP
6FFC 11   238  PC ADDRESS         = 11
```

VERFOLGUNGSSPRITES

Tonsprites und Verfolgungssprites sind die beiden schwierigsten Fu nen von Laser-BASIC. Wir raten Ihnen daher, diese beiden Spritestypen erst dann in Angriff zu nehmen, wenn Sie das Paket etwas besser kennen. Verfolgungssprites verfügen über einen begrenzten Kollisonsschutz. Es ist für uns nicht immer leicht, technische Fragen in Hinblick auf die richtige Verwendung von Verfolgungssprites zu beantworten. Die Ergebnisse können allerdings nach einer gewissen Übungszeit recht zufriedenstellend ausfallen. In diesem letzten Teil des Abschnitts "STARTEN MIT LASER-BASIC" werden wir uns einige Beispiele zur Verwendung von Verfolgungssprites ansehen.

In diesem Zuasammenhang versteht man unter einem Verfolgungssprite eine Sprite, das ein Programm zur Bewegungssteuerung eines Grafiksprites bzw. von Sprites enthält. Das interne Format wird unter "Die Laser-BASIC-Befehle im Detail" beschrieben. Da Verfolgungssprites gewöhnlich verwendet werden, um Sprites auf vorbestimmten Spuren zu bewegen, ist es notwendig ein Sprite wie z.B. PTXR auf den Bildschirm zu bringen. Wird normalerweise vor XMOV bzw. XBNC gesetzt. Dies wird meistens mit dem TPUT-Befehl ausgeführt, der das Bild auf den Bildschirm bringt und einige der internen Parameter der Verfolger einleitet. Der Verfolger hat eine eigene Aufzeichnung über die letzte Position, in welche er ein Sprite gesetzt hat, und welches Sprite er zuletzt auf den Bildschirm gebracht hat und welche Art von 'PUT' er verwendet, und wo er in sich selbst die Ausführung durchführt. Diese Parameter können mit POKE-Befehlen aufgestellt werden. Es ist aber wahrscheinlich einfacher, diese mit einem TPUT-Befehl aufzubauen.

Bei diesem ersten Beispiel werden wir ein ganz einfaches Verfolgungssprite aufbauen, das ein Sprite um ein vordefiniertes Fenster hüpfen läßt und sich selbst dreht. Tippen Sie "NEW", um das laufende Programm zu löschen, und geben Sie anschließend folgendes ein und lassen Sie es laufen (RUN):

```
5 ' BEISPIEL VON VERFOLGUNGSSPRITES
10 X%=0:Y%=0:!SPN,40:!SPR,@X%,@X%,@X%,@Y%
20 DATA 2,0,4,4,181,162,0,0
30 FOR I%=Y%+5 TO Y%+12 :READ X%:POKE I%,X%:NEXT I%
40 !SET,4:!KEY,51:!SPN,51 :!COL,0  :!ROW,75:!HGT,50:!LEN,80:!BWST
50 !HGT,4:!LEN,2 :!COL,40 :!ROW,100:!SP1,46:!SP2,47:!SP3,48:!SP4,49
60 !SET,0:!COL,40:!ROW,100:!KEY,40 :!SP1,46:!SCLS  :!TPUT,1
70 !TMOV,1000,1
```

Sprite 46 sollte in Spalte 15, Zeile 25 erscheinen.

Zeile 10 Leitet X% und Y% ein und untersucht anschließend Sprite 40, um seine dem Y% zugeteilte Anfangsadresse zu finden. Sprite 40 ist 13 Bytes lang.

Zeile 20 Daten für das Sprite-Verfolgungsprogramm.

Zeile 30 POKEt in das Verfolgungsspriteprogramm. Die ersten fünf Bytes des Sprites 40 werden vom Verfolgerprogramm selst gebraucht. Die Daten haben folgende Bedeutung:

1. Byte = 2	Dies ist die Bewegungsart – XOR
2. Byte = 0	Dies ist das erste Byte des Programms, das den Verfolger darauf hinweist, daß das nächste Byte eine Steurcode ist.
3. Byte = 4	Dieser Code weist den Verfolger darauf hin, daß das nächste Byte eine SET-Nummer ist und die beiden nachfolgenden Bytes die Adresse eines Laser-BASIC-Befehls.
4. Byte = 4	Dies ist das SET – SET 4
5. Byte = 181	Diese beiden Bytes sind die Ausführungsadresse vom
6. Byte = 162	Laser-BASIC-Befehl XBNC, die durch ADDR ermittelt wurde (siehe Abschnitt über compilerbezogene Befehle)
7. Byte = 0	Dies weist den Verfolger wieder darauf hin, daß ein Steuercode zu erwarten ist
8. Byte = 0	Dieser Steuercode laßt den Verfolger die erste Anweisung ausfuhren. Da es nur eine andere Anweisung ('XBNC') gibt, bedeutet dies, daß der Verfolger zurückspringen wird und bei jedem Aufruf ein 'XBNC' ausführt

Variablen die notwendigen Werte zuzuteilen. Wir werden die Spur in Spalte 15 und Zeile 25 mit einem Sprite 46 beginnen. Damit werden diese und alle anderen Variablen sc das Sprungfenster in dieser Zeile und auf Zeile 50 aufgebaut.

Zeile 60 Diese Zeile bestimmt die Startspalte und -zeile, setzt die Verfolgernummer in KEY, weist den Verfolger darauf hin, daß das erste Sprite, das auf den Bildschirm gebracht werden soll, Sprite 46 ist, und setzt es dann mit einem TPUT in Gang. Die "1" nach dem TPUT weist den Verfolger darauf hin, daß die erste Anweisung einen Zählerwert von 1 hat, d.h. die erste Anweisung nach dem "Tipp"-Byte.

Dieses Programm baut das Verfolgersprite auf und bringt es auf den Bildschirm. Wir brauchen jetzt nur noch die eigentliche Spur ausführen. Dazu müssen wir bei 60 eine Zeile anreihen, die die Spur in einer Maschinencode-Schleife ausführen wird. Löschen Sie nicht das im Speicher befindliche Programm, sondern tippen Sie nur:

```
70 ITMOV,1000,1
```

Dies wird 1000-mal mit Rücklaufsynchronisierung ausgeführt. Um dies auszuführen, brauchen Sie nur "RUN" einzutippen.

Betrachten wir uns jetzt ein weiteres Beispiel. Diesmal werden wir ein Sprite mittels Joystick bzw. Tastatur am Bildschirm herumbewegen und die Kollisionserfassung einschalten. Löschen Sie das letzte Beispiel durch Eintippen von "NEW" und geben Sie folgendes ein:

```
5 ' BEISPIEL VON VERFOLGUNGSSPRITES II
10 X%=0:Y%=0:!SPN,41:!SPR,@X%,@X%,@X%,@Y%
20 DATA 206,50,2,8,0,0
30 !SET,0
40 FOR I%=Y%+5 TO Y%+10:READ X%:POKE I%,X%:NEXT I%
50 !COL,15:!ROW,25:!KEY,41:!SP1,50:!TPUT,1
60 A$="TMOVA#":!SET,@A$:!RUN,0
70 GOTO 70
```

Zeile 10 Leitet X% und Y% ein und teilt Y% die Anfangsadresse von Sprite 41 zu.

Zeile 20 Diese enthält die Daten für das Verfolgungsspriteprogramm, die folgende Bedeutung haben:

1. Byte = 206	Dies ist das Bewegungsartbyte. Die ersten beiden Bits bestimmen die Bewegungsart, welche fur die Exklusiv-ODER-Bewegung 2 ist. Das dritte Bit weist das System darauf hin, ob das Sprite vom Joystick bzw. von der Tastatur gesteuert wird. Hierbei wird es so gesteuert, und Bit 3 wird so eingestellt, daß 4 dazugerechnet wird. Bit 4 bis 6 weisen das System auf den Joystick bzw. auf die Tastenreihe hin. In diesem Fall wird durch den Joystick 72 dazuaddiert. (Wenn Sie keinen Joystick haben, könnten Sie 48,6 = aufwärts 5 = abwärts R = links T = rechts, einsetzen). Letztlich wird Bit 7 eingestellt da Kollisonserfassung eingeschaltet ist, was 128 hinzufugt. Daher: Summe = 2 + 4 + 72 + 128 = 206 oder 2 + 4 + 48 + 128 = 182
2 Byte = 50	Dies ist die Nummer des Sprites, das wir bewegen.
3 Byte = 2	Erhohung in X-Richtung
4 Byte = 8	Erhohung in Y-Richtung
5. Byte = 0	Steuercode folgt
6 Byte = 0	Vom Anfang laufen lassen

Zeile 30 Wahlt SET 0

Zeile 40 POKEt die Daten in ...

Zeile 60 Setzt das Verfolgungssprite mittels SET 0 unter Inte ts in Bewegung.
Zeile 70 Endlose Schleife, um Rücksprung auf Befehlsmodus zu verhindern. Im Befehlsmodus werden die Tastencodes/Joysticksteuercodes am Bildschirm ausgegeben.

In unserem dritten Beispiel werden wir zwei Verfolgungssprites aufbauen. Das eine wird sich um Uhrzeigersinn bewegen und das andere gegen den Uhrzeigersinn.

```
5 ' BEISPIEL VON VERFOLGUNGSSPRITES III
20  DEFINT A-Z
30  !SCLS
40  UP=248:DN=8:LT=254:RT=2
50  X%=0:Y%=0
60  !SPN,42:!ISPR,@X%,@X%,@X%,@Y%:POKE Y%+5,2:Y%=Y%+6
70  C=0:R=0:S=52
80  C=RT:R=0:FOR N=1 TO 17:GOSUB 250:NEXT N
90  C=0:R=DN:FOR N=1 TO 9 :GOSUB 250:NEXT N
100 C=LT:R=0:FOR N=1 TO 17:GOSUB 250:NEXT N
110 C=0:R=UP:FOR N=1 TO 9 :GOSUB 250:NEXT N
120 C=0:R=0:S=0:GOSUB 250
130 !SPN,43:!ISPR,@X%,@X%,@X%,@Y%:POKE Y%+5,2:Y%=Y%+6
140 C=0:R=0:S=52
150 C=0:R=DN:FOR N=1 TO 9 :GOSUB 250:NEXT N
160 C=RT:R=0:FOR N=1 TO 17:GOSUB 250:NEXT N
170 C=0:R=UP:FOR N=1 TO 9 :GOSUB 250:NEXT N
180 C=LT:R=0:FOR N=1 TO 17:GOSUB 250:NEXT N
190 C=0:R=0:S=0:GOSUB 250
191 LOCATE 1,18:FOR N=1 TO 4:PRINT"      LASER BASIC VON OCEAN I.Q.":NEXT N
200 !SET,0:!COL,0 :!ROW,100:!KEY,42:!SP1,52:!TPUT,1
210 !TMOV,520,1
220 !SET,0:!COL,40:!ROW,100:!KEY,43:!SP1,52:!TPUT,1
230 !TMOV,520,1
240 END
250 POKE Y%,S:Y%=Y%+1:POKE Y%,C:Y%=Y%+1:POKE Y%,R:Y%=Y%+1:RETURN
```

Zeile 20 Gibt alle Variablen als ganzzahlig an.
Zeile 30 Löscht den Bildschirm.
Zeile 40 Stellt vier Konstanten ein, die den Bewegungen in die X- und Y-Richtung nach oben und unten, und nach links und rechts entsprechen. Achten Sie darauf, daß 248 -8 ist und 254 -1 ist.
Zeile 50 Leitet X% und Y% ein.
Zeile 60 Untersucht Sprite 42 (der Verfolger), um dessen Anfangsadresse zu finden und POKEt dann das Bewegungsartbyte mit 2 (Exklusiv-ODER-Bewegung). Der Datenzeiger (Y%) wird dann erhöht, um das erste Programmbyte anzuzeigen.
Zeile 70 Leitet C (X-Erhöhung), R (Y-Erhöhung), und S (Spritenummer) ein.
Zeile 80 POKEt in das Verfolgungssprite 17 Anweisungen und verschiebt es um 17 Zeichen nach rechts.
Zeile 90 POKEt in das Verfolgungssprite 9 Anweisungen, wodurch es um 9 Zeichen nach unten verschoben wird.
Zeile 100 Wie in Zeile 80, aber nach links.
Zeile 110 Wie in Zeile 90, aber nach oben.
Zeile 120 POKEt 3 Bytes (es werden nur die ersten beiden verwendet), damit der Verfolger zum Anfang des Verfolgungsprogramms zurückkehrt.
Zeile 130 Diese Zeile macht gemeinsam mit 140,150,160,170,180 und 190 genau das gleiche wie die Zahlen 70 bis 120. Diesmal wird aber ein Sprite, daß sich gegen den Uhrzeigersinn bewegt, aufgebaut.
Zeile 191 Druckt Text auf den Bildschirm.
Zeile 200 Setzt das erste Verfolgungsprite in Bewegung.
Zeile 210 Führt den ganzen Ablauf 10-mal durch.
Zeile 220 Setzt den zweiten Verfolger in Bewegung.
Zeile 230 Führt den zweiten Ablauf 10-mal durch.

Subroutine, die die Daten in den Verfolger POKEt. Das erste Byte ist die Spritenummer, der zweite die X-Erhöhung und das dritte die Y-Erhöhung.

Das letzte Beispiel ist eine Kombination der in den vorhergehenden Beispielen behandelten Vorfolgungssprites. Sie müssen also diese Beispiele durchführen, bevor Sie das nächste in Angriff nehmen.

```
5 ' BEISPIEL VON VERFOLGUNGSSPRITES IV
10  !SET,4
20  !KEY,51:!HGT,4:!LEN,2 :!COL,0 :!ROW,75:!SP1,46:!SP2,47:!SP3,48:!SP
30  !SET,0
40  !SCLS
50  X%=0:Y%=0
60  !SPN,44:!ISPR,@X%,@X%,@X%,@Y%
70  FOR N=0 TO 18:READ A:POKE Y%+N,A:NEXT N
80  !SPN,45:!ISPR,@X%,@X%,@X%,@Y%:TGT=Y%+3
90  FOR N=0 TO 6 :READ A:POKE Y%+N,A:NEXT N
100 !SPN,42:!ISPR,@X%,@X%,@X%,@Y%
110 POKE Y%+162,0:POKE Y%+163,1:POKE Y%+164,43
120 !SPN,43:!ISPR,@X%,@X%,@X%,@Y%
130 POKE Y%+162,0:POKE Y%+163,1:POKE Y%+164,42
140 !SPN,44:!CPUT
150 !SPN,45
160 !CMOV,1,1:IF PEEK(TGT)<>0 THEN PRINT CHR$(7):LOCATE 1,1
170 GOTO 160
180 DATA 40,0,75,46,1,0,41,15,25,50,1,0,42,0,100,52,1,0,0
190 DATA 42,0,41,0,40,0,0
```

Zeile 10 Wählt SET 4.
Zeile 20 Setzt die Information in SET 4, die vom Sprungsprite im Verfolger 40 benutzt wird.
Zeile 30 Wählt SET 0.
Zeile 40 Löscht den Bildschirm.
Zeile 50 Leitet X% und Y% ein.
Zeile 60 Untersucht Sprite 44 und teilt Y% die Anfangsadresse zu.
Zeile 70 POKEt die 19 Datenbytes, die die 3 Verfolger in Bewegung setzen. Pro bewegtes Sprite gibt es 6 Bytes (siehe CPUT) und ein Abgrenzungszeichen.
Zeile 80 Untersucht Sprite 45 und teilt Y% die Anfangsadresse zu.Die Adresse der Kollisionserfassung des zweiten Sprites (siehe CMOV) wird TGT zugeteilt.
Zeile 90 POKEt die 7 Datensprites, die die 3 Verfolger steuern.Pro gesteuertes Sprite gibt es zwei Bytes und ein Abgrenzungszeichen (siehe CMOV).
Zeile 100 Untersucht Sprite 42 und ordnet den Wert Y% zu.
Zeile 110 Stellt das Ende des Verfolgers 42 ein, so daß er nicht nur eine Schleife bildet, sondern auch zum Anfang des Verfolgers 43 springt.
Zeile 120 Untersucht Sprite 43 und ordnet den Wert Y% zu.
Zeile 130 Stellt das Ende des Verfolgers 43 ein, so daß er nicht nur eine Schleife bildet, sondern auch zum Anfang des Verfolgers 42 zurückspringt. Kontrolle wechselt daher zwischen 42 und 43.
Zeile 140 Setzt die 3 Verfolger in Bewegung.
Zeile 150 Wählt Sprite 45 (um CMOV auszuführen).
Zeile 160 Führt den Steuerbefehl aus und überprüft danach, ob der durch den Joystickgesteuerte Sprite mit einem anderen Bildschirmbild zusammengeprallt ist. Wenn dies der Fall ist, wird ein Ton erklingen.
Zeile 170 Springt auf 160 zurück.
Zeile 180 Daten für CPUT.
Zeile 190 Daten für CMOV.

Somit endet der Abschnitt über Verfolgungssprites und der Beispielprogramme "Starten mit Laser-BASIC".

SPRITEDIENSTPROGRAMME

Befehl	Vorgang
SSPR,e1,e2	Baut Sprites auf. Dieser Befehl muß jedesmal ausgeführt werden, bevor irgendwelche Spriteoperationen ausgeführt werden können. Der erste Ausdruck e1 gibt dem System den maximalen Spritewert und somit den erforderlichen Tabellenplatz an. Die Tabelle wird 4 Bytes pro Sprite und 4 Bytes für Sprite 0 verteilen. Der zweite Ausdruck e2 gibt dem System die Adresse des Spriteplatzes an, und die Tabelle sowie die Sprites werden sich von dieser Adresse nach unten erweitern. Alle vorhandenen Sprites werden vernichtet.
Parameter	Verwendung
e1	Ein BASIC-Ausdruck, welcher die maximale vom Benutzer nutzbare Spritenummer angibt. Wird eine Spritenummer mit einem Wert größer als e1 benutzt, dann wird ** SPN TOO HIGH ** (Spritenummer zu groß) gemeldet. Ist der Ausdruckswert 0, dann wird ** SPN OF ZERO ** (Spritenummer gleich null) angezeigt. Es wird in beiden Fällen keine Aktion stattfinden. e1 muß eigentlich zwischen 1 und 255 liegen.
e2	Dieser BASIC-Ausdruck gibt dem System den niedrigsten, geschützten Speicherbyte an. Sprites werden vom ersten Byte unter dieser Adresse nach unten aufgebaut.
Beispiel:	SSPR,7,&7000 ergibt eine maximale Spritenummer von 7 und der obere Bereich der Sprites gleicht 6FFF hex.
Befehl:	Vorgang
DSPR	Das Sprite, dessen Nummer in SPN enthalten ist, wird gelöscht. Die Spritetabelleneingabe und die Spritedaten werden gelöscht und der Spriteplatz wird nach oben hin verdichtet. Falls das Sprite vorher noch nicht definiert wurde, dann wird ** SPN DOESN'T EXIST ** (Spritenummer nicht vorhanden) angezeigt.
Parameter	Verwendung
SPN	Anzahl der zu löschenden Sprites.
Befehl	Vorgang
ESPR	Maximale Spritenummer erhöhen/verringern und Spriteplatz erweitern/verdichten. Ist der neue Höchstwert niedriger als der alte, dann werden alle vorhandenen Sprites mit höheren Werten als der neue Höchstwert gelöscht.
Parameter	Verwendung
SPN	Neuer maximaler Spritewert
Befehl	Vorgang
CSPR	Ein Sprite mit der Nummer in SPN und Abmessungen in HGT und LEN aufbauen. Spriteplatz erweitert sich nach unten. Ist das Sprite bereits vorhanden, dann wird ** SPN EXISTS ** (Spritenummer vorhanden) angezeigt
Parameter	Verwendung
SPN	Nummer, das zu erzeugenden Sprites.
HGT	Höhe des neuen Sprites in Bildpunkten.
LEN	Breite des neuen Sprites in Bytes.
Anmerkung:	Jedes Byte ist mit 4-Farbenmodus 4 Bildpunkte und im 16-Farbenmodus 2 Bildpunkte breit.
Befehl	Vorgang
RSPR,e1	Spriteplatz um die durch den BASIC-Ausdruck e1 gegebene Einheit verschieben. Ein positiver Wert bewegt ein Sprite in einen höheren Speicherbereich und ein negatives Vorzeichen bewegt ein Sprite zu einer niedrigeren Adresse. Dieser Befehl wird sehr selten benutzt.
Parameter	Verwendung
e1	BASIC-Ausdruck, der die Größe und die Richtung der Bewegung angibt.
Befehl	Vorgang
PSPR,@V$	Sprites werden auf Band bzw. auf Diskette aufgenommen. Der Dateiname muß in einer Kettenvariable enthalten sein und die Form "NAMESPR" HABEN: Es werden in Wirklichkeit drei Dateinamen gespeichert. Die erste Datei enthält die Systemvariablen SMAX, STAB, SPST und SPND, die die maximale Spritenummer, den Tabellenanfang, den Spritedatenanfang und das Spritedatenende enthalten. Falls "SPR" nicht die letzten drei Zeichen des Dateinamens darstellt, dann wird ** ILLEGAL FILENAME ** (ungültiger Dateiname) angezeigt. Dateinamen müssen in Großbuchstaben eingetippt werden.
Parameter	Verwendung
@V$	Die Adresse der 3-Byte-Bezeichnung für den Dateinamen unter dem die drei "NAMESYS", "NAMETAB" und "NAMESPR" gespeichert werden sollen.
Beispiel:	A$="TESTSPR" gefolgt von PSPR,@$ wird die drei folgenden Dateien erzeugen: "TESTSYS" Die laufenden Systemvariablen "TESTTAB" Die Spritetabelle "TESTSPR" Die eigentlichen Spritedaten
Befehl	Vorgang
GSPR,@V$	Holt die Sprites von Band bzw. von Diskette. Der Dateiname ist wiederum in einer Zeichenkette enthalten. Die letzten 3 Zeichen mussen "SPR" bzw. die letzten 7 Zeichen mussen "SPR.BAK" sein. Die drei Dateien werden geladen und dort positioniert, wo sie gespeichert wurden. Dateinamen mussen in Großbuchstaben eingegeben werden. Bandaufforderungen werden unterdrückt.
Parameter	Verwendung
@V$	Die Adresse der 3-Byte Bezeichnung für den Dateinamen der zu ladenden Datei.
Beispiel:	A$="TESTSPR" gefolgt von GSPR,@A$ wird diese drei Dateien laden "TESTSYS", "TESTTAB" und "TESTSPR"

MSPR,@V$	Sprites von Band bzw. Diskette werden g scht. Die Dateinamenangabe ist gleich wie für PSPR und GSPR. Die Dateitabelle, die geladen wird, wird mit der residenten Datei vermischt. Hat jedoch ein Sprite, das geladen wird, eine bereits vorhandene Spritenummer, dann liegt ein Fehler vor und es wird weiter nichts stattfinden. In diesem Fall ist der Dateiname in der Form von "NAMETAB" angegeben und muß vor einem zweiten Versuch neu geordnet werden. Nachdem die Tabellen erfolgreich vermischt wurden und die höchste, geladenen Spritenummer kleiner als der laufende Spritehöchstwert ist bzw. diesem gleicht, werden die Spritedaten geladen und der Spriteplatz wird nach unten erweitert, um das neue Sprite aufzunehmen. MSPR kann verwendet werden, um Sprites erfolgreich in eine andere Adresse, als wo sie gespeichert wurden, zu laden. Dateinamen müssen in Großbuchstaben eingetippt werden. Bandaufforderungen werden unterdrückt.
Parameter	**Verwendung**
@V$	Die Adresse der 3-Byte-Bezeichnung für das zu mischende Sprite.
Anmerkung:	Wenn der Fehler ** OUT OF MEMORY ** (Kein Speicherplatz mehr) erzeugt wird, müssen Sie MEMORY (Speicher) (Schneider BASIC) auf eine niedrigere Adresse einstellen und MSET ausführen. Dies gilt auch für GSPR und MSPR.
Befehl	**Vorgang**
RNUM	Sprite SP1 (falls vorhanden) neu numerieren, so daß es SP2 wird (falls noch nicht vorhanden). SP1 wird nicht weiter vorhanden sein.
Parameter	**Verwendung**
SP1	Sprite, das neu numeriert werden soll.
SP2	Neue Spritenummer.
Befehl	**Vorgang**
ADNM	Sämtliche Spritenummern werden um den in SPN enthaltenen Wert erhöht. Fehler werden entstehen, wenn dabei ein laufendes Sprite den maximalen Spritewert überschreitet. Der in SPN gehaltene Wert muß positiv und ungleich null sein.
Parameter	**Verwendung**
SPN	Enthält jenen Wert, um den alle laufenden Spritenummern erhöht werden.
Befehl	**Vorgang**
ISPR,@V1,@V2, @V3,@V4	Sprites werden im Detail untersucht. Folgende Variablen werden die jeweiligen Systemformationen zugeteilt bekommen: V1 Anfang der Spritetabelle (die niedrigste Adresse, die von Sprites benutzt wird). V2 Anfang der Spritedaten. V3 Ende der Sprites (die höchste Adresse, die von Sprites benutzt wird). V4 Datenadresse jenes Sprites, dessen Nummer in SPN enthalten ist.

eingestellt, dessen Nummer in SPN enthalten ist. Wenn fest stellt wird, daß das Sprite nicht vorhanden ist, dann wird zwar keine Fehlermeldung erzeugt aber SPN wird stattdessen auf null eingestellt. Die BASIC- und Grafikvariablen bleiben unverändert.

Befehl	**Vorgang**
MASK	Ein maskiertes Sprite wird von einem unmaskierten Sprite aufgebaut. Die Bewegungsbefehle FMOV und BMOV lassen Sprites auf nichtzerstörende Weise vor oder hinter Bildschirmdaten laufen. Dies ist keine Exklusiv-ODER Operation, sie bietet aber dem Benutzer eine ähnliche Funktion wie sie von Hardwaresprites geboten wird. Mit dieser Methode wird eine negative Maske von Bildpunktdaten erzeugt und ein Sprite mit abwechselnden Daten und Maskenbytes aufgebaut. Dies bedeutet, daß das angezeigte Sprite nur halb so breit ist wie das eigentliche Sprite. Jedes Sprite, das maskiert werden soll muß daher eine gleichmäßige Breite haben, da sonst ** CAN'T MASK (Maskieren nicht möglich) angezeigt wird. Vor dem Maskieren dürfen die anzuzeigenden Daten nur die linke Hälfte des Sprites belegen. Maskierte Sprites sollen nur mit den Befehlen FMOV, BMOV, FMVJ, BMVJ, FBNC, BBNC, FSWP, BPUT, BGET, RMASK, DMSK, MASK verwendet oder als Verfolgungssprites benutzt werden.
Parameter	**Verwendung**
SPN	Die Nummer des zu maskierenden Sprites.
Befehl	**Vorgang**
RMSK	Ein maskiertes Sprite wird von einem Sprite neu aufgebaut, das bereits maskiert wurde, jetzt aber neu maskiert werden muß. Dies wäre der Fall wenn ein Sprite hinter Bildschirmdaten bewegt wurde und danach wieder Bildschirmdaten bewegt werden muß.
Befehl	**Vorgang**
DMSK	Ein zuvor maskiertes Sprite wird entmaskiert und die Bildpunktdaten in der linken Hälfte werden neu aufgebaut. Die rechte Hälfte wird gelöscht.
Parameter	**Verwendung**
SPN	Die Nummer des Sprites, das entmaskiert werden soll.
Befehl	**Vorgang**
HRSP	Ein Spritepaar mit hoher Auflösung wird aufgebaut. Wenn die Software einen 160-Spalten-Modus eingestellt wird, kann die Spaltenauflösung ohne weiteres um die Hälfte reduziert werden. Dies läßt sich erzielen, indem Spaltenwert durch 2 dividiert wird und der Rest (1 oder 0) zur Spritenummer addiert wird. Dies erfolgt automatisch nach Ausführung von CLHI und setzt sich solange fort bis das System nach Anwendung von CLLO zum 80-Spalten-Modus zurückkehrt. Mit HRSP wird dann ein neues Sprite produziert, dessen Spritenummer um 1 größer als die des angegebenen Sprites ist. Dieses neue Sprite ist mit dem gegebenen Sprite identisch. Daten werden aber um ein halbes Byte gerollt (2 Bildpunkte im Farbenmodus oder 1 Bildpunkt im 16-Farbenmodus). Wenn Sprite SPN bereits vorhanden ist, wird sich ein Fehler ergeben.

MASK, RMSK, DMSK und HRSP testen vor der Ausführung das Betriebskennzeichen. Wird die Operation mit 4-Farbendaten ausgeführt, dann MUSS vor den er erwähnten Befehlen unbedingt ONHI ausgeführt werden. Wird die Operation mit 16-Farbendaten ausgeführt, dann MUSS vorher unbedingt ONLO ausgeführt werden.

Parameter	Verwendung
SPN	Die Nummer des zu paarenden Sprites.

ANMERKUNG: Das zu paarende Sprite soll in der ganz rechten Spalte keine Daten enthalten, da diese sonst in die ganze linke Spalte des Zielsprites gerollt werden.

Befehl	Vorgang
FREE,@V1	Der Umfang des freien Spriteplatzes wird getestet und das Ergebnis V1 zugeordnet. Dieser Befehl ist vorgesehen, um den Umfang des freien Spriteplatzes, der den Sprites zur Verfügung steht, zu untersuchen. Das Ergebnis wird in die Variable V1 zurückgeführt.

Parameter	Verwendung
V1	Der Betrag hinsichtlich freien Spriteplatzes wird in die ganzzahlige BASIC-Variable V1 zurückgeführt.

Befehl	Vorgang
MSET,e1	Stellt MBOT ein. Gibt dem System das niedrigste, freie Byte an, das zur Verfügung steht, und sollte jedesmal auf HIMEM + 1 eingestellt sein, wenn der BASIC-Befehl MEMORY (Speicher) ausgeführt wird.

Parameter	Verwendung
e1	Der Wert des Ausdrucks e1 wird der Systemvariable e1 zugeordnet.

PARAMETERBEZOGENE BEFEHLE

Befehl	Vorgang
SET,e1	Der Ausdruckswert von e1 wird der Grafikvariable SET zugeordnet. Der Ausdruckswert muß zwischen 0 und 15 liegen und wird verwendet, um einen der 16 Sätze von Grafikvariablen zu bestimmen.
XCL,e1	Der Ausdruckswert von e1 wird der Grafikvariable XCL zugeordnet. Der Wert muß zwischen 0 und 319 liegen und wird vom FILL-Befehl benutzt, um die X-Koordinate jenes Punktes anzugeben, an dem das Füllen beginnen soll.
IK1,e1	Der Ausdruckswert von e1 wird der Grafikvariable IK1 zugeordnet. Der Wert muß zwischen 0 und 15 liegen und wird benutzt, um die INK-Nummer für die FILL-, STCV- und SETV-Befehle einzustellen.
IK2,e1	Der Ausdruckswert von e1 wird der Grafikvariable IK2 zugeordnet. Der Wert muß zwischen 0 und 15 liegen und wird verwendet, um die INK-Nummer für den SETV-Befehl einzustellen.
COL,e1	Der Ausdruckswert von e1 wird der Grafikvariable COL zugeordnet. Der Wert muß zwischen -128 und 255 liegen und wird dazu verwendet, die Bildschirmspalte für verschiedenen Operationen zu bestimmen.

ROW,e1	Der Ausdruckswert von e1 wird der Grafikvariable ROW zugeordnet. D Wert muß zwischen -128 und 255 liegen und wird dazu benutzt, d Bildschirmreihe (in Bildpunkten) für verschiedene Operationen zu be timmen.
LEN,e1	Der Ausdruckswert von e1 wird der Grafikvariable LEN zugeordnet. Der We muß zwischen -128 und 255 liegen und wird dazu verwendet, die Breite d Bildschirmfenster, der Spritefenster, die Spriteabmessungen und die Zunahme für die MOVE-Befehle (Bewegungen) zu bestimmen.
HGT,e1	Der Ausdruckswert von e1 wird der Grafikvariable HGT zugeordnet. De Wert muß zwischen -128 und 255 liegen und wird für ähnliche Operatione wie LEN verwendet.
SPN,e1	Der Ausdruckswert von e1 wird der Grafikvariable SPN zugeordnet. Der We muß zwischen 1 und 255 liegen und wird verwendet, um bei verschiedene Anwendungsfällen die Spritenummer zu bestimmen.
SP1,e1	Der Ausdruckswert von e1 wird der Grafikvariable SP1 zugeordnet. Der Wer muß zwischen 1 und 255 liegen und wird dazu verwendet, eine von zwe Spritenummern bei Sprite-zu-Sprite-Operation zu bestimmen bzw. eine vor vier Spritenummern bei Trickabläufen anzugeben.
SP2,e1	Der Ausdruckswert von e1 wird der Grafikvariable SP2 zugeordnet. Der Wert muß zwischen 1 und 255 liegen und wird ähnlich wie SP1 benutzt.
SP3,e1	Der Ausdruckswert von e1 wird der Grafikvariable SP3 zugeordnet. Der Wert muß zwischen 1 und 255 liegen und wird als eine von vier Spritenummern in Trickabläufen verwendet.
SP4,e1	Der Ausdruckswert von e1 wird der Grafikvariable SP4 zugeordnet. Der Wert muß zwischen 1 und 255 liegen und wird als eine von vier Spritenummern in Trickabläufen verwendet.
SCL,e1	Der Ausdruckswert von e1 wird der Grafikvariable SCL zugeordnet. Der Wert muß zwischen 0 und 255 liegen und wird verwendet, um die Spalte eines Spritefensters innerhalb eines Sprites bzw. die Bildschirmspalte für das Ziel einer Rotation zu bestimmen.
SRW,e1	Der Ausdruckswert von e1 wird der Grafikvariable SRW zugeordnet. Der Wert muß zwischen 0 und 255 liegen und wird verwendet, um die Reihe eines Spritefensters innerhalb eines Sprites bzw. die Bildschirmreihe für das Ziel einer Rotation zu bestimmen.
NPX,e1	Der Ausdruckswert von e1 wird der Grafikvariable NPX zugeordnet. Der Wert muß zwischen -128 und 255 liegen und wird verwendet, um die Größe (in Bildpunkten) und die Richtung einer vertikalen Rollbewegung eines Bildschirmfensters, eines Spritefensters bzw. eines ganzen Sprites zu bestimmen.
KEY,e1	Der Ausdruckswert von e1 wird der Grafikvariable KEY zugeordnet. Der Wert muß zwischen 0 und 79 liegen und bestimmt unter anderen den KEY, nach dem durch den KBD N-Befehl abgetastet wird bzw. die Tasten-/Joystickreihe die durch die Befehle FMVJ, BMVJ, XMVJ und WMVJ abgetastet wird.
SETQ,@V1	Der Wert in der Grafikvariable SET wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.
XCLQ,@V1	Der Wert in der Grafikvariable XCL wird der BASIC-Variable V1 zugeordnet V1 muß eine

IK1Q,@V1 — Der Wert in der Grafikvariable IK1 wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

IK2Q,@V1 — Der Wert in der Grafikvariable IK2 wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

COLQ,@V1 — Der Wert in der Grafikvariable COL wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

ROWQ,@V1 — Der Wert in der Grafikvariable ROW wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

LENQ,@V1 — Der Wert in der Grafikvariable LEN wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

HGTQ,@V1 — Der Wert in der Grafikvariable HGT wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

SPNQ,@V1 — Der Wert in der Grafikvariable SPN wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

SP1Q,@V1 — Der Wert in der Grafikvariable SP1 wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

SP2Q,@V1 — Der Wert in der Grafikvariable SP2 wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

SP3Q,@V1 — Der Wert in der Grafikvariable SP3 wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

SP4Q,@V1 — Der Wert in der Grafikvariable SP4 wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

SCLQ,@V1 — Der Wert in der Grafikvariable SCL wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

SRWQ,@V1 — Der Wert in der Grafikvariable SRW wird der BASIC-Variable V1 zugewiesen. V1 muß eine ganzzahlige Variable sein.

NPXQ,@V1 — Der Wert in der Grafikvariable NPX wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

KEYQ,@V1 — Der Wert in der Grafikvariable KEY wird der BASIC-Variable V1 zugeordnet. V1 muß eine ganzzahlige Variable sein.

EXXV — Vertauscht die Vorder- und Hintergrund-SET-Variablen. Um die leistungsfähigen EVERY- und AFTER-Befehle von Locomotive BASIC zu nutzen, ist es notwendig, den laufenden Wert für SET zu speichern und den vom Interrupt benötigten Wert zuzuweisen. EXXV sollte als erster und letzter Befehl all Interrupt-Routinen ausgeführt werden.

SWPS — Vertauscht die Spritezahlen in SP2 und SP4. Dieser Befehl wird zur Verfügung gestellt, um den Bewegungsablauf umzukehren.

ASTV — Weist dem laufenden Variablen-SET Spritedaten zu. Das Sprite sollte 20 Bytes Information enthalten und muß eine Höhe von 1 und eine Länge von 20 haben, da sonst ein Parameter erzeugt wird.

Parameter	Verwendung
SPN	Die Nummer des Sprite, das die Daten für die Variablen enhält.
SET	Der Satz, dem die Daten zugewiesen werden.

AVTS — Ordnen dem Sprite, dessen Nummer in SPN gehalten wird, den laufenden Variablensatz zu. Das Sprite muß eine Höhe von 1 und eine Länge von 20 haben, da sonst ein Parameter erzeugt wird.

Parameter	Verwendung
SPN	Die Nummer des Sprite, das Daten enthalten wird.
SET	Der Variablensatz, der zugewiesen werden soll.

ESAV — Vertauscht den laufenden Variablensatz mit den Daten jenes Sprite, dessen Nummer in SPN gehalten wird. Das Sprite muß eine Höhe von 1 und eine Länge von 20 haben, da sonst ein Parameter erzeugt wird.

Parameter	Verwendung
SPN	Nummer jenes Sprite, das die zu vertauschenden Daten enthält.
SET	Der zu vertauschende Variablensatz.

SYSTEMSCHALTER

ONLO — Richtet die Hardware und Software auf 16-Farbenbetrieb ein. Dies sollte anstatt des MODE-Befehls von BASIC ausgeführt werden.

ONHI — Stellt die Hardware auf 4-Farbenmodus ein. Dies ersetzt evenfalls den MODE-Befehl von BASIC.

CLLO — Bringt die Software auf 80-Spaltemmodus. Bei diesem Modus ist Spalte 79 die extrem rechte Spalte.

CLHI — Bringt die Software auf 160-Spaltenmodus. In diesem wird der Bildschirm von allen GETs und PUTs als 160 Spalten breit behandelt und Spalte 159 ist die extrem rechte Spalte. Achten Sie aber darauf, daß der Bildschirm für alle anderen Zwecke immer noch als 80 Spalten breit behandelt wird, insbesondere für Durchlauf definierte Fenster, Spiegeln usw., werden den Bildschirm weiterhin als 80 Spalten breit betrachten.

WICHTIGER HINWEIS:

Wenn Operationen auf Sprites bzw. Spritefenster ausgeführt werden, werden die ONHI- und ONLO-Befehle benötigt, um das System darauf hinzuweisen, daß der Modus VOR Durchführung der Operation geändert wurde. Wenn z. B. ein 16-farbiger Sprite MASKiert ist und der zuletzt benutzte Schalter ONHI war, dann wird das Sprite verstümmelt.

GRUPPE 1 GETs und PUTs

GETs und PUTs der Gruppe 1 haben ein GT bzw. ein PT vorgestellt. GET.Befehle benutzen ein Bildschirmfenster als deren Datenquelle und ein Sprite als deren Bestimmungsort. PUTs verwenden ein Sprite als Datenquelle und den Bildschirm als Bestimmungsort. Jeder Befehl hat eine von sechs Nachsilben.

BL — Daten werden blockweise von der Quelle zum Bestimmungsort bewegt und ersetzen die Daten, die vorher am Bestimmungsort enthalten waren.

OR — Daten von der Quelle werden mit einem OR mit den laufenden Daten am Bestimmungsort verknüpft.

verknüpft.

XR Daten von der Quelle werden mit einem XOR mit de laufenden Daten am Bestimmungsort verknüpft.

BH Daten von der Quelle werden hinter Daten am Bestimmungsort gesetzt.

IF Daten von der Quelle werden vor Daten am Bestimmungsort gesetzt.

In allen Fällen wird SPN verwendet, um das Sprite zu bestimmen und COL und ROW definieren die obere linke Ecke des Spritefensters. Die Fenstermaße entsprechen den Spriteabmessungen. Wenn der Bildschirmrand vom Fenster überlappt wird, findet die Operation am "ON-Screen"-Teil (am Bildschirm befindlichen Teil) des Fensters statt. Befindet sich das ganze Fenster aus dem Bildschirmbereich, dann wird keine Operation stattfinden. Es werden allerdings keine Fehler erzeugt.

Für jeden der 12 Befehle aus Gruppe 1:

Parameter	Verwendung
SPN	Die Nummer des zu verwendenden Sprite.
COL	Die Bildschirmspalte des Fensters.
ROW	Die Bildschirmreihe des Fensters.
@V1	Wird bei der Kollisionserfassung verwendet. (s. nächsten Abschnitt).

Befehl	Vorgang
GTBL	Spritefenster wird blockweise in das Sprite bewegt.
GTOR	Bildschirmfenster wird mittels OR mit Sprite verknüpft.
GTND	Bildschirmfenster wird mittels AND mit Sprite verknüpft.
GTXR	Bildschirmfenster wird mittels XOR mit Sprite verknüpft.
GTBH	Bringt die Daten des Bildschirmfensters hinter Spritedaten.
GTIF	Bringt die Daten des Bildschirmfensters vor Spritedaten.
PTBL	Sprite wird blockweise in das Bildschirmfenster bewegt.
PTOR	Sprite wird mittels OR mit Bildschirmfenster verknüpft.
PTND	Sprite wird mittels AND mit Bildschirmfenster verknüpft.
PTXR	Sprite wird mittels XOR mit Bildschirmfenster verknüpft.
PTBH	Spritedaten werden hinter die Bildschirmfensterdaten gesetzt.
PTIF	Spritedaten werden vor die Bildschirmfensterdaten gesetzt.

Kollisionserfassung

Alle 12 Befehle in dieser Gruppe können mit oder ohne Kollisionserfassung ausgeführt werden. Falls einer der Befehle ohne einen nachgestellten Parameter ausgeführt wird, dann wird die Erfassung mit "OFF" (AUS) angezeigt. Wird ein Befehl mit der Adresse einer ganzzahligen Variable als Einzeloperand ausgeführt, dann wird die Erfassung mit "ON" (EIN) angezeigt. Kollisionserfassung verlangsamt die Ausführungsgeschwindigkeit und sollte nur wenn notwendig verwendet werden. Falls die Quellendaten mit den Zieldaten kollidieren, wird der Wert in der BASIC-Variable erhöht, falls nicht, bleibt er unverändert.

Beispiel: Falls der Wert von X% auf 7 eingestellt ist und der Befehl PTBH,@X% ausgeführt wird, dann wird X% bei einer Kollision zwischen Sprite- und Bildschirmdaten auf 8 erhöht.

GRUPPE II GETs und PUTs

GETs und PUTs der Gruppe II haben ein GW bzw. ein PW vorgestellt. Die GET-Befehle verwenden ein Bildschirmfenster als deren Datenquelle und ein Spritefenster als deren Bestimmungsort. Die PUTs benutzen ein Spritefenster als deren Datenquellen und ein Bildschirmfenster als deren Bestimmungsort. Jeder Befehl hat eine von sechs Nachsilben:

BL Daten werden blockweise von der Quelle zum Bestimmungsort bewegt und ersetzen c Daten, die vorher am Bestimmungsort gehalten wurden.

OR Daten von der Quelle werden mittels OR mit den laufenden Daten am Bestimmungsc verknüpft.

ND Daten von der Quelle werden mittels AND mit den laufenden Daten am Bestimmungc verknüpft.

XR Daten von der Quelle werden mittels XOR mit den laufenden Daten am Bestimmungsc verknüpft.

BH Daten von der Quelle werden hinter die Daten am Bestimmungsort gesetzt.

IF Daten von der Quelle werden vor die Daten am Bestimmungsort gesetzt.

In allen Fällen wird SPN verwendet, um das Sprite zu bestimmen; SCL und SRW bestimmen d Spalte und die Reihe des Spritefensters innerhalb des Sprite; COL und ROW bestimmen die Spal und Reihe des Bildschirmfenster und HGT und LEN die Fenstermaße. Die Fensterabmessunge werden verkleinert, wenn es teilweise außerhalb des Sprite ("off-Sprite") bzw. außerhalb de Fensters ("off-window") liegt. Ist das Fenster gänzlich "off-Sprite" bzw. "off-window", dann wird keir Operation stattfinden. Es wird allerdings keine Fehlermeldung erzeugt.

Für jeden der 12 Befehle aus Gruppe II:

Parameter	Verwendung
SPN	Die Nummer des zu verwendenden Sprite.
SCL	Die Spritespalte des Fensters.
SRW	Die Spritereihe des Fensters.
COL	Die Bildschirmspalte des Fensters.
ROW	Die Bildschirmreihe des Fensters.
LEN	Die Breite des Fensters.
HGT	Die Höhe des Fensters.
@V1	Wird bei der Kollisionserfassung verwendet.

Befehl	Vorgang
GWBL	Spritefenster wird blockweise in das Spritefenster bewegt.
GWOR	Bildschirmfenster wird mittels OR mit Spritefenster verknüpft.
GWND	Bildschirmfenster wird mittels AND mit Spritefenster verknüpft.
GWXR	Bildschirmfenster wird mittels XOR mit Spritefenster verknüpft.
GWBH	Daten des Bildschirmfensters werden hinter die Daten des Spritefenster gesetzt.
GWIF	Daten des Bildschirmfensters werden vor die Daten des Spritefenster gesetzt.
PWBL	Spritefenster wird blockweise in das Bildschirmfenster bewegt.
PWOR	Spritefenster wird mittels OR mit Bildschirmfenster verknüpft.
PWND	Spritefenster wird mittels AND mit Bildschirmfenster verknüpft.
PWXR	Spritefenster wird mittels XOR mit Bildschirmfenster verknüpft.
PWBH	Daten des Spritefenster werden hinter die Daten des Bildschirmfenster gesetzt.
PWIF	Daten des Spritefenster werden vor die Daten des Bildschirmfenster gesetzt.

Kollisionserfassung

Kollisionserfassung für Befehle der Gruppe II funktioniert in der gleichen Art und Weise wie bei der vorhin beschriebenen Befehlen der Gruppe I. Ist dem Befehl kein Parameter nachgestellt, dann is die Erfassung ausgeschaltet. Ist dem Befehl eine ganzzahlige Variablenadresse nachgestellt, dann wird die Variable erhöht, falls eine Kollision stattfindet, falls nicht, bleibt sie unverändert. Kollisio verzögert auch hier die Ausführung des Befehl

GETs und PUTs der Gruppe III haben ein GM bzw. ein PM vorgestellt. Die GET-Befehle benutzen ein Spritefenster als Datenquelle und ein ganzes Sprite als deren Bestimmungsort. Die PUTs verwenden ein ganzes Sprite als Datenquelle und ein Spritefenster als Bestimmungsort. Jeder Befehl hat eine von seches Nachsilben:

BL Daten werden blockweise von der Quelle zum Bestimmungsort bewegt und ersetzen dabei die Daten, die vorher am Bestimmungsort enthalten waren.

OR Daten von der Quelle werden mittels OR mit den laufenden Daten am Bestimmungsort verknüpft.

ND Daten von der Quelle werden mittels AND mit den laufenden Daten am Bestimmungsort verknüpft.

XR Daten von der Quelle werden mittels XOR mit den laufenden Daten am Bestimmungsort verknüpft.

BH Daten von der Quelle werden hinter die Date am Bestimmungsort gesetzt.

IF Daten von der Quelle werden vor die Date am Bestimmungsort gesetzt.

In allen Fällen wird SP1 für die Bestimmung des ganzen Sprite verwendet und SP2 für das Sprite, das das Fenster enthält. SCL und SRW bestimmen die Spalte und Reihe des Fensters in Sprite SP2 und die Fenstermaße entsprechen den Abmessungen von Sprite SP1. Die Fensterabmessungen werden verkleinert wenn SP1 durch die Werte von SCL und SRW die Grenzen von Sprite SP2 überlappt. Falls SP1 gänzlich "off-Sprite" (außerhalb des Sprite) liegt, dann wird keine Operation stattfinden. Es wird allerdings auch keine Fehlermeldung erzeugt.

Für jeden der 12 Befehle der Gruppe III:

Parameter	Verwendung
SP1	Die Nummer des ganzen Sprite.
SP2	Die Nummer des Sprite, das das Spritefenster enthält.
SCL	Spalte des Spritefensters in Sprite SP2.
SRW	Reihe des Spritefensters in Sprite SP2
@V1	Wird bei der Kollisionserfassung verwendet.

Befehl	Vorgang
GMBL	Spritefenster wird blockweise in das ganze Sprite bewegt.
GMOR	Spritefenster wird mittels OR mit dem ganzen Sprite verknüpft.
GMND	Spritefenster wird mittels AND mit dem ganzen Sprite verknüpft.
GMXR	Spritefenster wird mittels XOR mit dem ganzen Sprite verknüpft.
GMBH	Daten des ganzen Sprite werden hinter die Daten des Spritefensters gesetzt.
GMIF	Daten des ganzen Sprite werden vor die Daten des Spritefensters gesetzt.
PMBL	Das ganze Sprite wird blockweise in das Spritefenster bewegt.
PMOR	Das ganze Sprite wird mittels OR mit dem Spritefenster verknüpft.
PMND	Das ganze Sprite wird mittels AND mit dem Spritefenster verknüpft.
PMXR	Das ganze Sprite wird mittels XOR mit dem Spritefenster verknupft.
PMBH	Daten des Spritefensters werden hinter die Daten des ganzen Sprite gesetzt.

PMIF Daten des Spritefensters werden vor die Daten des ganzen Sprit gesetzt.

Kollisionserfassung

Kollisionserfassung für die Befehle der Gruppe III funktioniert in der gleichen Art und Weise wie fü die vorhin beschriebenen Befehle der Gruppe I und II. Wenn dem Befehl kein Parameter nachgestel ist, dann ist die Erfassung ausgeschaltet. Ist dem Befehl eine ganzzahlige Variablenadress nachgestellt, dann wird die Variable erhöht, wenn eine Kollision stattfindet, falls nicht, bleibt si unverändert. Die Erfassung wird auch hier die Ausführungsgeschwindigkeit verzögern.

WICHTIGER HINWEIS:

Achten Sie darauf, daß ONLO und ONHI verwendet werden, um Laser BASIC auf den richtige Modus einzustellen, bevor Sie 'GM' – bzw. 'PM'-Befehle verwenden.

GRUPPE I SCROLLS UND WRAPS (Rollen und Umlauf)

Scrolls und Wraps der Gruppe I haben SV bzw. WV vorgestellt. Die Scroll-Befehle rollen Daten ohn Umlauf und die Wraps rollen Daten mit Umlauf. Jeder Befehl verfügt über eine von sechs vorsilben

R1 Daten bewegen sich um 1 Bildpunkt nach rechts. Dies wäre im 4-Farbenmodus 1/32 der Bildschirmbreite und im 16-Farbenmodus 1/160 der Bildschirmbreite.

L1 Daten bewegen sich um 1 Bildpunkt nach links. Im 4-Farbenmodus wäre dies 1/320 de Bildschirmbreite und im 16-Farbenmodus wäre dies 1/160 der Bildschirmbreite.

R4 Daten bewegen sich um 1 Byte nach rechts. Dies wäre bei beiden Betriebsarten 1/8 der Bildschirmbreite. Im 4-Farbenmodus entspricht dies einer Bewegung um Bildpunkte und im 16-Farbenmodus einer Bewegung um 2 Bildpunkte.

L4 Daten bewegen sich um 1 Byte nach links. Dies wäre bei beiden Betriebsarten 1/80 de Bildschirmbreite. Im 4-Farbenmodus entspricht dies einer Bewegung um 4 Bildpunkt und im 16-Farbenmodus einer Bewegung um 2 Bildpunkte.

R8 Daten bewegen sich um 2 Bytes nach rechts. Dies wäre bei beiden Betriebsarten 1/4 der Bildschirmbreite. Im 4-Farbenmodus entspricht dies einer Bewegung um Bildpunkte und im 16-Farbenmodus einer Bewegung um 4 Bildpunkte.

L8 Daten bewegen sich um 2 Bytes nach links. Dies wäre bei beiden Betriebsarten 1/4 der Bildschirmbreite. Im 4-Farbenmodus entspricht dies einer Bewegung um Bildpunkte und im 16-Farbenmodus einer Bewegung um 4 Bildpunkte.

In allen Fällen bestimmen COL und ROW die Spalte und Reihe der oberen linken Ecke de Bildschirmfensters und HGT und LEN bestimmen die Fenstermaße. Die Fenstermaße werde verstellt, wenn das Fenster teilweise außerhalb des Bildschirms ("off-screen") liegt. Liegt das Fenste gänzlich außerhalb des Bildschirms, dann wird keine Operation stattfinden. Es wird allerdings auch keine Fehlermeldung erzeugt.

Für jeden der 12 Befehle in Grupp I

Parameter	Verwendung
COL	Spalte des Bildschirmfensters
ROW	Reihe des Bildschirmfensters
LEN	Breite des Bildschirmfensters.
HGT	Höhe des Bildschirmfensters.
e1	Optionaler Parameter zur Bestimmung der Ausführungshäufigkeit.
e2	Optionaler Parameter zur Bestimmung des Rahmensynchronisierstatus.

SVR1	Bildschirmfenster wird nach rechts gerollt,	1 Bildpunkt,	kein Umlauf.
SVL1	Bildschirmfenster wird nach links gerollt,	1 Bildpunkt,	kein Umlauf.
SVR4	Bildschirmfenster wird nach rechts gerollt,	1 Byte,	kein Umlauf.
SVL4	Bildschirmfenster wird nach links gerollt,	1 Byte,	kein Umlauf.
SVR8	Bildschirmfenster wird nach rechts gerollt,	2 Bytes,	kein Umlauf.
SVL8	Bildschirmfenster wird nach links gerollt,	2 Bytes,	kein Umlauf.
WVR1	Bildschirmfenster wird nach rechts gerollt,	1 Bildpunkt,	mit Umlauf.
WVL1	Bildschirmfenster wird nach links gerollt,	1 Bildpunkt,	mit Umlauf.
WVR4	Bildschirmfenster wird nach rechts gerollt,	1 Byte,	mit Umlauf.
WVL4	Bildschirmfenster wird nach links gerollt,	1 Byte,	mit Umlauf.
WVR8	Bildschirmfenster wird nach rechts gerollt,	2 Bytes,	mit Umlauf.
WVL8	Bildschirmfenster wird nach links gerollt,	2 Bytes,	mit Umlauf.

Wiederholte Ausführung

Werden Scroll- bzw. Wrap-Befehle der Gruppe I (oder aber auch Scrolls und Wraps der Gruppe II und III) ohne nachgestellte Parameter ausgeführt, dann wird der Befehl nur einmal ausgeführt. Sind dem Befehl jedoch 2 Ausdrücke nachgestellt, dann wird er mehrmals ausgeführt. Der erste Ausdruck gibt die Anzahl der Befehlsausführungen an. Der zweite Parameter bestimmt, ob der Befehl und der Rahmenrücklauf synchronisiert sind oder nicht. Ist der Wert des Zweiten Parameters null, dann wird der Befehl ohne Verzögerung mehrmals ausgeführt. Ein Wert ungleich null bewirkt, daß die Ausführung 50 mal pro Sekunde beginnt (oder je nach Ausführungszeit Vielfache von 1/50 Sekundenperioden) und somit eine viel glattere Bewegung erzeugt wird. Sind dem Befehl 1 bzw. mehr als 2 Parameter nachgestellt, dann wird ** PARAMETER ERROR** (Parameterfehler) angezeigt.

GRUPPE II SCROLLS UND WRAPS (Rollen und Umlauf)

Scrolls und Wraps der Gruppe II haben SS bzw. WS vorgestellt. Die Scroll-Befehle rollen Daten ohne Umlauf und die Wraps rollen Daten mit Umlauf. Jeder Befehl verfügt über eine von sechs Vorsilben:

R1 Daten werden um 1 Bildpunkt nach rechts bewegt. Dies entspricht im 4-Farbenmodus 1/320 der Bildschirmbreite und im 16-Farbenmodus 1/160 der Bildschirmbreite.

L1 Daten werden um 1 Bildpunkt nach links bewegt. Dies entspricht im 4-Farbenmodus 1/320 der Bildschirmbreite und im 16-Farbenmodus 1/160 der Bildschirmbreite.

R4 Daten werden um 1 Byte nach rechts bewegt. Dies entspricht bei beiden Betriebsarten 1/80 der Bildschirmbreite. Im 4-Farbenmodus wären dies 4 Bildpunkte und im 16-Farbenmodus wäre dies eine Bewegung von 2 Bildpunkten.

L4 Daten werden um 1 Byte nach links bewegt. Dies entspricht bei beiden Betriebsarten 1/80 der Bildschirmbreite. Im 4-Farbenmodus wären dies 4 Bildpunkte und im 16-Farbenmodus wäre dies eine Bewegung von 2 Bildpunkten.

R8 Daten werden um 2 Bytes nach rechts bewegt. Dies entspricht bei beiden Betriebsarten 1/40 der Bildschirmbreite. Im 4-Farbenmodus wären dies 8 Bildpunkte und im 16-Farbenmodus wäre dies eine Bewegung von 4 Bildpunkten.

L8 Daten werden um 2 Bytes nach links bewegt. Dies entspricht bei beiden Betriebsarten 1/40 der Bildschirmbreite. Im 4-Farbenmodus wären dies 8 Bildpunkte und im 16-Farbenmodus wäre dies eine Bewegung von 4 Bildpunkten.

SPN bestimmt in jedem Fall das Sprite, das gerollt werden soll. Für jeden der 12 Befehle in Gruppe II:

Parameter	Verwendung
SPN	Das Sprite, das mit oder ohne Umlauf gerollt werden soll

Befehl	Vorgang		
SSR1	te wird nach rechts gerollt,	1 Bildpunkt,	kein Umlau
SSL1	Sprite wird nach links gerollt,	1 Bildpunkt,	kein Umlau
SSR4	Sprite wird nach rechts gerollt,	1 Byte,	kein Umlau
SSL4	Sprite wird nach links gerollt,	1 Byte,	kein Umlau
SSR8	Sprite wird nach rechts gerollt,	2 Bytes,	kein Umlau
SSL8	Sprite wird nach links gerollt,	2 Bytes,	kein Umlaul
WSR1	Sprite wird nach rechts gerollt,	1 Bildpunkt,	mit Umlaul
WSL1	Sprite wird nach links gerollt,	1 Bildpunkt,	mit Umlauf
WSR4	Sprite wird nach rechts gerollt,	1 Byte,	mit Umlauf
WSL4	Sprite wird nach links gerollt,	1 Byte,	mit Umlauf
WSR8	Sprite wird nach rechts gerollt,	2 Bytes,	mit Umlauf
WSL8	Sprite wird nach links gerollt,	2 Bytes,	mit Umlauf

Wiederholte Ausführung

Scrolls und Wraps der Gruppe II können genauso wiederholt werden wie die vorhin beschriebenen wiederholten Ausführungen für die Gruppe I.

GRUPPE III SCROLLS UND WRAPS

Scrolls und Wraps der Gruppe III haben SP bzw. WP vorgestellt. Die Scroll-Befehle rollen Daten ohne Umlauf und die Wraps rollen Daten mit Umlauf. Jeder Befehl hat eine von sechs Nachsilben:

R1 Daten werden um 1 Bildpunkt nach rechts bewegt. Dies entspricht im 4-Farbenmodus 1/320 der Bildschirmbreite und im 16-Farbenmodus 1/160 der Bildschirmbreite.

L1 Daten werden um 1 Bildpunkt nach links gerollt. Dies entspricht im 4-Farbenmodus 1/320 der Bildschirmbreite und im 16-Farbenmodus 1/160 der Bildschirmbreite.

R4 Daten werden um 1 Byte nach rechts bewegt. Dies entspricht bei beiden Betriebsarten 1/80 der Bildschirmbreite. Im 4/Farbenmodus wäre dies eine Bewegung um 4 Bildpunkte und im 16/Farbenmodus eine Bewegung um 2 Bildpunkte.

L4 Daten werden um 1 Byte nach links bewegt. Dies entspricht bei beiden Betriebsarten 1/80 der Bildschirmbreite. Im 4-Farbenmodus wäre dies eine Bewegung um 4 Bildpunkte und im 16-Farbenmodus eine Bewegung um 2 Bildpunkte.

R8 Daten werden um 2 Bytes nach rechts bewegt. Dies entspricht bei beiden Betriebsarten 1/40 der Bildschirmbreite. Im 4-Farbenmodus wäre dies eine Bewegung um 8 Bildpunkte und im 16-Farbenmodus eine Bewegung um 4 Bildpunkte.

L8 Daten werden um 2 Bytes nach links bewegt. Dies entspricht bei beiden Betriebsarten 1/40 der Bildschirmbreite. Im 4-Farbenmodus wäre dies eine Bewegung um 8 Bildpunkte und im 16-Farbenmodus eine Bewegung um 4 Bildpunkte.

SPN bestimmt in jedem Fall das Sprite, das das zu rollende Fenster enthält. COL und ROW enthalten die Spalte und Reihe der linken oberen Ecke des Fensters innerhalb des Sprite, und LEN und HGT bestimmen die Fenstermaße. Falls COL und ROW so gewählt werden, daß das Fenster teilweise außerhalb des Sprite ("off-sprite") liegt, dann werden die Fensterabmessungen entsprechend verstellt. Liegt das Fenster gänzlich "off-Sprite", dann wird keine Operation stattfinden. Es wird allerdings keine Fehlermeldung erzeugt.

Parameter	Verwendung
SPN	Nummer des Sprite, das das Fenster enthält.
COL	Spritespalte des Fensters.
ROW	Spritereihe des Fensters.
LEN	Breite des Fensters.
HGT	Höhe des Fensters.

SPR1	Spritefenster wird nach rechts gerollt,	1 Bildpunkt,	kein Umlauf.
SPL1	Spritefenster wird nach links gerollt,	1 Bildpunkt,	kein Umlauf.
SPR4	Spritefenster wird nach rechts gerollt,	1 Byte,	kein Umlauf.
SPL4	Spritefenster wird nach links gerollt,	1 Byte,	kein Umlauf.
SPR8	Spritefenster wird nach rechts gerollt,	2 Bytes,	kein Umlauf.
SPL8	Spritefenster wird nach links gerollt,	2 Bytes,	kein Umlauf.
WPR1	Spritefenster wird nach rechts gerollt,	1 Bildpunkt,	mit Umlauf.
WPL1	Spritefenster wird nach links gerollt,	1 Bildpunkt,	mit Umlauf.
WPR4	Spritefenster wird nach rechts gerollt,	1 Byte,	mit Umlauf.
WPL4	Spritefenster wird nach links gerollt,	1 Byte,	mit Umlauf.
WPR8	Spritefenster wird nach rechts gerollt,	2 Bytes,	mit Umlauf.
WPL8	Spritefenster wird nach links gerollt,	2 Bytes,	mit Umlauf.

Wiederholte Ausführung

Scrolls und Wraps der Gruppe III können genauso wiederholt werden wie die vorhin beschriebenen wiederholten Ausführungen der Gruppe I und II.

GRUPPE IV SCROLLS UND WRAPS (Rollen und Umlauf)

Dies sind vertikale Scrolls und Wraps, denen ein VN (Vertikal-NPX) vorgestellt ist. Die Scroll-Befehle rollen ohne Umlauf und haben ein S vorgestellt, und die Wraps rollen mit Umlauf und haben ein W vorgestellt. Es gibt drei verschiedene Typen (durch das zweite Zeichen im Namen bestimmt). Diese sind:

Typ V

Rollen erfolgt auf einem Bildschirmfenster und COL und ROW bestimmen die obere linke Ecke des Fensters, HGT und LEN bestimmen die Fensterabmessungen. Die Fensterparameter werden verstellt, wenn das Fenster teilweise außerhalb des Bildschirms ("off-screen") liegt.

Typ S

Das Rollen erfolgt auf dem ganzen Sprite, dessen Nummer in SPN enthalten ist.

Typ P

Das Rollen erfolgt auf einem Spritefenster. Die Nummer des Sprite wird durch SPN angegeben und das Fenster wird durch COL, ROW, HGT und LEN bestimmt. Das Fenster wird verstellt, wenn es teilweise außerhalb des Sprite ("off-sprite") liegt.

Befehl	Vorgang
SVVN	Das Bildschirmfenster wird vertikal um NPX Bildpunkte ohne Umlauf gerollt.
WVVN	Das Bildschirmfenster wird mit Umlauf um NPX Bildpunkte vertikal gerollt.

Parameter	Verwendung
COL	Bildschirmspalte der linken oberen Fensterecke.
ROW	Bildschirmreihe der linken oberen Fensterecke.
LEN	Breite des Bildschirmfensters.
HGT	Höhe des Bildschirmfensters.
NPX	Größe und Richtung der Scroll-Bewegung in Bildpunkten (1/200 der Bildschirmhöhe). Positive Werte bewirken eine Aufwärtsbewegung und negative Werte eine Abwärtsbewegung.

Befehl	Vorgang
SSVN	Das Sprite wird ohne Umlauf um NPX Bildpunkte vertikal gerollt.
WSVN	Das Sprite wird mit Umlauf um NPX Bildpunkte vertikal gerollt.

Parameter	Verwendung
SPN	Nummer des zu rollenden Sprite.
NPX	Größe und Richtung der Scroll-Bewegung in Bildpunkten.

Befehl	Vorgang
SPVN	Das Spritefenster wird ohne Umlauf um NPX Bildpunkte vertikal gerollt.
WPVN	Das Spritefenster wird mit Umlauf um NPX Bildpunkte vertikal gerollt.

Parameter	Verwendung
SPN	Nummer des zu rollenden Sprite.
COL	Spritespalte der oberen linken Fensterecke.
ROW	Spritereihe der oberen linken Fensterecke.
LEN	Breite des Spritefensters.
HGT	Höhe des Spritefensters.
NPX	Größe und Richtung der Scroll-Bewegung in Bildpunkten.

Wiederholte Ausführung.

Scrolls und Wraps der Gruppe IV können genauso wiederholt werden wie die vorhin beschriebene Befehle der Gruppe I, II und III.

WICHTIGER HINWEIS:

ONLO und ONHI müssen ausgeführt werden, bevor irgendwelche Operationen auf den Sprites bzw Spritefenstern ausgeführt werden.

TRANSFORMATIONEN

In diesem Paket gibt es eine Reihe von Befehlen, die Transformationen an Sprites, Spritefensterr bzw. Bildschirmfenstern ausführe. Nur das Fenster bzw. das Sprite selbst sind davon betroffen Zwischen dem Fenster bzw. dem Sprite, die transformiert werden, und anderen Fenstern oder Sprites findet kein Datenfluß statt. Es gibt drei Datentypen (durch die Befehlsnachsilbe angegeben). Diese sind:

Typ V

Transformationen erfolgen auf einem Bildschirmfenster, dessen obere linke Ecke von COL und ROW bestimmt wird und dessen Abmessung von HGT und LEN angegeben wird. Die Fensterparameter werden verstellt, wenn das Fenster teilweise außerhalb des Bildschirms ("off-screen") liegt.

Typ S

Die Transformation erfolgt auf dem ganzen Sprite, dessen Nummer in SPN gehalten wird.

Typ P

Die Transformation erfolgt auf einem Spritefenster. Die Nummer des Sprite wird durch SPN angegeben und das Fenster wird durch COL, ROW, HGT und LEN bestimmt. Das Fenster wird verstellt, wenn es teilweise außerhalb des Sprite ("off-sprite") liegt.

Transformationen der Gruppe I gehören alle zum Type V und werden auf Bildschirmfenstern ausgeführt. Jedem Befehl ist ein V vorgestellt. Für die folgenden Befehle der Gruppe I:

Parameter	Verwendung
COL	Bildschirmspalte der linken oberen Fensterecke.
ROW	Bildschirmreihe der oberen linken Fensterecke.
LEN	Breite des Fensters.
HGT	Höhe des Fensters.

Befehl	Vorgang
CLSV	Fenster wird auf Papierfarbe (INK O) eingestellt.
MGXV	X-Expansion der linken Hälfte des Bildschirmfensters in das ganze Bildschirmfenster.
MGYV	Y-Expansion der oberen Hälfte des Bildschirmfensters in das ganze Bildschirmfenster.
MIRV	Bildschirmfenster wird um dessen vertikale Mittellinie gespiegelt.
MORV	Linke Hälfte des Bildschirmsfensters wird in die rechte Hälfte hineingespiegelt (erzeugt ein in horizontaler Richtung symmetrisches Fenster).
FIPV	Bildschirmfenster wird um dessen horizontale Mittellinie gespiegelt.
FOPV	Obere Hälfte des Bildschirmfensters wird in die untere Hälfte hineingespiegelt (erzeugt ein in vertikaler Richtung symmetrisches Fenster).
INVV	Invertiert (1er Komplement) sämtliche Bildpunktdaten im Bildschirmfenster.

Für die folgenden Befehle der Gruppe I:

Parameter	Verwendung
COL	Bildschirmbreite der linken oberen Fensterecke.
ROW	Bildschirmreihe der linken oberen Fensterecke.
LEN	Breite des Fensters.
HGT	Höhe des Fensters.
IK1	Erste INK-Farbe.
IK2	Zweite INK-Farbe.

Befehl	Vorgang
STCV	Bildschirmfenster wird mit INK gefüllt, dessen Nummer in IK1 gehalten wird.
SETV	Farbänderung einer Fenstergrafik. Ersetzt jeden Bildpunkt, dessen Farbnummer in IK1 gehalten wird, durch einen Bildpunkt, dessen Farbnummer in IK2 gehalten wird.

GRUPPE II – TRANSFORMATIONEN

Transformationen der Gruppe II gehören alle zum Type S und werden auf einem ganzen Sprite ausgeführt und jedem Befehl ist ein S vorgestellt.

Für die folgenden Befehle der Gruppe II:

Parameter	Verwendung
SPN	Nummer des Sprite, das transformiert werden soll.

Befehl	Vorgang
CLSS	Stellt Sprite auf Papierfarbe (INK O) ein.
MGXS	X-Expansion der linken Spritehälfte in das ganze Sprite.
MIRS	Sprite wird um dessen vertikale Mittellinie gespiegelt.
MORS	Linke Spritehälfte wird in die rechte Spritehälfte hineingespiegelt (erzeugt ein in horizontaler Richtung symmetrisches Sprite)
FIPS	Sprite wird um dessen horizontale Mittellinie gespiegelt.
FOPS	Obere Spritehälfte wird in die untere Spritehälfte hineingespiegelt (erzeugt ein in vertikaler Richtung symmetrisches Sprite).

Für die folgenden Befehle der Gruppe II:

Parameter	Verwendung
SPN	Nummer des Sprite, das transformiert werden soll.
IK1	Erste INK-Nummer.
IK2	Zweite INK-Nummer.

Befehl	Vorgang
STCS	Das ganze Sprite wird mit dem INK gefüllt, dessen Nummer in IK1 gehalten wird.
SETS	Farbänderung einer Spritegrafik. Ersetzt jeden Bildpunkt, dessen Farbnummer in IK1 gehalten wird, durch einen Bildpunkt, dessen Farbnummer in IK2 gehalten wird.

GRUPPE III – TRANSFORMATIONEN

Transformationen der Gruppe III gehören alle zur Type P und werden auf einem Spritefenster ausgeführt. Jeder Befehl hat ein P vorgestellt.

Für die folgenden Befehle der Gruppe III:

Parameter	Verwendung
SPN	Nummer des Sprite, das transformiert werden soll.
COL	Spritespalte der oberen linken Fensterecke.
ROW	Spritereihe der oberen linken Fensterecke.
LEN	Breite des Spritefensters.
HGT	Höhe des Spritefensters.

Befehl	Vorgang
CLSP	Spritefenster wird auf Papierfarbe (INK O) eingestellt.
MGXP	X-Expansion der linken Hälfte des Spritefensters in das ganze Fenster.
MIRP	Spritefenster wird um dessen vertikale Mittellinie gespiegelt.
MORP	Linke Hälfte des Spritefensters wird in die rechte Hälfte hineingespiegelt (erzeugt ein in horizontaler Richtung symmetrisches Spritefenster).
INVP	Sämtliche Bildpunkte im Spritefenster werden invertiert (1er Komplement).

Für die folgenden Befehle der Gruppe III:

Parameter	Verwendung
SPN	Nummer des Sprite, das transformiert werden soll.
COL	Spritespalte der linken oberen Fensterecke.
ROW	Spritereihe der linken oberen Fensterecke.
LEN	Breite des Spritefensters.
HGT	Höhe des Spritefensters.
IK1	Erste INK-Nummer.
IK2	Zweite INK-Nummer.

Befehl	Vorgang
STCP	Spritefenster wird mit dem INK gefüllt, dessen Nummer in IK1 gehalten

WICHTIGER HINWEIS:

ONLO und ONHI müssen korrekt ausgeführt werden, bevor irgendeine Operation auf Sprites bzw. Spritefenstern ausgeführt wird.

DATENAUSTAUSCH

Es sind drei Datenaustauschbefehle vorgesehen, die unter Verwendung von MASKierten Sprites Daten hinter oder vor Bildschirmdaten setzen (PUT) und die die Bildschirmdaten im Sprite speichern, so daß der Bildschirm und das Sprite wieder auf deren ursprünglichen Zustand zurückgestellt werden können.

SPN bestimmt in allen Fällen die Nummer des MASKierten Sprite und COL und ROW geben die linke obere Ecke des Bildschirmfensters an. Die Fensterhöhe entspricht der Höhe des Sprite, die Fensterbreite ist jedoch nur die Hälfte von der Spritebreite. Dies ist auf das interne Format von MASKierten Sprites zurückzuführen (s. MASK). Falls einer dieser Befehle mit einem unmaskierten Sprite ausgeführt wird, dann ist das Ergebnis nicht vorhersehbar. Überlappt das Fenster den Bildschirmrand, dann wird die Operation nur auf dem am Bildschirm befindlichen Teil des Fensters stattfinden. Ist das Fenster gänzlich außerhalb des Bildschirms ("off-screen"), dann wird keine Operation stattfinden. Es wird allerdings auch keine Fehlermeldung erzeugt. Obwohl die obige Beschreibung etwas verwirrend klingen mag, ist die Operation selbt ganz einfach.

Für jeden der drei Befehle:

Parameter	Verwendung
SPN	Die Nummer des maskierten Sprite, das verwendet werden soll.
COL	Die Bildschirmspalte des Fensters.
ROW	Die Bildschirmreihe des Fensters.
@V1	Wird bei Kollisionserfassung verwendet.

Befehl	Vorgang
FSWP	Das maskierte Sprite wird vor die Bildschirmdaten gesetzt und die Bildschirmdaten werden in das Sprite gehoben. Achten Sie darauf, daß ein zweiter FSWP beim gleichen COL und ROW und mit demselben Sprite, das Sprite und den Bildschirm auf den vorherigen Zustand zurückbringt. Wird FSWP kein zweites Mal ausgeführt, dann bleibt das Sprite verstümmelt.
BPUT	Das maskierte Sprite wird hinter Bildschirmdaten gesetzt und die Bildschirmdaten werden in das Sprite gehoben. Dieser Befehl bewirkt einen anderen Austausch wie FSWP und BPUT sollte deshalb nicht zweimal ausgeführt werden, um die Daten wieder mit BGET auf den Anfangswert zurückzustellen.
BGET	Die Daten des maskierten Sprite, die mit BPUT hinter die Bildschirmdaten gesetzt wurden, werden in das Sprite zurückgehoben und der Bildschirm wird wieder auf jene Zustand zurückgestellt, den wir vor Verwendung von BPUT hatten. Achten Sie darauf, daß nach BPUT immer BGET verwendet wird, da sonst das maskierte Sprite verstümmelt bleibt.

ANMERKUNG: Wird ein Sprite in einer BPUT-BGET-Befehlsfolge verwendet, dann kann es nicht in einer FSWP-FSWP-Befehlsfolge verwendet werden, es sei denn, das Sprite wird mit RMSK neu maskiert.

Kollisionserfassung

Datenaustausch unterstützt Kollisionserfassung in der gleichen Art und Weise wie die früher beschriebenen GETs und PUTs.

Es gibt vier MOVE-Befehle (Bewegungsbefehle), mit denen ein eingestelltes Sprite von eine Bildschirmste... zu einer anderen bewegt werden kann. Der einzige Unterschied liegt in de Operationsart, die sie zur Durchführung der Bewegung verwenden. Alle MOVE-Befehle könne mehrmals ausgeführt werden und haben eine Kollisionserfassungsoption. Sämtliche MOVE-Befehl sind mit der Nachsilbe MOV versehen.

Bewegungsabläufe

Die MOVE-Befehle bieten auch eine Möglichkeit für Trickbewegungen und sie benutzen die vi Grafikvariablen SP1, SP2, SP3, SP4 für diesen Zweck. Jedesmal wenn ein MOVE-Befehl ausgefüh wird, wird eine Kontrolle durchgeführt, um zu sehen, ob HGT bzw. LEN ungleich null sind. Wen beide tatsächlich null sind, dann erfolgt keine Ausführung und Trickbewegungen finden auch nicl statt. Falls HGT bzw. LEN oder beide null sind, wird eine Bewegung stattfinden. Vorläufig nehmen w an, daß SP1 jenes Sprite ist, das vorher auf die in COL und ROW gehaltenen Koordinaten geset wurde. Dieses Sprite wird entfernt, die Erhöhungen HGT und LEN werden zu COL und ROW addie und Sprite SP2 wird auf die neue COL- und ROW-Position gesetzt (PUT). Weiters werden die alte Werte für COL und ROW durch die neuen Werte (erhöhten Werte) ersetzt und die vi Spritenummern werden rotiert. Das heißt, SP1 übernimmt den Wert von SP2, SP2 übernimmt de Wert von SP3, SP3 übernimmt den Wert von SP4 und SP4 übernimmt den Wert von SP1. Soll e Sprite ohne Trickbewegung bewegt w...rden, dann brauchen SP1, SP2, SP3 und SP4 nur s eingestellt werden, daß sie die gleiche Nummer enthalten.

Für jede der vier MOVE-Befehle:

Parameter	Verwendung
SP1	Das Sprite, das bewegt werden soll.
SP2	Das neue Sprite, das auf den Bildschirm gebracht werden soll (PUT).
SP3	Das Sprite, das nach Entfernung von SP2 eingestellt wird (PUT).
SP4	Das Sprite, das nach Entfernung von SP3 eingestellt wird (PUT).
COL	Bildschirmspalte des Sprite, das bewegt werden soll.
ROW	Bildschirmreihe des Sprite, das bewegt werden soll.
HGT	Y-Zunahme der Bewegung.
LEN	X-Zunahme der Bewegung.
@V1	Variablenadresse für Kollisionserfassung.
e1	Häufigkeit der Befehlsausführung.
e2	Hinweismarke für Rahmenrücklauf.

Befehl	Vorgang
FMOV	Bewegung und Trickeffekt vor den Bildschirmdaten. Beachten Sie, daß da maskierte Sprite auf den Bildschirm gesetzt (PUT) und für die Bewegung bereit se muß, bevor der erst FMOV-Befehl ausgeführt werden kann. Dies läßt sich entwed durch einen FSWP (wenn SPN gleich SP1 ist) erzielen oder durch Bewegung v einer Position außerhalb des Bildschirms ("off-Screen") zu einer Position a Bildschirm ("on-screen") bzw. einer teilweise am Bildschirm befindlichen Positio Eine FMOV-Befehlsfolge kann entweder durch einen anderen FWSP mit SP gleich SP1 abgebrochen werden, oder durch Bewegung aus dem Bildschir ("off-screen").
FMOV,@V1	Bewegung und Trickeffekt erfolgen wie bei FMOV, aber diesmal ist die Kollisionse fassung eingeschaltet. Die ganzzahlige Variable V1 wird erhöht, wenn eine Kollisi festgestellt wird.
FMOV,e1,e2	Bewegung und Trickeffekt erfolgen wie bei FMOV, e1 mal, mit Rücklaufsynchro sierung, wenn e2 null ist.
FMOV, @V1,e1,e2	Bewegung und trickeffekte erfolgen wie bei FMOV, mit Kollisionserfassung, e1 m mit oder ohne Rücklaufsynchronisierung.

das maskierte Sprite auf den Bildschirm gesetzt (PUT) und für die Bewegung bereit sein muß, bevor der erste BMOV-Befehl ausgeführt werden kann. Dies läßt sich entweder durch einen BPUT (wo SPN gleich SP1 ist) erzielen oder durch Bewegung von einer Position außerhalb des Bildschirms ("off-screen") zu einer Position am Bildschirm ("on-screen") bzw. einer teilweise am Bildschirm befindlichen Position. Eine BMOV-Befehlsfolge kann entweder durch Ausführung eines BGET mit SPN gleich SP1 abgebrochen werden, oder durch Bewegung aus dem Bildschirm ("off-screen"). BMOV und BGET können nur mit maskierten Sprites verwendet werden. Jedes Sprite, an dem ein BMOV ausgeführt wurde, muß neu maskiert werden, bevor ein FMOV ausgeführt werden kann.

BMOV,@V1 Bewegung und Trickeffekte erfolgen wie bei BMOV, aber diesmal ist die Kollisionserfassung in Betrieb. Die ganzzahlige Variable V1 wird erhöht, wenn eine Kollision festgestellt wird.

BMOV,e1,e2 Bewegung und Trickeffekte erfolgen wie bei BMOV, e1 mal, mit Rücklaufsynchronisierung, wenn e2 ungleich null ist, ohne Rücklaufsynchronisierung, wenn e2 null ist.

BMOV,@V1, e1,e2 Bewegung und Trickeffekte erfolgen wie bei BMOV, mit Kollisionserfassung, e1 mal, mit oder ohne Rücklaufsynchronisierung.

XMOV Bewegung und Trickeffekte erfolgen unter Verwendung der Exklusiv-ODER-Operation. Achten Sie darauf, daß das maskierte Sprite auf den Bildschirm gesetzt (PUT) und für die Bewegung bereit sein muß, bevor der erste XMOV-Befehl ausgeführt werden kann. Dies läßt sich entweder durch einen PTXR-Befehl (wo SPN gleich SP1 ist) erzielen oder durch Bewegung von einer Position außerhalb des Bildschirms ("off-screen") zu einer Position am Bildschirm ("on-screen") bzw. einer teilweise am Bildschirm befindlichen Position. Eine XMOV-Befehlsfolge muß nicht abgebrochen werden, da die verwendeten Sprites nicht von der XOR-Operation betroffen sind. Soll jedoch das Sprite vom Bildschirm entfernt werden (ohne das Bildschirmfenster zu löschen), dann kann dies erzielt werden, indem ein weiterer PTXR ausgeführt wird (mit SPN gleich SP1) oder eine Bewegung aus dem Bildschirm ("off-screen") erfolgt.

XMOV,@V1 Bewegung und Trickeffeckte erfolgen wie bei XMOV, aber diesmal ist die Kollisionerfassung in Betrieb. Die ganzzahlige Variable V1 wird erhöht, wenn eine Kollision festgestellt wird.

XMOV,e1,e2 Bewegung und Trickeffekte wie bei XMOV, e1 mal, mit Rahmenrücklaufsynchronisierung, wenn e2 ungleich null ist, ohne Rahmenrucklaufsynchronisierung, wenn e2 null ist.

XMOV,@V1, e1,e2 Bewegung und Trickeffekte erfolgen wie bei XMOV, mit Kollisionserfassung, e1 mal, mit oder ohne Rahmenrückaufsynchronisierung.

WMOV Bewegung und Trickeffekte erfolgen unter Verwendung einer Blocküberschreibungs-Operation. Die Blocküberschreibungs-Operation ist um einiges einfacher als die vorhin beschriebenen drei MOV-Befehle, sie kann aber um das 10fache schneller ausgeführt werden. Bei der Ausführung von WMOV wird angenommen, daß SP1 beim laufenden COL und ROW auf den Bildschirm gebracht wurde (PUT). WMOV addiert im Prinzip nur LEN und HGT zu den laufenden COL- und ROW-Werten und setzt (PUT) SP2 auf diese neuen Koordinaten. Aus diesem Grund muß SP2 einen leeren Rand haben (keine Daten dürfen enthalten sein), der mindestens so groß sein muß wie die Erhöhungen HGT und LEN, so daß das alte Sprite SP1 vom neuen Sprite SP2 vollkommen überdeckt wird. Es werden hierbei natürlich alle anderen von SP2 verwendeten Bildschirmdaten im Fenster zerstört. Diese Operation sollte aber jedesmal verwendet werden, wenn ein Sprite über einen leeren Hintergrund bewegt wird und keine anderen Daten vorhanden sind, da sich dadurch eine erhebliche Zeiteinsparung erzielen läßt. Das Sprite kann unter Verwendung von CLSV entfernt werden oder durch Bewegung aus dem Bildschirm ("off-screen").

WMOV,@V1 Bewegung und Trickeffekte erfolgen wie bei WMOV, aber diesmal ist die Kollisionserfassung in Betrieb und die ganzzahlige Variable V1 wird erhöht, wenn eine Kollision stattfindet. In der Praxis bringt es nicht viel, diese Option zu verwenden, da eine Kollisionserfassung unvermeidbar ist (SP1 wird gelöscht und Kollisionserfassung verlangsamt die Ausführung erheblich).

WMOV,e1,e2 Bewegung und Trickeffekte erfolgen wie bei WMOV, e1 mal, mit Rücklaufsynchronisierung, wenn e2 ungleich null ist, ohne Rücklaufsynchronisierung, wenn e1 null ist.

WMOV,@V1, e1,e2 Bewegung und Trickeffekte erfolgen wie bei WMOV, mit Kollisionserfassung, e1 mal, mit oder ohne Rücklaufsynchronisierung. Es ist auch hier unwahrscheinlich, daß die Kollisionserfassung von Nutzen sein wird.

JOYSTICK-/TASTATUR-MOVE-BEFEHLE

In dieser Gruppe gibt es vier Befehle, mit deren Hilfe ein Sprite mit Tastatur-bzw. Joysticksteuerung von einer Bildschirmstelle zu einer anderen bewegt werden kann. Wie bei den vorhin beschriebenen MOVE-Befehlen, liegt der Unterschied einzig und allein in der Operationsart, die für die Bewegung verwendet wird. Die vier Befehle in dieser Kategorie können mehrmals ausgeführt werden und haben eine Kollisionserfassungsoption zur Verfügung. Alle Joystick-/Tastatur-MOVE-Befehle haben die Nachsilbe MVJ.

Bewegungsabläufe

Die J/K-(Joystick/Tastatur) MOVE-Befehle bieten die gleichen Bewegungsmöglichkeiten wie die vorhin beschriebenen linearen MOVE-Befehle. Wenn der Joystick bzw. die Tastatur nicht aktiviert werden, dann erhalten die X- und Y-Zunahmen Nullwerte und es findet keine Bewegung statt. Dies bedeutet, daß die Trickbewegungsfolge auch nicht erhöht wird, und es wird somit verhindert, daß die Trickbewegung aus dem Schritt fällt.

Für jeden der vier Befehle

Parameter	Verwendung
SP1	Das Sprite, das entfernt werden soll.
SP2	Das nächste Sprite, das auf den Bildschirm gebracht werden soll (PUT)
SP3	Das Sprite, das auf den Bildschirm gebracht wird (PUT), nachdem SP2 entfernt wurde
SP4	Das Sprite, das auf den Bildschirm gebracht wird (PUT), nachdem SP3 entfernt wurde
COL	Bildschirmspalte des Sprite, das entfernt werden soll.
ROW	Bildschirmreihe des Sprite, das entfernt werden soll.
HGT	Y-Zunahme der Bewegung.
LEN	X-Zunahme der Bewegung
@V1	Variablenadresse für Kollisionserfassung.
e1	Häufigkeit der Befehlsausführung.
e2	Hinweismarke für Rahmenrücklaufsynchronisierung.
KEY	Eigentliche Reihe, die für die Bewegung abgefragt wird (s. "STARTEN MIT LASER BASIC").

Joystick-/Keyboard-Steuerrung

Wenn der Joystick bzw. die Tastatur nicht aktiviert werden, dann findet keine Operation statt.

Wird der Joystick/die Tastatur nach links bewegt, dann bewegt sich das Sprite um den in LEN gehaltenen Wert nach links. Wenn LEN einen negativen Wert enthält, dann bewegt es sich nach rechts. Ist LEN null, dann folgt keine Bewegung bzw. Trickeffekt.

Wenn der Joystick/die Tastatur nach rechts bewegt wird, dann wird sich das Sprite um den in LEN gehaltenen Wert nach rechts bewegen. Wenn LEN

... HGT
... enthält, dann bewegt es sich
... ist HGT null, dann findet keine Bewe... g bzw. Trickeffekt statt

Wenn der Joystick/die Tastatur nach unten bewegt wird, dann wird sich das Sprite um den in HGT gehalten Wert nach unten bewegen. Wenn HGT einen negativen Wert enthält, dann bewegt es sich nach oben. Ist HGT null, dann findet keine Bewegung bzw. Trickeffekt statt.

Kombinationen dieser Bewegungstypen bewirken diagonale Bewegungen.

Befehl	Vorgang
FMVJ	Derselbe Vorgang wie beim linearen Bewegungsbefehl FMOV, d.h. Bewegung erfolgt vor den Bildschirmdaten, die Bewegungsrichtung wird aber vom Joystick bzw. von der Tastatur gesteuert.
FMVJ,@V1	Derselbe Vorgang wie beim linearen Bewegungsbefehl FMOV,@V1, d.h. die Bewegung erfolgt vor den Bildschirmdaten, mit Kollisionserfassung, die Bewegungsrichtung wird aber vom Joystick bzw. von der Tastatur gesteuert.
FMVJ,@V1, e1,e2	Derselbe Vorgang wie beim linearen Bewegungsbefehl FMOV,@V1,e1,e2, d.h. Bewegung erfolgt vor den Bildschirmdaten, e1 mal, mit oder ohne Rahmenrücklaufsynchronisierung. Bewegungsrichtung wird vom Joystick von der Tastatur gesteuert.
BMVJ	Derselbe Vorgang wie beim linearen Bewegungsbefehl BMOV, d.h. die Bewegung erfolgt hinter den Bildschirmdaten, die Bewegungsrichtung wird aber vom Joystick bzw. von der Tastatur gesteuert.
BMVJ,@V1	Derselbe Vorgang wie beim linearen Bewegungsbefehl BMOV,@V1, d.h. die Bewegung erfolgt hinter Bildschirmdaten, mit Kollisionserfassung, die Bewegungsrichtung wird aber vom Joystick bzw. von der Tastatur gesteuert.
BMVJ,@V1, e1,e2	Derselbe Vorgang wie beim linearen Bewegungsbefehl BMOV,@V1,e1,e2 d.h. die Bewegung erfolgt hinter Bildschirmdaten, e1 mal, mit oder ohne Rücklaufsynchronisierung. Die Bewegungsrichtung wird vom Joystick bzw. von der Tastatur gesteuert.
XMVJ	Derselbe Vorgang wie beim linearen Bewegungsbefehl XMOV, d.h. die Bewegung erfolgt unter Verwendung der Exklusiv-ODER-Operation, die Bewegungsrichtung wird aber vom Joystick bzw. von der Tastatur gesteuert.
XMJV,@V1	Derselbe Vorgang wie beim linearen Bewegungsbefehl XMOV,@V1, d.h. die Bewegung erfolgt unter Verwendung der Exklusiv-ODER-Operation, mit Kollisionserfassung, die Bewegungsrichtung wird aber vom Joystick bzw. von der Tastatur gesteuert.
XMVJ,@V1, e1,e2	Derselbe Vorgang wie beim linearen Bewegungsbefehl XMOV,@V1,e1,e2 d.h. die Bewegung erfolgt unter Verwendung der Exklusiv-ODER-Operation, e1 mal, mit oder ohne Rahmenrücklaufsynchronisierung. Die bewegungsrichtung wird vom Joystick bzw. von der Tastatur gesteuert.
WMVJ	Derselbe Vorgang wie beim linearen Bewegungsbefehl WMOV, d.h. die Bewegung erfolgt mit der Blocküberschreibungs-Operation, die Bewegungsrichtung wird aber vom Joystick bzw. von der Tastatur gesteuert
WMJV,@V1	Derselbe Vorgang wie beim linearen Bewegungsbefehl WMOV,@V1, d.h. die Bewegung erfolgt unter Verwendung der Blocküberschreibungs-Operation mit Kollisionserfassung, die Bewegungsrichtung wird aber vom Joystick bzw. von der Tastatur gesteuert

WMVJ,@V... e1,e2	Derselbe Vorgang wie beim linearen Bewegungsbefehl WMOV,@V1,e1,e2, d.h. Bewegung erfolgt unter Verwendung der Blocküberschreibungs-Operation, e1 r... mit oder ohne Rahmenrücklaufsynchronisierung. Die Bewegungsrichtung wird v... Joystick bzw. von der Tastatur gesteuert.

SPRUNG-MOVE-BEFEHLE

Es gibt in dieser Gruppe vier Befehle. Mit diesen Befehlen ist es möglich, ein bereits eingestell... Sprite innerhalb des Rahmens eines vordefinierten rechteckigen Fensters springen zu lassen. D... Fenster wird mit dem BWST-Befehl aufgebaut. Dieser Befehl wird am Ende dieses Abschnitts näl... beschrieben. Der Unterschied zwischen den vier Befehlen in dieser Gruppe liegt wiederum in d... verwendeten Bewegungsart. Jeder Befehl in dieser Kategorie kann mehrmals ausgeführt werd... und hat eine Sprungerfassungsoption zur Verfügung. Alle Sprung-MOVE-Befehle sind mit d... Nachsilbe BNC versehen.

Bewegungsabläufe

Die Sprung-MOVE-Befehle benutzen SP1, SP2 SP3 und SP4 um eine Option für Trickbewegunge... zu bieten. Die 'Rahmen' werden jedesmal gedreht, wenn ein Befehl ausgeführt wird, es sei denn d... MOVE-Zunahmen in HGT und LEN sind auf null eingestellt. In diesem Fall wird keine Operatio... stattfinden. Wenn eine 'Sprungbewegung' stattfindet, dann wird der Bewegungsablauf automatisc... umgekehrt. Wenn die Sprungerfassung in Betrieb ist, dann hat der Anwender die Möglichkeit de... Ablauf unter Verwendung von SWPS neuerlich umzukehren, wenn auf einen Sprung getestet wurd... und ein solcher tatsächlich stattgefunden hat. Wenn allerdings das Sprite auf zwei Seiten gleichzeiti... springt, dann werden sich die beiden 'wiederholten Umkehrungen' gegenseitig aufheben und di... Trickbewegung läuft nicht in die andere Richtung ab. Es sollte ebenfalls darauf geachtet werden, da... ein Doppelsprung die Kollisionserfassungsvariable V1 weiterhin nur um 1 erhöhen wird. Es gibt kein... Möglichkeit einen Doppelsprung zu testen, ohne die Kollisionserfassungsvariable und SP2 z... untersuchen.

Für jeden der vier Befehle:

Parameter	Verwendung
KEY	Nummer des Sprite, das die Sprungfensterparameter enthält. Achten Sie darauf, daß KEY und nicht SPN verwendet wird.
SP1	Sprite, das entfernt werden soll.
SP2	Das nächste Sprite das auf den Bildschirm gesetzt wird (PUT)
SP3	Das Sprite, das auf den Bildschirm gesetzt wird (PUT), nachdem SP2 entfernt wurde.
SP4	Das Sprite, das auf den Bildschirm gesetzt wird (PUT), nachdem SP3 entfernt wurde..
COL	Bildschirmspalte des Sprite, das entfernt werden soll.
ROW	Bildschirmreihe des Sprite, das entfernt werden soll.
HGT	Y-Zunahme der Bewegung.
LEN	X-Zunahme der Bewegung.
@V1	Variablenadresse für Kollisionserfassung
e1	Häufigkeit der Befehlsausführung
e2	Hinweismarke für Rahmenrücklauf.

Befehl	Vorgang
FBNC	Derselbe Vorgang wie beim linearen Bewegungsbefehl FMOV, d.h. die Bewegung erfolgt vor den Bildschirmdaten, die Bewegungsrichtung ist aber von der Sprungbedingung abhängig, die vom vordefinierten Sprungfenster bestimmt wird.
FBNC,@V1	Derselbe Vorgang wie beim linearen Bewegungsbefehl FMOV,@V1 d.h. die Bewegung erfolgt vor den Bildschirmdaten. Diesmal aber mit Sprungerfassung. Die Bewegungsrichtung ist allerdings von der Sprungbedingung abhängig.

e1,e2	Bewegung erfolgt vor den Bildschirmdaten, mit Kollisionserfassung, e1 ma'. mit oder ohne Rahmenrücklaufsynchronisie g. Bewegungsrichtung ist von der Sprungbedingung abhängig.
BBNC	Dieselbe Operation wie bei FBNC, die Bewegung läuft aber hinter den Bildschirmdaten ab.
BBNC,@V1	Dieselbe Operation wie bei FBNC,@V1, die Bewegung findet aber hinter Bildschirmdaten statt.
BBNC,@V1, e1,e2	Dieselbe Operation wie bei FBNC,@V1,e1,e2, die Bewegung findet aber hinter Bildschirmdaten statt.
XBNC	Dieselbe Operation wie bei FBNC, die Bewegung erfolgt aber unter Verwendung der XOR-Operation.
XBNC,@V1	Dieselbe Operation wie bei FBNC,@V1, die Bewegung erfolgt aber unter Verwendung der XOR-Operation.
XBNC,@V1, e1,e2	Dieselbe Operation wie bei FBNC,@V1,e1,e2, die Bewegung erfolgt aber unter Verwendung der XOR-Operation.
WBNC	Dieselbe Operation wie bei FBNC, die Bewegung erfolgt aber unter Verwendung der Blocküberschreibungs-Operation.
WBNC,@V1	Dieselbe Operation wie bei FBNC,@V1, die Bewegung erfolgt aber unter Verwendung der Blocküberschreibungs-Operation.
WBNC,@V1, e1,e2	Dieselbe Operation wie bei FBNC,@V1,e1,e2, die Bewegung erfolgt aber unter Verwendung der Blocküberschreibungs-Operation.

BWST

Es ist ein zusätzlicher Befehl erforderlich, um ein Fenster aufzubauen, in dem Sprungsprites springen sollen. Der Befehl wird in diesem Abschnitt beschrieben, da er nur für diese Operationsgruppe gültig ist.

Parameter	Verwendung
KEY	Die Nummer des Sprite, das die Fensterinformationen enthalten wird.
COL	Die linke Spalte des Fensters. Das Sprite wird sich nicht weiter nach links bewegen.
ROW	Die oberste Reihe des Fensters. Dies ist der niedrigste Reihenwert, bevor das Sprite zu springen beginnt.
LEN	Dies ist die Länge des Fensters und entspricht dem horizontalen Weg, gemessen von COL, den das Sprite entlang bewegt wird, bevor es zu springen beginnt. Achten Sie darauf, daß die Spritegröße die Fenstergröße nicht beeinflussen wird, da die Spritekoordinaten innerhalb des Fenster verändert werden.
HGT	Dies ist die Höhe des Fensters und entspricht dem vertikalem Weg, gemessen von ROW, den das Sprite entlangbewegt wird, bevor es zu springen beginnt.

Befehl	Vorgang
BWST	Die Sprungfensterdaten werden in Sprite KEY eingestellt. Achten Sie darauf, daß das Sprite, das die Daten enthält, eine Höhe von 1 und eine Länge von 4 haben muß, da sonst ein Parameterfehler erzeugt wird

DIE DATENABTASTBEFEHLE

Es gibt vier Befehle, die es erlauben, Bildschirmfenster, Sprites und Spritefenster auf Bildpunktdaten zu untersuchen. Diese sind als Ergänzung zur Collisionserfassung vorgesehen. Sie können gemeinsam mit den logischen PUTs und GETs verwendet werden, um Kollisionserfassung und Musteranpassung zu ermöglichen. Wenn z.B. ein Bildschirmfenster mit einem Sprite mittels AN verknüft wird, dann können die SCAN- (Abtast) Befehle dazu verwendet werden, Daten zu erfasse die, falls vor len, eine Kollision andeuten. Wenn ein Bildschirmfenster mit einem Sprite mitte XOR verknüpft wird, dann können die SCAN-Befehle ebenfalls verwendet werden, um Daten z erfassen, die, falls nicht vorhanden, eine Musteranpassung andeuten. Es gibt hierzu drei Datenarte (durch die Befehlsnachsilbe angedeutet). Diese sind:

Typ V

Das Abtasten erfolgt auf einem Bildschirmfenster, dessen linke obere Ecke durch COL und RO definiert ist, und dessen Abmessungen durch HGT und LEN bestimmt sind. Die Fensterparamet werden verstellt, wenn das Fenster teilweise außerhalb des Bildschirms ("off-screen") liegt.

Typ S

Das Abtasten erfolgt auf dem ganzen Sprite, dessen Nummer in SPN gehalten wird.

Typ P

Das Abtasten erfolgt auf einem Spritefenster. Die Nummer des Sprite wird von SPN angegeben ur das Fenster wird durch COL, ROW, HGT, und LEN definiert. Das Fenster wird verstellt, wenn e teilweise außerhalb des Sprite ("off-sprite") liegt.

Parameter	Verwendung
COL	Bildschirmspalte der linken oberen Fensterecke.
ROW	Bildschirmreihe der linken oberen Fensterecke.
LEN	Fensterbreite.
HGT	Fensterhöhe.
V1	Datenerfassungsvariable.

Befehl	Vorgang
SCNV,@V1	Der Bildschirm wird auf Bildpunktdaten untersucht. Wenn diese gefunden werde wird die ganzzahlige Variable V1 erhöht.
SUMV,@V1	Die 16-Bit-Summe sämtlicher Daten im Spritefenster wird gebildet und das Ergebni wird der BASIC-Variable V1 zugeordnet.
SUMV, e1,e2....@V1	Wie vorhin wird auch hier die 16-Bit-Summe sämtlicher Daten im Bildschirmfenste gebildet. Wenn jedoch mehr als ein Parameter verwendet wird, dann wird di Summe mit jedem Ausdruck in der Liste und der Position des ersten Ausdrucks i der Liste verglichen. Um die Muster anzupassen, wird die berechnete Summe V zugeordnet. Wenn kein passendes Muster gefunden wurde, dann wird V1 zugeordnet. Es sind maximal 8 Parameter (einschließlich @V1) erlaubt.

Parameter	Verwendung
SPN	Die Nummer des Sprite, das abgetastet werden soll.
V1	Datenerfassungsvariable.

Befehl	Vorgang
SCNS,@V1	Das Sprite wird nach Bildpunktdaten abgetastet. Falls welche gefunden werde wird die ganzzahlige Variable V1 erhöht
SUMS,@V1	Die 16-Bit-Summe samtlicher Spritedaten wird berechnet und das Ergebnis wird de BASIC-Variable V1 zugeordnet.
SUMS, e1,e2....@V1	Dieser Befehl funktioniert genauso wie der vorhin beschriebene SUMV-Befeh Hierbei wird jedoch anstatt des Bildschirmfensters das Sprite verwendet, desse

SPN	Die Nummer des Sprite, das das abzutaste[illegible] Fenster enthält
COL	Die Spalte der linken oberen Fensterecke.
ROW	Die Reihe der linken oberen Fensterecke.
LEN	Die Breite des Spritefensters.
HGT	Die Höhe des Spritefensters.

Befehl	Vorgang
SCNP,@V1	Das Spritefenster wird auf Bildpunktdaten untersucht. Falls welche gefunden werden, wird die ganzzahlige Variable V1 erhöht.
SUMP,@V1	Die 16-Bit-Summe sämtlicher Daten im Spritefenster wird berechnet und das Ergebnis wird der BASIC-Variable V1 zugeordnet.
SUMP, e1,e2,...@V1	Dieser Befehl funktioniert genauso wie die vorhin beschriebenen SUMV- und SUMS-Befehle. Hierbei wird allerdings ein Spritefenster für die Datensummierung verwendet.

SONSTIGE BEFEHLE

FILL

Dieser Befehl ist vorgesehen, um unregelmäßige Flächen mit einer bestimmten Farbe (INK) zu füllen. Der Anwender muß die Koordinaten (in Bildpunkten) bestimmen, bei denen das Füllen beginnen soll, sowie die Farbe (INK), die zum Füllen (FILL) verwendet werden soll. Der Bereich, der die gewählte Farbe des Bildpunktes enthält, bei dem begonnen wird, wird mit der entsprechenden Farbe (INK) gefüllt. Im 4-Farbenmodus ist der Bildschirm 320 Bildpunkte breit, im 16-Farbenmodus ist der Bildschirm 160 Bildpunkte breit. In beiden Fällen ist der Bildschirm 200 Bildpunkte hoch. Der FILL-Befehl belastet den Maschinenstapelspeicher sehr stark. Und wenn der Gegenstand besonders kompliziert ist, könnte unter Umständen ** OUT OF MEMORY ** (Kein Speicherplatz mehr) erscheinen, bevor die Befehlsausführung abgeschlossen wird. Der Befehl funktioniert nur mit einer Datenart – dem Bildschirm.

Parameter	Verwendung
XCL	Bildpunktkoordinate am Anfangspunkt (0 bis 319 im 4-Farbenmodus oder 0 – 159 im 16-Farbenmodus) gemessen von der linken Seite.
ROW	Bildpunktkoordinate am Anfangspunkt (0 bis 199 in beiden Farbmodi) gemessen von der Bildschirmoberseite.
IK1	INK-Nummer zum Füllen (0 bis 3 oder 0 bis 15).

Befehl	Vorgang
FILL	Der Bildschirm wird vom Anfangspunkt aus gefüllt. Die Umgrenzung besteht aus Bildpunkten mit einer anderen Farbe als der Anfangspunkt oder der Bildschirmrand.

SPNV

Das Bildschirmfenster, angegeben durch COL, ROW, HGT und LEN, wird um 90 Grad in ein Fenster rotiert, dessen Abmessungen die transponierte Version des Ausgangsfensters sind. Die linke obere Ecke dieses Fensters wird durch SCL und SRW angegeben. Die Fenster werden verstellt, so daß sie auf dem Bildschirm liegen ("on-screen"). Die Ausgangs- und Zielfenster sollten sich nicht überlappen, da das Ergebnis sonst nicht vorherbestimmbar ist und recht spektakulär ausfallen könnte. Dieser Befehl dient nur für Bildschirmdaten. Achten Sie darauf, daß die Höhe des zu drehenden ('SPUN') Fensters auf die nächsten 8 Bildpunkte abgerundet wird, d.h. HGT-Werte von 25, 26, 27, 28, 29, 30 und 31 würden alle auf 24 abgerundet werden usw.

Parameter	Verwendung
COL	[illegible]schirmspalte der linken oberen Ecke des Ausgangsfensters.
ROW	Bildschirmreihe der linken oberen Ecke des Ausgangsfenster.
HGT	Höhe des Fensters, das rotiert werden soll. (Mod 8).
LEN	Breite des Fensters, das rotiert werden soll.
SCL	Bildschirmspalte der linken oberen Ecke des Zielfensters.
SRW	Bildschirmreihe der linken oberen Ecke des Zielfensters.

Befehl	Vorgang
SPNV	Das Bildschirmfenster wird um 90 Grad im Uhrzeigersinn in ein zweites Fenster rotiert. Die transponierten Abmessungen der linken oberen Ecke des Zielfensters werden durch SCL und SRW bestimmt.

KBFN,@V1

Dieser Befehl tastet die Tastatur-Hardware ab (das BASIC-äquivalent tastet die Tabelle ab, die bei Interrupt auf den neuesten Stand gebracht wird). KBFN kann daher auch zum Abtasten verwendet werden, wenn Interrupt blockiert ist. Dem Befehl KBFN wird ein Parameter nachgestellt, der der Adresse der ganzzahligen BASIC-Variable V1 entspricht. Wenn die in der Grafikvariable KEY angegebene Taste gedrückt wird, dann wird V1 erhöht, ansonsten bleibt sie unverändert. Es können daher mehrere Tasten gedrückt werden. Wenn aber in einer Reihe drei Tasten gedrückt werden, dann kann die Hardware unter Umständen einen vierten Tastendruck anzeigen.

Parameter	Verwendung
KEY	Nummer der Taste, die getestet werden soll (s. Anhang A).

Befehl	Vorgang
KBFN,@V1	Die Taste, deren Nummer in KEY gehalten wird, wird getestet und V1 wird erhöht, wenn sie gedrückt wird.

SCLS

Der ganze Bildschirm wird auf INK 0 (Papierfarbe) eingestellt, der Cursor springt auf die Ausgangsstellung in der linken oberen Ecke, der Bildschirm wird auf Fenster 0 mit voller Breite und Höhe von 24 Reihen eingestellt. Die unterste Reihe kann von BASIC nicht benutzt werden, die Grafikroutinen benutzen allerdings den ganzen Bildschirm. Wenn der Anwender das laufende Fenster so einstellt, daß es dem ganzen Bildschirm entspricht, dann wird dadurch Hardware-Rollen ermöglicht. Falls dies passiert, ist es notwendig, SCLS vor den erweiterten Grafikbefehlen auszuführen. SCLS hat keine Parameter.

Befehl	Vorgang
SCLS	Bildschirm wird gelöscht, Cursor wird auf die Ausgangsstellung zurückgestellt, der Bildschirm wird der laufende Strom und ein Fenster mit der Breite des Bildschirms und der Höhe von 24 Reihen wird definiert (1 weniger als der volle Bildschirm).

BILD

Dieser Befehl wird benutzt, um die Bildschirmkulissen aufzubauen. Er läßt sich am besten durch ein Beispiel erklären. Eine Datenkomprimierungsmethode wird benutzt, so daß Spritebaublocke durch Bits in einem Datensprite dargestellt werden. Der Befehl hat 5 Parameter. COL und ROW werden verwendet, um den Bauanfang zu bestimmen. Achten Sie darauf, daß COL und ROW außerhalb des Bildschirm liegen können und die Kulissen größer als der Bildschirm sein könnten. In jedem Fall wird der Befehl soviele Daten wie möglich auf den Bildschirm bringen. Die Grafikvariable KEY enthält die Operationsart, die verwendet werden soll, um die Daten auf den Bildschirm zu bringen (PUT).

0	Blocküberschreibung
1	Exklusiv-ODER
2	PUT vor den laufenden Daten
3	PUT hinter den laufenden Daten

Die Grafikvariable SP1 wird verwendet, um die Nummer jenes Sprite zu halten, das als Baublock verwendet wird, und die Variable SPN enthält die Nummer jenes Sprite, das das BIT-Muster angibt, welches festlegt, wo die einzelnen Baublöcke gesetzt werden (PUT).

Datenspriteformat

Das Format für das Datensprite (das das Bit-Muster enthält) ist recht einfach. Die Breite des Szenarios in "Sprites" wird 8 mal so groß sein wie die Breite des Datensprite, da es 8 Bits pro Byte gibt. Die Höhe des Szenarios in "Sprites" wird der Höhe des Datensprite entsprechen. Die eigentliche Größe des Szenarios hängt auch von der Größe des Sprite ab, das als Baublock verwendet wird. Wenn z.B. der Spritebaublock 20 Bildpunkte hoch und 5 Bytes breit wäre und das Datensprite 2 Bytes breit und 10 Bytes hoch ist, dann würde die Kulisse 5x2x8=80 Bytes breit und 20x10=200 Bildpunkte hoch sein und den Bildschirm ganz ausfüllen. Wenn also Ihr Grundbaublock 20 Bildpunkte hoch und 5 Bytes breit ist, dann würden Sie 20 Bytes pro "Bild" benötigen.

Parameter	Verwendung
COL	Bildschirmspalte, bei der das Bauen beginnt
ROW	Bildschirmreihe, bei der das Bauen beginnt.
KEY	Operationsart.
SPN	Nummer des Datensprite (enthält Bit-Tafel).
SP1	Nummer des Sprite, das für das Bauen verwendet wird.

Befehl	Vorgang
BILD	Das Datensprite, dessen Nummer in SPN gehalten wird, wird von links nach rechts Bit für Bit abgetastet. Wenn ein Bit eingestellt ist, dann wird das Sprite, dessen Nummer in SP1 gehalten wird, auf den Bildschirm gebracht und die Spalte um die Breite des Bausprite vorgerückt. Wenn ein Bit nicht eingestellt ist, dann wird die Spalte einfach um die Breite des Bausprite vorgerückt. Die Operation wird für jede einzelne Reihe im Datensprite wiederholt und die Bildschirmreihe wird jedesmal um die Höhe des Bausprite vorgerückt. Dies bewirkt in der Praxis bloß eine Erweiterung vom Bit zum Sprite.

ANMERKUNG: Die Lücken zwischen den Baublöcken (durch Bits, die auf 0 eingestellt sind, dargestellt) werden nicht löschen überschrieben, und breits vorhandene Daten werden somit nicht gelöscht. Das Datensprite kann unter Verwendung von WSVN, WSL4, WSR4, WSL8 und WSR8 gerollt werden, ohne die Datenstruktur zu verstümmeln. Durch Verwendung der Scroll-Befehle und passenden Werten für COL und ROW können Anwender Szenarios erzeugen, die mehrere Bildschirme hoch und breit sind, und durch sie ein Bildschirmfenster hindurchbewegen. Hierbei handelt es sich jedoch um eine fortgeschrittenen Methode, die nicht verwendet werden soll bis das ganze Paket erlernt wurde.

DEEK und DOKE

DEEK und DOKE sind die 16-Bit-Äquivalente von PEEK und POKE, die diesem Paket hinzugefügt wurden, da sie bei Locomotive BASIC fehlen.

Befehl	Vorgang
DEEK,e1,@V1	16-Bit-Äquivalent von PEEK. Der Inhalt der Adresse e1 wird der ganzzahligen BASIC-Variablen V1 zugeordnet
@V1	Adresse der Variable in der Zuweisung.
e1	SIC-Ausdruck für die Adresse e1.

Befehl	Vorgang
DOKE,e1,e2	16-Bit-Äquivalent von POKE. Das niedrige Byte des Wertes von e1 wird in e1 eingesetzt und das hohe Byte wird in e1+1 gesetzt.

Parameter	Verwendung
e1	Adresse, die "geDOKEt" werden soll.
e2	Daten, die "geDOKEt" werden sollen.

VERFOLGUNGSSPRITES

Obwohl es in diesem Paket darauf ankommt, die einzelnen Funktionen so einfach wie möglich zu halten, wurden für gewagte Anwender einige komplexe Funktionen hinzugefügt. Bevor Sie sich an den Verfolgungssprites versuchen, sollten Sie mit dem restlichen Paket voll vertraut sein. Diese Befehle sind nicht gegen Störungen geschützt und das Ergebnis bei Fehlern ist nicht vorhersehbar.

Format für Verfolgungssprites

Ein Verfolgungssprite, wie es in LASER BASIC vorkommt, ist eigentlich gar kein richtiges Sprite. Ein Verfolgungssprite ist im Prinzip ein primitives Programm, das in einem Blindsprite gehalten wird. Das Programm steuert die Bewgungsrichtung des Sprite, die Operationsart, die für die Bewegung verwendet wird, und welches Sprite für den Bewegungsablauf verwendet werden soll Dies bietet viel mehr Flexibilität als die vorhin beschriebenen MOVE-Befehle und der geübte Programmierer wird schnell bemerken, daß allein mit Verfolgungssprites ganze Programme geschrieben werden können. Das Format des Verfolgungssprittes ist wie folgt:

BYTE	VERWENDUNG
0,1	16-Bit-Programmzähler. Dies ist ein Zeiger, der die nächste Anweisung im Verfolgungssprite anzeigt, und der jedesmal wenn eine Anweisung ausgeführt wird um die Länge der Anweisung vorgeruckt wird.
2	Die Bildschirmspalte, auf die das Sprite zuletzt gesetzt wurde (PUT).
3	Die Bildschirmreihe, auf die das Sprite zuletzt gesetzt wurde (PUT).
4	Die Nummer des Sprite, das zuletzt auf den Bildschirm gebracht wurde (PUT).
5	Die Bewegungsart, die der Verfolger zur Zeit benutzt. Das Typenbyte hat vier Felder: Bits 0 und 1. Diese bestimmen die Bewegungsoperation. 00 = Bewegung erfolgt vor den Bildschirmdaten. 01 = Bewegung erfolgt hinter den Bildschirmdaten. 10 = Bewegung erfolgt unter Verwendung von XOR. 11 = Bewegung erfolgt mit Blockuberschreibung. Bit 2 Wenn Bit 2 eingestellt ist, dann wird die Zunahme vom Joystick/von der Tastatur gesteuert. Bits 3,6 Wenn Bit 2 eingestellt ist, dann bestimmen Bits 3 bis 6 die Tastenreihe/den Joystick, die zu verwenden sind

...ollisions-/Sprungerfassung in Betrieb gesetzt.

Nach den ersten 6 Bytes Systeminformation kommen die Anweisungen selbst. Wenn ein Sprite mit Trickeffekt bewegt werden soll, dann sind die Anweisungen 3 Bytes lang:

BYTE	OPERATION
0	Die Nummer des Sprite, das auf den Bildschirm gebracht werden soll (PUT). Das System weiß noch die Nummer des letzten Sprite, das auf den Bildschirm gebracht wurde (PUT), und wenn X-Z unahme und der Y-Z unahme (s. nächsten Abschnitt) ungleich null ist, dann wird das letzte Sprite, das auf den Bildschirm gebracht wurde (PUT), entfernt und das neue Sprite, das auf den Bildschirm gebracht wird (PUT), erhält dieses Byte. Wenn dieses Byte null ist, dann wird das System keinen MOVE-Befehl ausführen, sondern einen Steuercode suchen (s. Steuercodes im nächsten Abschnitt).
1	Dies ist der X-Zunahme und der Betrag um welchen eine horizontale Bewegung erfolgt. Der MOVE-Befehl achtet darauf, ob das System im 80- bzw. 160-Spaltenmodus ist. Positive Werte bewegen sich nach rechts und negative Werte bewegen sich nach links.
2	Dies ist die Y-Zunahme und der Betrag, um welchen eine vertikal Bewegung erfolgt. Positive Werte bewegen sich nach unten und negative Werte bewegen sich nach oben.

ANMERKUNG: Falls die X-Zunahme und die. Y-Zunahme null sind, wird die Anweisung nicht beachtet und der Programmzähler wird zur nächsten Anweisung vorgerückt, bevor er zurückspringt.

Steuercodes

Falls das Stuerprogramm herausfindet, daß Byte 0 (es wird glauben, daß dies eine Spritenummer ist) eine Null enthält, dann wird es wissen, daß ein Steuercode folgen muß. Es gibt vier Steuercodes und diese sind:

Code	Operation
0	Anfang bei der ersten Anweisung. Der Programmzähler wird so eingestellt, daß er auf die erste Anweisung im Verfolger zeigt. Diese Anweisung wird ausgeführt und der Programmzähler wird zur nächsten Anweisung vorgerückt, bevor er zurückspringt.
1	Eine neue Spur wird eingestellt, wobei die Spritenummer gleich ist wie der Wert im nächsten Byte des laufenden Verfolgers. Das Sprite, das zuletzt auf den Bildschirm gebracht wurde (PUT), wird mit Hilfe der laufenden Operation entfernt und das neue Verfolgungssprite wird anschließend eingeleitet. Beim Verfolgen muß der Programmzähler auf die erste Anweisung zeigen und die COL- und ROW-Werte müssen auf die des zuletzt entfernten Sprite eingestellt werden, und das soeben entfernte Sprite muß mit Hilfe der im neuen Verfolger angegebenen Operation neu eingestellt werden. Aus diesem Grund muß der neue Verfolger dieselbe Spriteart (maskiert oder unmaskiert) verwenden wie der alte Verfolger. Dieser Code soll nur dann verwendet werden, wenn der Verfolger von einem Steuerbaustein ausgeführt wird, da das Ergebnis sonst unberechenbar ist.
2	Spur wird mit einer Bewegungstype fortgesetzt, die gleich ist wie der Wert im nächsten Byte des laufenden Verfolgers. Das laufende Sprite wird von Bildschirm mit Hilfe der laufenden Operation entfernt, und anschließend unter Verwendung der neuen Operation zurückgebracht. Byte 5 der Verfolgervariablen wird abgeändert, un

...gungstype ... halten. Danach wird die nächste ... sung des laufenden Verfolgers ausgeführt. Ein Code von 2 führt daher praktisch ...anweisungen aus. Der Programmzähler wird also um 6 Bytes vorgerückt.

3	Eine neue Spur wird begonnen, wobei die Spritenummer gleich ist wie der Wert i nächsten Byte des laufenden Verfolgers und der Programmzähler ist gleich de Wert in den darauffolgenden nächsten zwei Bytes. Die Wirkung ist genau die gleic wie bei CODE 1. Der neue Verfolger kann allerdings von jeder beliebigen Stelle au eingeleitet werden und muß nicht am Anfang beginnen. Auch dieser Code soll n dann verwendet werden, wenn der Verfolger von einem Steuerbaustein au ausgefuhrt wird.
4	Jener Befehl wird ausgefuhrt, dessen Adresse (s. ADDR) in den übernächsten Bytes ist. Hierbei wird der Variablensatz verwendet, dessen Nummer im nächste Byte ist. Befehlsadressen sind im Abschnitt über compilerbezogene Befehle näh beschrieben.

Einleiten eines Verfolgers

Bevor ein Verfolger mit Hilfe des TMOV-Befehls ausgeführt werden kann, muß er mit dem TPUT-Befehl eingeleitet werden. Der TPUT-Befehl braucht 5 Parameter. Die Spalte und Reihe, in der die Spur beginnt, die Nummer des ersten Sprite, das auf den Bildschirm gebracht werden soll (PUT), die Nummer des Verfolgers und den Wert des Programmzählers, an dem die Ausfuhrung beginnen soll. Das erste Sprite wird auf den Bildschirm gebracht (PUT) und die Systeminformation wird im gewünschten Verfolger aufgebaut.

Parameter	Verwendung
COL	Bildschirmspalte am Anfang der Spur.
ROW	Bildschirmreihe am Anfang der Spur
KEY	Enthält die Nummer jenes Sprite, das das Verfolgungsprogramm enthält.
SP1	Erste Sprite, das auf den Bildschirm gebracht werden soll (PUT).
e1	Erster Programmzählerwert (1 = erste Anweisung im Verfolger)

Befehl	Vorgang
TPUT,e1	Vorfolger-KEY wird initialisiert und Sprite SP1 wird mit Hilfe des Verfolgersprite-KE auf die durch COL und ROW festgelegte Bildschirmposition gesetzt. Der Programm zähler im Verfolger-KEY wird auf den Wert des Ausdrucks e1 eingestellt. De Verfolger kann jetzt mit Hilfe des TMOV-Befehls ausgeführt werden.

Bewegen eines Verfolgers

Parameter	Verwendung
KEY	Die Nummer des Verfolgers, von dem eine Anweisung ausgeführt werden soll.
@V1	Optionaler Parameter, der die Adresse der ganzzahligen BASIC-VARIABLE darstellt die erhöht wird, wenn die Kollisonserfassung in Betrieb ist und eine Kollisior festgestellt wird.

Befehl	Vorgang
TMOV	Eine Anweisung des Verfolgers, dessen Nummer in KEY gehalten wird, wird ausgeführt
TMOV,@ V1	Eine Anweisung des Verfolgers, dessen Nummer in KEY gehalten wird, wird ausgefuhrt. Wenn die Kollisionserfassung in Betrieb ist (BIT 7 der Bewegungsart) und eine Kollision festgestellt wird, dann wird die ganzzahlige BASIC-Variable V1 erhöht.

Es gibt zwei Steuerbefehle – CPUT und CMOV. Diese sind analog zu den TPUT- uind TMOV-Befehlen, arbeiten aber mit Verfolgersätzen anstatt einzelnen Verfolgern. In jedem Fall wird die Information in Sprites gehalten. Der einzige notwendige Parameter ist also die Nummer des Sprite, das die Steuerinformationen enthält.

CPUT

Der CPUT-Befehl leitet alle Verfolger aus der im Datensprite enthaltenen Liste ein, dessen Nummer in SPN gehalten wird. Das Format für den Steuerbaustein, der die Einleitungsinformationen enthält, ist wie folgt:

BYTE	VERWENDUNG
0	Nummer des Verfolgers, der eingeleitet werden soll.
1	Bildschirmspalte, bei der die Einleitung beginnt.
2	Bildschirmreihe, bei der die Einleitung beginnt.
3	Nummer des ersten Sprite, das auf den Bildschirm gebracht wird (PUT).
4,5	16-Bit-Verschiebung in den Verfolger, wo die Ausführung beginnen soll.

Bytes 6 bis 11 enthalten die Informationen für den zweiten Verfolger, der eingeleitet werden soll, 12 bis 17 für den nächsten usw. Das Ende der Liste ist mit einem Null-Byte gekennzeichnet, so daß die Nummer des Sprite, das eingeleitet werden soll, als null gelesen wird.

Parameter	Verwendung
SPN	Nummer des Sprite, das die Einleitungsinformationen enthält.

Befehl	Vorgang
CPUT	Der Verfolgersatz, dessen Einleitungsinformationen in jenem Sprite enthalten sind, dessen Nummer in SPN gehalten wird, wird eingeleitet. Dies bedeutet, daß eine Reihe von TPUTs ausgeführt wird, wobei die im Steuerbaustein enthaltenen Parameter verwendet werden.

CMOV

Nachdem mit Hilfe von CPUT ein Satz von Verfolgern eingeleitet wurde, kann der Satz mit CMOV ausgeführt werden. Das Format für den Steuerbaustein ist wie folgt:

BYTE	Verwendung
0	Nummer des Verfolgers, der ausgeführt werden soll.
1	Dieses Byte wird auf 2 eingestellt, wenn Kollisionserfassung im Verfolger in Betrieb ist und eine Kollision stattgefunden hat. Wenn beim nächsten CMOV keine Kollision stattfindet wird es wieder auf 0 eingestellt.

Bytes 2 bis 3 halten den zweiten Verfolger, der ausgeführt werden soll, usw. Auch hier wird die Liste mit einer Null beendet.

Parameter	Verwendung
SPN	Die Nummer des Sprite, das die Liste der auszuführenden Verfolger enthält.

Befehl	Vorgang
CMOV	In jedem Verfolger, aus dem Datensprite, dessen Nummer in SPN gehalten wird, wird eine Anweisung ausgeführt.

Wenn ein Verfolger mit der Anweisung "start new tracker" (neuen Verfolger beginnen) ausgeführt wird, dann wird das Datensprite automatisch modifiziert, um die Nummer des neuen Verfolgers zu halten.

HINTERGRUNDAUSFÜHRUNG:

Eine der besonderen und leistungsstärksten Eigenschaften von LASER BASIC besteht in der Möglichkeit Befehle wiederholt unter Interrupt auszuführen. Dies bedeutet, daß eine Liste von Befehlen, jeder mit seinem eigenen Variablensatz, im Hintergrund ablaufen können (auf der Rahmenrücklauf synchronisiert), während das normale BASIC-Programm im Vordergrund abläuft. Durch diese Möglichkeit werden dem Anwender optimale Geschwindigeiten und gleichmäßige Programme geboten.

Befehl	Vorgang
ISET,@V$	Die Befehle und deren Variablensätze, die in der Kettenvariable V$ enthalten sind werden in die Hintergrundtabelle kompiliert. Das Format für die Kette ist wie folgt

Bytes	Verwendung
0.3	Erster Befehl
4	Variablensatz für ersten Befehl
5.8	Zweiter Befehl.
9	Variablensatz für zweiten Befehl.
.	
.	
.	
N,N+3	Letzter Befehl
N+4	Variablensatz für letzten Befehl.

ANMERKUNG: Der erforderlich Variablensatz wird unter Verwendung eines der 16 Buchstaben von A bis P gewählt. Das letzte Zeichen in der Kette muß ein # sein.

Beispiel: "WVR1AINVVBXMOVC#" würde in die Tabelle die Befehle WVR1, INVV und XMOV kompilieren. WVR1 würde SET 0 verwenden, INVV SET 1 und XMOV SET 2. Diese Befehle würden in der obigen Reihenfolge ausgeführt werden, und die Ausführung des Vordergrundbefehls würde solange angehalten werden, bis die Hintergrundaufgabe abgeschlossen ist.

Befehl	Vorgang
IRUN,e1	Die Ausführung des durch ISET kompilierten Hintergrundprogramms wird begonnen. Der Wert des BASIC-Ausdrucks e1 wird verwendet, um die Häufigkeit der Ausführung des Hintergrundprorams zu überwachen. Wenn e1 null ist, dann wird das Programm bei jedem Rucklauf (50 mal pro Sekunde) ausgeführt wenn e1 ist dann wird zwischen jeder Ausführung ein Rücklauf stattfinden usw., bis zu 65535 wo nur alle 20 Minuten eine Ausführung stattfinden wird.

ANMERKUNG: IRUN soll nie ausgeführt werden, ohne daß vorher ISET ausgeführt wurde. IRUN soll auch nie ausgeführt werden, wenn ein Hintergrundsprogramm bereits läuft. Um zu gewährleisten, daß kein Hintergrundprogramm ausgeführt wird, muß IENT benutzt werden (s. nächsten Abschnitt), um die Hintergrundaufgabe anzuhalten oder die "ESC" Taste gedruckt werden. Wenn die Ausführungszeit für die Aufgabe mehr als eine 50tel Sekunde beträgt und e1 0 ist, dann wird die Kontrolle vom Hintergrundprogramm nicht mehr zurückkehren. Dies wird unter Umständen gewünscht. Drücken der "ESC"-Taste wird allerdings immer die Ausführung der Hintergrundaufgabe anhalten und es wird dabei verhindert, daß das System blockiert wird. Fehler in den erweiterten, im Hintergrund laufenden Befehlen werden die Hintergrundaufgabe ebenfalls zum Anhalten bringen.

Befehl Vorgang

IEND Die Ausführung der Hintergrundaufgabe wird angehalten. Dieser Befehl muß immer ausgeführt werden, bevor ISET bzw. IRUN nochmals ausgeführt werden.

Eine Liste mit Befehlen, die im Hintergrund ausgeführt werden können, ist im Befehlsverzeichnis enthalten.

COMPILERBEZOGENEN BEFEHLE:

Mit Ausnahme von ADDR, der mit Verfolgungssprites verwendet wird, werden diese Befehle wahrscheinlich nur dann verwendet, wenn Sie ein LASER-BASIC-Programm, das Sie schreiben, kompilieren wollen, oder wenn Sie nicht genügend Platz haben.

Diese zusätzlichen Befehle werden benötigt, da die RSX-Befehlstabelle nicht zu, entgültigen kompilierten Programm gehört.

ADDR,@V1,@A$

Dieser Befehl ordnet der ganzzahligen BASIC-Variable V1 die Ausführungsadresse jenes Befehls zu, dessen Zeichenkette in A$ enthalten ist. Um z.B. die Ausführungsadresse von WVR1 zu ermitteln verwenden Sie:

```
A$="WVR1":|ADDR,@X%,@A$:PRINT X%
```

Der Befehl muß in Großbuchstaben eingegeben werden, da sonst "Illegal Command" (ungültiger Befehl) erscheint. Die ermittelte Adresse wird im allgemeinen als Information für Verfolgungssprites verwendet, wenn Steuercode 4 verwendet wird. ADDR kann vom Laser-BASIC-Compiler nicht kompiliert werden.

IPUT und IGET

Da die RSX-Tabelle in kompilierten Programmen nicht vorhanden ist, kann ISET auch nicht kompiliert werden. Dies würde die Ausführung von Hintergrundprogrammen verhindern und deshalb wurden zwei zusätzliche Befehle hinzugefügt.

IPUT

IPUT wird die laufende Interrupt-Tabelle (erzeugt durch ISET) in dem Sprite speichern, dessen Nummer in SPN gehalten wird.

IGET

IGET wird die Daten von der Interrupt-Tabelle wieder aus dem Sprite hervorholen, dessen Nummer in SPN gehalten wird.

Wenn Sie Ihr Laser-BASIC-Programm kompilieren wollen, dann brauchen Sie nur den ISET-Befehl ausführen, um die Interrupt-Tabelle aufzubauen, und anschließend IPUT ausführen, um die Information in einem Sprite zu speichern. Das Sprite kann gemeinsam mit den anderen Sprites in das kompilierte Programm geladen werden. Danach kann IGET verwendet werden, um die Tabelle wieder auf den ursprünglichen Zustand zu bringen. Out-Of-Memory-Fehler (Speicherplatz voll) treten auf, wenn das Sprite nicht groß genut ist, um die Tabelle unterzubringen (drei Bytes pro Befehl in der Interrupt-Liste plus ein Abgrenzungsbyte), oder wenn eine Tabelle geladen wird, die den Platz in der Interrupt-Tabelle (90 Bytes) überschreitet.

LASER-BASIC-TON

Diese letzte Abschnitt behandelt wahrscheinlich die schwierigste Funktion von Laser-BASIC. Das BASIC vom Schneider ermöglicht die Tonsteuerung in einer recht einfachen Art und Weise und diese erweiterte Möglichkeit sollte nur bei sehr fortgeschrittenen Anwendungsfällen verwendet werden. Wie bei den Verfolgungssprites wird auch hier nur geringer Störungsschutz geboten, und die Wirkung

von [illegible] unter [illegible] wohl dies in der Pra[illegible]ie Regel ist.

Die Philosophie, die hinter dem Tonsystem von Laser Basic steht, ist sehr ähnlich mit jener bei den Verfolgungssprites – das Datensprite enthält ein primitives Programm. In diesem Fall enthält der Satz 20 Anweisungen. Das Programm im Datensprite muß nicht unter Interrupt gelaufen werden. Bei den meisten Anwendungen ist dies jedoch bestimmt der Fall. Um unter Kontrolle eines Verfolgungssprite Ton laufen zu lassen, wird der PLAY-Befehl in ein normales Verfolgungssprite eingegeben (siehe Code 4 unter dem Abschnitt über Verfolgungssprite).

Befehl	Vorgang
PLAY,e1,e2	Der 16-Bit-Programmzähler im Tonprogramm, das in dem Sprite enthalten ist, dessen Nummer in KEY gehalten wird, ist auf e1 eingestellt. Die Häufigkeit der Befehlsausführung wird auf e2 eingestellt. Das eigentliche Tonprogramm wird aber nicht ausgeführt.
PLAY	Das Tonprogramm aus dem Sprite, dessen Nummer in KEY gehalten wird, wird ausgeführt

Parameter	Verwendung
KEY	Hält die Nummer des Sprite, das das Tonprogramm enthält.

Format für das Programm

BYTE	Verwendung
0,1	16-Bit-Programmzähler. Zeigt auf die nächste Anweisung im Datensprite und wird jedesmal um die Länge der Ausführung vorgerückt, wenn eine Ausführung stattfindet. Im Gegensatz zu Verfolgungssprites werden Tonprogramme solange ausgeführt, bis sie eine Anweisung erreichen, die die Ausführung anhält
2	Dies ist ein 8-Bit-Zähler, der jedesmal erhöht wird, wenn PLAY ausgeführt wird. Das Tonprogramm wird nur dann ausgeführt, wenn dieser Wert gleich hinzugefügt, da Tonprogramme gewöhnlich viel langsamer laufen als Verfolgungssprites und deshalb bei ausgewählten Verfolgerausführungen ausgeführt werden können.
3	Dies ist die 8-Bit-'Grenze', die die Ausführungsgeschwindigkeit des Tonprogramms festlegt. Eine Grenze von 0 erzwingt jedesmal eine Ausführung, wenn PLAY ausgeführt wird, ein Grenze von 1 erzwingt bei jeder zweiten Ausführung von PLAY eine Ausführung usw.

Nach diesen ersten vier Informationsbytes kommt das Programm selbst. Die erste Anweisung (Byte 4) entspricht einem Programmzählerwert von 1. Betrachten wir uns jetzt die eigentlichen Anweisungen:

CODE	ANWEISUNG	DATENLÄNGE (EINSCHLIESSLICH CODE)
0	SOUND	10
1	WAIT-SOUND	10
2	RESET	1
3	RELEASE	2
4	HOLD	1
5	CONTINUE	1
6	AMP-ENV	VARIABEL – BIS ZU18
7	TONE-ENV	VARIABEL – BIS ZU18
8	RE-RUN	1
9	JUMP	3
10	RE-LIM	3
11	CALL-CHANNEL	2
12	CALL-AMP-ENV	2

14	CALL-TONE-PERIOD	3
15	CALL-NOISE-PERIOD	2
16	CALL-INITIAL-AMP	2
17	CALL-DURATION	3
18	CALL-TONE-DURATION	5
19	STOP	1

Um die Operation jeder dieser Anweisungen besser zu verstehen, sollten Sie sich auf die Amsoft-Firmware bzw. die BASIC-Handbücher beziehen. Im nächsten Abschnitt werden diese Anweisungen jedoch ausreichend erklärt. Wo es notwendig ist, wird der Einstiegspunkt in die Firmware-JUMP-Tabelle angeführt.

SOUND – JUMP-Tabelle #BCAA

Diese Anweisung versucht der Tonwarteschlange eines bzw. mehrerer Kanäle einen 9-Byte-Ton hinzuzufügen. Falls eine der Warteschlangen belegt ist, wird keine Operation stattfinden, und das Tonprogramm wird bei der nächsten Anweisung fortgesetzt. Das Format für die 9 Bytes des Tones ist wie folgt:

BYTE	VERWENDUNG	POSITION IN DER LISTE VON BASIC-"TON"-BEFEHLEN
0	Kanalstatus	1
1	Amplitudenhüllkurve	5
2	Tonhüllkurve	6
3,4	Tonperiode	2
5	Schallperiode	7
6	Lautstärke	4
7,8	Dauer	3

Eine vollständige Beschreibung für den Bereich von jedem der oben erwähnten Parameter und deren Einsatz bei der Definierung des erzeugten Tones ist in Ihrem BASIC-Handbuch bzw. im "TON"-Abschnitt dieses Handbuchs erhältlich.

WAIT-SOUND

Diese Anweisung funktioniert gleich wie die Tonanweisungen und verwendet auch dieselben Parameter. Sie unterscheidet sich allerdings in einer Hinsicht. Falls eine der Warteschlangen, die die Anweisung verwenden möchte, belegt ist, wird die Ausführung des PLAY-Befehls abgebrochen und der Programmzähler verbleibt bei der laufenden 'WAIT-SOUND-Anweisung. Die 'WAIT-SOUND' wird praktisch immer anstelle der 'SOUND'-Anweisung verwendet.

RESET – JUMP-Tabelle #BCA7

Diese Anweisung wird alle individuell gehaltenen Töne freigeben (RELEASE). Sie benutzt ein Datenbyte, welches den Kanal bestimmt, der freigegeben werden soll. Von diesem Byte werden nur 3 Bits gebraucht – siehe BASIC-Befehl RELEASE.

HOLD – JUMP-Tabelle #BCB6

Diese Anweisung friert sämtliche Töne. Diese werden durch Ausführung von SOUND, WAIT-SOUND, RELEASE bzw. CONTINUE automatisch neu gestartet.

CONTINUE – JUMP-Tabelle #BCB9

Startet alle angehaltenen Töne neu

AMP-ENV – JUMP-Tabelle #BCBC

Diese Anweisung ist analog zum ENV-Befehl von BASIC. Sie gebraucht bis zu 18 Datenbytes und das Format für den Datenblock ist wie folgt:

0	Hüllkurvennummer
1	Anzahl der Abschnitte
2,3,4	Schrittzählung, Schrittgröße, Pausenzeit für Abschnitt 1
5,6,7	Schrittzählung, Schrittgröße, Pausenzeit für Abschnitt 2
.	
.	
.	
14,15,16	Schrittzählung, Schrittgröße, Pausenzeit für Abschnitt 5

Beachten Sie, daß die Daten nur für die Anzahl der in Byte 1 angegebenen Abschnitte vorgesehen sind. Dies ist daher eine Anweisung mit veränderlicher Länge. Eine komplette Beschriebung der Parameter und deren Anwendung ist erhältlich vom ENV-Befehl in BASIC.

TONE-ENV – JUMP-Tabelle #BCBF

Diese Anweisung ist analog zum ENT-Befehl von BASIC. Sie gebraucht bis zu 18 Datenbytes und das Format für den Datenblock ist wie folgt:

BYTE	VERWENDUNG
0	Hüllkurvennummer
1	Anzahl der Abschnitte
2,3,4	Schrittzahlung, Schrittgröße, Pausenzeit für Abschnitt 1
5,6,7	Schrittzahlung, Schrittgröße, Pausenzeit für Abschnitt 2
.	
.	
.	
14,15,16	Schrittzählung, Schrittgröße, Pausenzeit für Abschnitt 5

Die Daten haben wiederum eine veränderliche Länge und müssen der Anzahl von Abschnitten entsprechen, die von Byte 1 bestimmt werden. Eine ausführliche Beschreibung bietet der BASIC-Befehl ENT.

RE-RUN

Diese Anweisung läßt die Kontrolle auf die erste Anweisung im Programm springen und die Ausführung von diesem Punkt fortsetzen. Diese Funktion ermöglicht es, Tonprogramme unendlich zu wiederholen.

JUMP

Diese 3-Byte-Anweisung ist analog zum GOTO-Befehl von BASIC bzw. zur JP-Anweisung des Z80. Die beiden Bytes, die den 16-Bit-Programmzählerwert bilden, zu dem gesprungen wird, sind in der Reihenfolge LSB, MSB, wie alle anderen 16-Bit-Werte, die in diesem Abschnitt verwendet werden.

RE-LIM

Diese Anweisung wurde vorgesehen, um die Gesamtgeschwindigkeit und die Ladezone des Programms zu steuern. Sie gebraucht zwei Datenbytes, die den Bytes 2 und 3 des Tonprogramms übergeben werden. Dadurch wird die Programmzählung und -grenze neu eingestellt. Wenn eine kleinere Grenze übergeben wird, kann die Ausführungshäufigkeit des Programms erhöht werden (obgleich die Melodie dadurch nicht schneller spielen wird!) und umgekehrt. Falls ein Programmzählwert übergeben wird, der gleich groß ist wie der Grenzwert, dann wird das Programm beim nächsten Aufruf ausgeführt

CALL-Befehle

Es gibt insgesamt 7 CALL-Anweisungen. Um diese verwenden zu können, müssen Sie sich

...weisung (Code 1) ist, da sonst "ILLEGAL TRACKER CODE" (ungültiger Verfolgungscode) erzeugt wird. Wie bereits erwähnt besteht eine SOUND-Anweisung aus 9 Bytes. Diese Anweisungen modifizieren 1 bzw. mehrere Bytes in der SOUND-Anweisung und kehren danach zu der Programmstelle zurück, von der sie ursprünglich aufgerufen wurden. Falls CALL-TONE-PERIOD bzw. CALL-TONE-DURATION ausgeführt wird, dann wird SOUND bzw. WAIT-SOUND automatisch ausgeführt. Falls die aufgerufene Anweisung eine SOUND-Anweisung ist, wird die Kontrolle bei der nächsten Anweisung nach CALL fortgesetzt, egal ob der Ton (SOUND) der Warteschlange hinzugefügt wurde oder nicht. Wenn die Anweisung eine WAIT-SOUND-Anweisung war, dann wird die Kontrolle nur dann bei der nächsten Anweisung nach CALL fortgesetzt, wenn der Ton der Warteschlange hinzugefügt wurde. Falls nicht, wird der Programmzähler bei CALL stehenbleiben und das Programm wird abgebrochen. Wenn dies der Fall ist, dann wird diese CALL-Anweisung bei der nachsten Ausführung von PLAY mit einem Zählwert gleich dem Grenzwert zuerst ausgeführt.

CALL-CHANNEL

Ruft die erste Anweisung im Programm auf, die eine SOUND- bzw. WAIT-SOUND-Anweisung sein muß. Der Kanalstatus im 9-Byte-SOUND-Block wird durch den neuen Kanalstatus ersetzt und der Ton wird ausgeführt. Der neue Kanalstatus verbleibt im Datenblock nachdem die Kontrolle zurückkehrt. Eine SOUND- bzw. WAIT-SOUND-Anweisung wird allerdings nicht ausgeführt.

CALL-AMP-ENV

Diese funktioniert genauso wie CALL-CHANNEL. Die neuen Daten sind aber durch eine Lautstärkenhüllkurvennummer dargestellt.

CALL-TONE-ENV

Diese funktioniert genauso wie CALL-AMP-ENV. Die neuen Daten werden aber durch eine Tonhüllkurvennummer dargestellt.

CALL-TONE-PERIOD

Diese funktioniert genauso wie CALL-TONE-ENV. Die Daten sind diesmal aber 2 Bytes lang und stellen die Tonperiode dar. Weiters wird diese Anweisung SOUND bzw. WAIT-SOUND mit der modifizierten TON-PERIOD ausführen.

CALL-NOISE-PERIOD

Diese funktioniert genauso wie CALL-TONE-ENV. Die Daten sind diesmal aber 1 Byte lang und stellen die Schallperiode dar.

CALL-INITIAL-AMP

Funktioniert wie CALL-NOISE-PERIOD. Die Daten stellen aber die Ausgangslautstärke dar (die von der Lautstärkenhüllkurve geändert werden kann, falls eine bestimmt wurde).

CALL-DURATION

Funktioniert wie die anderen Befehle. Ihre beiden Datenbytes geben allerdings die neue Länge an.

CALL-TONE-DURATION

Funktioniert genauso wie CALL-TONE-PERIOD. Diese Anweisung hat aber 4 Datenbytes, die ersten beiden stellen eine neue Länge dar und das dritte und vierte stellen eine neue Tonperiode dar. Wie bei CALL-TONE-PERIOD wird die aufgerufene SOUND-bzw. WAIT-SOUND-Anweisung ausgeführt.

ANMERKUNG: Die beiden Bytenummern werden je nach Wichtigkeitsgrad eingegeben.

Diese Anweisung wird das Tonprogramm abbrechen und das Programm zu jener Stelle zurückstellen, wo die Ausführung des PLAY-Befehls begann (gewöhnlich ein Verfolger). Das Programm wird verlassen und der Programmzähler verbleibt bei der Anweisung nach STOP, welche als nächste Anweisung zur Ausführung gelangt, wenn PLAY wieder ausgeführt wird.

Hiermit ist der letzt Abschnitt über Laser-BASIC-Befehle im Detail zu Ende. Weitere Beispiele interruptgesteuerten Ton sind in dem Abschnitt über TON im Hauptteil dieses Handbuchs enthalt

Fehler 1 – ** SPN TOO HIGH **

Tritt auf, wenn eine Spritenummer mit einem Wert größer als SMAX verwendet wird. Die Variablen, die die Spritenummer enthalten sind SPN, SP1, SP2, SP3, SP4 und KEY Spritenummer werden auch in Verfolgungssprites gehalten. Versichern Sie sich, ob Sie den richtigen SET verwenden, insbesondere bei der Verwendung von Hintergrundprogrammen, Verfolgungssprites, EXXV, ASTV, AVTS und ESAV.

Fehler 2 – ** SPN EXISTS **

Tritt auf, wenn man versucht eine Sprite in die Tabelle einzureihen, dessen Nummer bereits zugewiesen wurde. Dieser Fehler ist möglich mit CSPR, HRSP, RNUM, ADNM und MSPR. Er läßt sich gewöhnlich sehr leicht identifizieren.

Fehler 3 – ** SPN OF ZERO **

Dieser Fehler tritt auf, wenn man versucht eine Spritenummer von null zu verwenden. Dies ist in vielen Fällen möglich. Man untersucht hierbei am besten die Werte in SPN, SP1, SP2, SP3, SP4 und KEY. Obwohl dieser Fehler sehr viele Ursachen haben kann, ist es gewöhnlich ziemlich klar, wo man suchen muß.

Fehler 4 – ** SMAX OF ZERO **

Dieser Fehler tritt auf, wenn ein SMAX von null als erster Parameter von SSPR gegeben wird. Dieser Fehler tritt nicht sehr häufig auf, da SSPR nur selten verwendet wird.

Fehler 5 – ** ZERO LENGTH DATA **

Dieser Fehler tritt sehr selten auf und signalisiert, wenn man versucht einem Code mit einer Länge von 0 blockweise zu bewegen. Dies kann vorkommen, wenn man versucht eine leere Spritedatei zu versetzen bzw. wenn IPUT/IGET verwendet werden, um eine leere Interruptliste zu bewegen. Dieser Fehler verweist jedoch im allgemeinen darauf, daß eine Spritetabelle bzw. Systemvariablen verstümmelt wurden. Es ist nicht ratsam hier fortzusetzen, es sei denn, Sie sind sich über die Ursache des Fehlers sicher.

Fehler 6 – ** SPN DOESN'T EXIST **

Auch dieser Fehler kann mehrere Ursachen haben. Durch Untersuchung der Variablen SPN, SP1, SP2, SP3, SP4 und KEY kann man gewöhnlich feststellen, wo der Fehler aufgetreten ist.

Fehler 7 – ** CAN'T – MASK **

Es wurde versucht ein Sprite zu MASKieren, neu zu MASKieren bzw. zu entMASKieren, das eine 'ungerade' statt eine 'gerade' Breite hat (in Bytes).

Fehler 8 – **ILLEGAL TRACKER TERMINATOR **

In einem Verfolger- bzw. Tonprogramm wurde ein ungültiger Steuercode gefunden.

Fehler 9 – ** OUT OF MEMORY **

Es wurde versucht eine Operation auszuführen, die mehr Platz in Anspruch nimmt, als zwischen MBOT (dies sollte HIMEM + 1 sein) und der Unterseite der Sprites erhältlich ist, oder es gibt keinen Stapelspeicherplatz mehr. Um dies zu umgehen, müssen Sie HIMEM mit Hilfe des MEMORY-Befehls (gefolgt von einem MSET-Befehl) reduzieren, eines oder mehrere Sprites löschen bzw. Sprites nach oben verschieben (falls dies möglich ist). Dieser Fehler kann unter Umständen von FILL stammen, wenn nicht genügend Stapelplatz vorhanden ist. Es gibt hierbei keinen Ausweg, da der Platz, der einem Stapel zugewiesen wird, fixiert ist. Die folgenden Befehle benötigen freien Speicherplatz:

IPUT und IGET können diesen Fehler ebenfalls verursachen. Siehe hierzu "Compilerbezogene Befehle".

Anmerkung: Die 'LOAD'-Befehle von BASIC benutzen einen Pufferspeicher, der nach dem Laden nicht mehr erforderlich ist. Dies bedeutet, daß Sie HIMEM (vergessen Sie nicht MSET!!) reduzieren können, nachdem Ihr BASIC-Programm geladen wurde. Sprites können dann in einen größeren Platz geladen werden.

Fehler 10 – ** ILLEGAL FILENAME **

Dieser Fehler tritt auf, wenn man versucht, Spritedateien zu laden, zu speichern oder zu mischen, und dabei einen Dateinamen verwendet, der nicht mit "SPR" bzw. "SPR.BAK" endet. Dateinamen müssen in Großbuchstaben geschrieben werden. Wenn während einer dieser drei Operationen ein Fehler auftritt, werden Sie unter Umständen feststellen, daß der Dateiname geändert wurde. Wenn dies der Fall ist, müssen Sie den Dateinamen neu schreiben.

Fehler 11 – ** PARAMETER ERROR **

Dieser Fehler tritt auf, wenn einem Laser-BASIC-Befehl die falsche Anzahl an Parametern nachgestellt wird oder wenn einem der SUM-Befehle mehr als 8 Parameter nachgestellt werden.

Fehler 12 – ** NO SPRITES **

Dieser Fehler tritt auf, wenn man versucht, eine leere Spritedatei zu speichern.

Fehler 13 – ** OUT OF RANGE **

Dieser Fehler wird auftreten, wenn man versucht eine der Laser-BASIC-Variablen mit einem ungültigen Wert zuzuweisen. Dieser Fehler wird ebenfalls auftreten, wenn man versucht ein Sprungfenster in einem Sprite aufzubauen, das nicht eine Höhe von 1 und eine Breite von 4 hat. Dieser Fehler wird auch noch vorkommen, wenn ASTV, AVTS oder ESAV mit Sprites verwendet werden, die nicht eine Höhe von 1 und eine Breite von 20 haben.

Fehler 14 – ** ILLEGAL COMMAND **

Dieser Fehler kommt vor, wenn der ISET-Befehl verwendet wird, und bedeutet, daß einer der Befehle in der Kette nicht vorhanden ist bzw. in Kleinbuchstaben eingegeben wurde, oder '#' am Ende der Kette fehlt, oder ein SET ohne einem der Buchstaben 'A' bis 'P' (in Großbuchstaben) gewählt wurde. Er kann auch auftreten, wenn der ADDR-Befehl verwendet wird.

Nachstehend folgt eine Zusammenfassung der erweiterten Befehle von Laser BASIC. Die Nummer, die hinter dem Befehl in der Tabelle steht, bezeichnet die Befehlsklasse, nachfolgend ist eine Beschreibung der einzelnen Klassen.

KLASSE 1

Befehle in dieser Klasse können sowohl in MODE 2 (2-Farbenmodus) als auch in MODE 0 und 1 verwendet werden. Diese Befehle haben keine Kollisionserfassungsoption, sie können nicht mit Verfolgungssprites verwendet werden und können nicht im Hintergrund ausgeführt werden (siehe ISET) und sie haben keine Wiederholoption. Die folgenden Befehle sind in Klasse 1 enthalten:

ADDR	COL	COLQ	DEEK	DOKE	HGT	HGTQ	IK1	IK1Q	ISET
ISPR	KBFN	KEY	KEYQ	LEN	LENQ	MSET	MSPR	NPX	NPXQ
PSPR	ROW	ROWQ	RSPR	SCL	SCNP	SCNS	SCNV	SET	SETQ
SP1	SP1Q	SP2	SP2Q	SP3	SP3Q	SP4	SP4Q	SPN	SPNQ
SRW	SRWQ	SSPR	SUMP	SUMS	SUMV	TPUT	XCL	XCLQ	

KLASSE 2

Befehle in dieser Klasse können nicht im MODE 2 verwendet werden und können nur im MODE 0 und 1 verwendet werden. Sie verfügen nicht über eine Kollisionserfassungsoption, können aber als Anweisungen in Verfolgungssprites verwendet werden. Sie können nicht im Hintergrund ausgeführt werden und verfügen nicht über eine Wiederholoption.

FILL	HRSP	MASK	MGXP	MGXS	MGXV	ONHI	ONLO	RMSK	SPNV

KLASSE 3

Befehle in dieser Klasse können im MODE 2 und im MODE 0 und 1 verwendet werden. Sie haben keine Kollisionserfassungsoption, können in einem Verfolgungssprite verwendet werden, können nicht im Hintergrund ausgeführt werden und verfügen nicht über eine Wiederholoption.

ADNM	CSPR	DMSK	DSPR	ESPR	GSPR	IEND	MGYS	MGYV	RNUM

KLASSE 4

Befehle in dieser Klasse können in allen Grafikmodi verwendet werden, sie haben keine Kollisionserfassungsoption, können als Anweisungen in Verfolgungssprites verwendet werden, können im Hintergrund ausgeführt werden, aber verfügen nicht über eine Wiederholoption.

ASTV	AVTS	BILD	BWST	CLHI	CLLO	CLSP	CLSS	CLSV	CPUT
ESAV	EXXV	FIPS	FIPV	FOPS	FOPV	IGET	INVP	INVS	INVV
IPUT	PLAY	SCLS	STCP	STCS	STCV	SWPS			

KLASSE 5

Befehle in dieser Klasse funktionieren nicht in MODE 2, haben keine Kollisionserfassungsoption, können in Verfolgern ausgeführt werden, können im Hintergrund ausgeführt werden, aber haben keine Wiederholoption.

MIRP	MIRS	MIRV	MORP	MORS	MORV	SETP	SETS	SETV

KLASSE 6

Befehle in dieser Klasse funktionieren nicht in MODE 2, haben aber eine Kollisionserfassungsoption, können in Verfolgern ausgeführt werden, konnenim Hintergrund ausgeführt werden, aber haben keine Wiederholoption.

BGET	BPUT	FSWP	GMBH	GMIF	GTBH	GTIF	GWBH	GWIF	PMBH
PMIF	PTBH	PTIF	PWBH	PWIF					

KLASSE 7

Befehle in dieser Klasse funktionieren in allen Grafikmodi, haben eine Kollisionserfassungsoption, können im Hintergrund ausgeführt werden, habe aber keine Wiederholoption.

GMBL	GMND	GMOR	GMXR	GTBL	GTND	GTOR	GTXR	GWBL	GWND
GWOR	GWXR	PMBL	PMND	PMOR	PMXR	PTBL	PTND	PTOR	PTXR
PWBL	PWND	PWOR	PWXR						

KLASSE 8

Befehle in dieser Klasse funktionieren nicht im Grafikmodus 2, haben keine Kollisionserfassungsoption, können aber in Verfolgern ausgeführt werden, können im Hintergrund ausgeführt werden und verfügen über eine Wiederholoption.

SPL1	SPR1	SSL1	SSR1	SVL1	SVR1	WPL1	WPR1	WSL1	WSR1
WVL1	WVR1								

KLASSE 9

Befehle in dieser Klasse funktionieren im Grafikmodus 2, haben keine Kollisionserfassungsoption, können aber in Verfolgern ausgeführt werden, sie können im Hintergrund ausgeführt werden und haben eine Wiederholoption.

CMOV	SPL4	SPL8	SPR4	SPR8	SPVN	SSL4	SSL8	SSR4	SSR8
SSVN	SVL4	SVL8	SVR4	SVR8	SVVN	WPL4	WPL8	WPR4	WPR8
WPVN	WSL4	WSL8	WSR4	WSR8	WSVN	WVL4	WVL8	WVR4	WVR8
WVVN									

KLASSE 10

Befehle in dieser Klasse funktionieren nicht in MODE 2, haben aber eine Kollisionserfassungsoption, sie können in Verfolgern ausgeführt werden sie können im Hintergrund ausgeführt werden und habe eine Wiederholoption.

BBNC	BMOV	BMJV	FBNC	FMOV	FMJV

KLASSE 11

Befehle in dieser Klasse funktionieren in MODE 2, haben eine Kollisionserfassungsoption und können in Verfolgern ausgeführt werden, sie können im Hintergrund ausgeführt werden und haben eine Wiederholoption.

TMOV	WBNC	WMOV	WMJV	XBNC	XMOV	XMVJ

KLASSENOPTIONEN

Nachfolgend ist eine Zusammenfassung der erhältlichen Optionen und eine Zusammenstellung der Klassen, die diese unterstützen.

MODE 2

Samtliche Laser-BASIC-Befehle können in MODE 0 (16-Farbenmodus) und MODE 1 (4-Farbenmodus) ausgeführt werden. Es können allerdings nur ausgewahlte Befehle in MODE 2 (2-Farbenmodus) korrekt ausgeführt werden. Die folgenden Klassen werden in MODE 2 funktionieren:

KLASSE 1, KLASSE 3, KLASSE 4, KLASSE 7, KLASSE 9, KLASSE 11

Laser-BASIC-Befehle, die Daten am Bildschirm und zwischen Sprites hin- und herbewegen, haben eine Kollisionserfassungsoption. Diese Option wird aber nicht bei Daten in MODE 2 funktionieren. Folgende Klassen unterstützen diese Option:

KLASSE 6, KLASSE 7, KLASSE 10, KLASSE 11.

Verfolgungsoption

Laser-BASIC-Befehle können von Verfolgungssprites aus ausgeführt werden (siehe Verfolgungssprites, Steuercode 4). Nur gewisse Befehle können auf diese Art und Weise ausgeführt werden und nur Befehle in den folgenden Klassen sollten hierfür verwendet werden:

KLASSE 2 , KLASSE 3 , KLASSE 4 , KLASSE 5 , KLASSE 6 , KLASSE 7 , KLASSE 8 , KLASSE 9 , KLASSE 10, KLASSE 11.

Hintergrundausführung

Bestimmte Befehle können in eine Interrupttabelle kompiliert werden und im Hintergrund unter Interrupt ausgeführt werden. Die folgenden Befehlsklassen unterstützen diese Option:

KLASSE 4, KLASSE 5, KLASSE 6, KLASSE 7, KLASSE 8, KLASSE 9, KLASSE 10, KLASSE 11.

Wiederholoption

Gewisse Befehle können wiederholt in einer Maschinencodeschleife mit oder ohne Rahmenrücklaufsynchronisierung ausgeführt werden. Die folgenden Klassen unterstützen diese Option:

KLASSE 8, KLASSE 9, KLASSE 10, KLASSE 11.

BEFEHL/ KLASSE		PARAMETER	VORGANG
ADDR	1	@V1,@V$	Die Ausführungsadresse des Befehls in der 4-Zeichenkette V$ wird der ganzzahligen BASIC-Variable V1 zugeordnet.
ADNM	3	SPN	Alle vorhandenen Spritenummern werden um den in SPN gehaltenen Wert erhöht.
ASTV	4	SET,SPN	Die 20 Bytes in Sprite SPN werden dem laufenden Variablensatz zugeordnet.
AVTS	4	SET,SPN	Der laufende Variablensatz wird em Sprite SPN zugeordnet.
BBNC	10	SP1,SP2,SP3, SP4,HGT,LEN, COL,ROW,KEY @V1,e1,e2	Sprung 'hinter' Bildschirmdaten.
BGET	6	SPN,COL,ROW, @V1	Entfernt ein Sprite, das vorhin auf den Bildschirm gebracht wurde ('BPUT').
BILD	4	COL,ROW,SPN, SP1,KEY	Expandiert BIT-Muster.
BMOV	10	SP1,SP2,SP3, SP4,HGT,LEN, COL,ROW,@V1, e1,e2	Lineare Bewegung 'hinter' Bildschirmdaten
BMSK	10	SP1,SP2,SP3, SP4,HGT,LEN, COL,ROW,KEY, @V1,e1,e2	Bewegung unter Tabular-Kopiersteuerung hinter Bildschirmdaten.
BPUT	6	SPN,COL,ROW, @V1	'PUT' (bringt) ein Sprite 'hinter' Bildschirmdaten.
BWST	4	KEY,COL,ROW, HGT,LEN	Sprungfensterdaten werden im Sprite KEY aufgebaut.
CLHI	4		Stellt die Software auf 160-Spaltenmodus ein.
CLLO	4		Stellt die Software auf 80-Spaltenmodus ein.
CLSP	4	SPN,COL,ROW, HGT,LEN	Stellt Spritefenster auf INK 0 ein.
CLSS	4	SPN	Stellt das ganze Sprite auf INK 0 ein.
CLSV	4	COL,ROW,HGT, LEN	Stellt Bildschirmfenster auf INK 0 ein.
CMOV	9	SPN	Führt in jedem in SPN gelisteten Verfolger eine Anweisung aus.
COL	1	e1	Der Grafikvariable COL wird der Ausdruckswert e1 zugewiesen.
COLQ	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert der Grafikvariable COL zugewiesen.
CPUT	4	SPN	Alle in SPN gelisteten Verfolger werden eingeleitet.
CSPR	3	SPN,HGT,LEN	Sprite SPN mit Höhe HGT und Länge LEN wird erzeugt.
DEEK	1	e1,@V1	Der ganzzahligen BASIC-Variable V1 wird der 16-Bit-Inhalt der Adresse e1,e1+1 zugewiesen.
DMSK	3	SPN	Sprite SPN wird entmaskiert.
DOKE	1	e1,e2	Der 16-Bit-Wert e2 wird in der Reihenfolge LSB,MSB in e1 und e1+1 gePOKEt.
DSPR	3	SPN	Sprite SPN wird gelöscht.
ESAV	4	SET,SPN	Der laufende SET wird mit dem Inhalt von Sprite SPN vertauscht.
ESPR	3	SPN	Spritetabellenplatz wird expandiert/verkleinert.
EXXV	4		Laufende und alternative SET-Werte werden vertauscht.
FBNC	10	SP1,SP2,SP3, SP4,HGT,LEN, COL,ROW,KEY, @V1,e1,e2	Sprung 'vor' Bildschirmdaten.
FILL	2	XCL,ROW,IK1	Die Form wird mit INK IK1 gefüllt.
FIPS	4	SPN	Sprite SPN wird vertikal reflektiert.

FMOV	10	SP1,SP2,SP3, SP4,HGT,LEN, COL,ROW,@V1 e1,e2	Lineare Bewegung 'vor' Bildschirmdaten.
FMVJ	10	SP1,SP2,SP3, SP4,HGT,LEN, COL,ROW,KEY, @V1,e1,e2	Bewegung unter Tastatur-/Joysticksteuerung 'vor' Bildschirmdaten
FOPS	4	SPN	Sprite SPN wird vertikal symmetrisch gemacht.
FOPV	4	COL,ROW,HGT LEN	Bildschirmfenster wird vertikal symmetrisch gemacht.
FREE	1	@V1	Freier Speicherplatz wird berechnet und das Ergebnis wird der ganzzahligen BASIC-Variable V1 zugewiesen.
FSWP	6	COL,ROW,SPN, @V1	Sprite- und Bildschirmdaten werden vor Bildschirmdaten ausgetauscht.
GMBH	6	SP1,SP2,SRW SCL,@V1	Sprite SP1 wird 'hinter' Fenster gesetzt.
GMBL	7	SP1,SP2,SRW, SCL,@V1	Sprite SP1 wird blockweise in das Fenster SP2 bewegt.
GMIF	6	SP1,SP2,SRW, SCL,@V1	Sprite SP1 wird 'vor' Fenster SP2 gesetzt. Daten in SP1
GMND	7	SP1,SP2,SRW, SCL,@V1	Sprite SP1 wird mittels AND mit Fenster SP2 verknüpft.
GMOR	7	SP1,SP2,SRW, SCL,@V1	Sprite SP1 wird mittels OR mit Fenster SP2 verknüpft .
GMXR	7	SP1,SP2,SRW, SCL,@V1	Sprite SP1 wird mittels XOR mit Fenster SP2 verknüpft.
GSPR	3	@V$	Sprites werden vom Band/von Diskette geladen.
GTBH	6	SPN,COL,ROW @V1	Bildschirmdaten werden 'hinter' Spritedaten gesetzt.
GTBL	7	SPN,COL,ROW @V1	Bildschirmfenster wird blockweise in das Sprite bewegt.
GTIF	6	SPN,COL,ROW, @V1	Bildschirmdaten werden 'vor' die Spritedaten geholt.(GET)
GTND	7	SPN,COL,ROW @V1	Bildschirmdaten werden mittels AND in das Sprite gesetzt.
GTOR	7	SPN,COL,ROW @V1	Bildschirmdaten werden mittels OR in das Sprite gesetzt.
GTXR	7	SPN,COL,ROW @V1	Bildschirmdaten werden mittels XOR in das Sprite gesetzt.

		'CL,SRW,HGT EN,@V1	
GWBL	7	SPB,COL,ROW SCL,SRW,HGT LEN,@V1	Bildschirmfenster wird blockweise in das Spritefenster bewegt.
GWIF	6	SPN,COL,ROW SCL,SRW,HGT LEN,@V1	Bildschirmfenster wird 'vor' das Spritefenstergeholt (GET).
GWND	7	SPN,COL,ROW SCL,SRW,HGT LEN,@V1	Bildschirmfenster wird mittels AND in das Spritefenster gesetzt.
GWOR	7	SPN,COL,ROW SCL,SRW,HGT LEN,@V1	Bildschirmfenster wird mittels OR in das Spritefenster gesetzt.
GWXR	7	SPN,COL,ROW SCL,SRW,HGT LEN,@V1	Bildschirmfenster wird mittels XOR in das Spritefenster gesetzt.
HGT	1	e1	Der Grafikvariable HGT wird der Wert e1 zugewiesen.
HGTQ	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert in der Grafikvariable e1 zugewiesen.
HRSP	2	SPN	In SPN und SPN+1 wird ein Hires-Paar erzeugt.
IEND	3		Hintergrundprogramm wird abgebrochen.
IGET	4	SPN	Daten vom Sprite SPN werden in die Interrupt-Tabelle bewegt.
IK1		e1	Der Grafikvariable IK1 wird der Wert von e1 zugeordnet.
IK1Q	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert in der Grafikvariable IK1 zugewiesen.
IK2	1	e1	Der Grafikvariable IK2 wird der Wert von e1 zugewiesen.
IK2Q	2	@	Der ganzzahligen BASIC-Variable V1 wird der Wert in der Grafikvariable IK2 zugewiesen.
INVP	4	SPN,COL,ROW HGT,LEN	Spritefenster wird invertiert (1er Komplement).
INVS	4	SPN	Das ganze Sprite wird invertiert (1er Komplement).
INVV	4	COL,ROW,HGT LEN	Bildschirmfenster wird invertiert (1er Komplement).
IPUT	4	SPN	Die Interrupt-Tabelle wird in das Sprite SPN bewegt.
IRUN	1	e1	Ablauf des Hintergrundprogramms wird mit Ausführungsabstand e1 eingestellt.
ISET	1	@V$	Sämtliche Befehle/Sätze in der Kette V$ werden in die Interrupt-Tabelle compiliert.

		@,V1,@V2,@V3 @V4	
KBFN	1	KEY,@V1	Die ganzzahlige BASIC-Variable V1 wird erhöht, wenn KEY gedrückt wird.
KEY	1	e1	Der Grafikvariable KEY wird der Wert von e1 zugewiesen.
KEYQ	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert in der Grafikvariable KEY zugeordnet.
LEN	1	e1	Der Grafikvariable LEN wird der Wert von e1 zugewiesen.
LENQ	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert der Grafikvariable LEN zugewiesen.
MASK	2	SPN	Sprite SPN wird in ein MASKiertes Sprite umgegwandelt.
MGPX	2	SPN,COL,ROW HGT,LEN	Spritefenster wird in X-Richtung expandiert.
MGXS	2	SPN	Das ganze Sprite wird in X-Richtung expandiert.
MGXV	2	COL,ROW,HGT LEN	Das Bildschirmfenster wird in X-Richtung expandiert.
MGYS	3	SPN	Das ganze Sprite wird in Y-Richtung expandiert.
MGYV	3	COL,ROW,HGT LEN	Bildschirmfenster wird in Y-Richtung expandiert.
MIRP	5	SPN,COL,ROW HGT,LEN	Spritefenster wird horizontal gespiegelt.
MIRS	5	SPN	Das ganze Sprite wird horizontal gespiegelt.
MIRV	5	COL,ROW,HGT LEN	Bildschirmfenster wird horizontal gespiegelt.
MORP	5	SPN,COL,ROW HGT,LEN	Spritefenster wird horizontal symmetrisch gemacht.
MORS	5	SPN	Das ganze Sprite wird horizontal symmetrisch gemacht.
MORV	5	COL,ROW,HGT LEN	Bildschirmfenster wird horizontal symmetrisch gemacht.
MSET	1	e1	Die niedrigste von Laser-BASIC verwendbare Adresse wird auf e1 eingestellt.
MSPR	1	@V$	Spritedateien werden gemischt.
NPX	1	e1	Der Grafikvariable NPX wird der Wert von e1 zugeordnet.
NPXQ	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert in der Grafikvariable NPX zugeordnet.
ONHI	2		Laser-BASIC wird auf 4-Farbenbetrieb eingestellt.
ONLO	2		Laser BASIC wird auf 16-Farbenbetrieb eingestellt

PEN	4	KEY,e1,e2	Tonprogramm wird ausgeführt.
PMBH	6	SP1,SP2,SCL SRW,@V1	Daten in Fenster SP2 werden hinter die Daten in SP1 geholt (GET).
PMBL	7	SP1,SP2,SCL SRW,@V1	Fenster in SP2 wird blockweise in das Sprite SP1 bewegt
PMIF	6	SP1,SP2,SCL SRW,@V1	Daten in Fenster SP2 werden 'vor'die Daten in SP1 gebrach (PUT).
PMND	7	SP1,SP2,SCL SRW,@V1	Fenster in SP2 wird mittels AND in das Sprite SP1 gesetzt.
PMOR	7	SP1,SP2,SCL SRW,@V1	Fenster in SP2 wird mittels OR in das Sprite SP1 gesetzt.
PMXR	7	SP1,SP2,SCL SRW,@V1	Fenster in SP2 wird mittels XOR in das Sprite SP1 gesetzt.
PSPR	1	@V1	Spritedatei wird auf BAND/DISKETTE geladen (Put).
PTBH	6	SPN,COL,ROW @V1	Sprite wird 'hinter' Bildschirmdaten gesetzt (PU).
PTBL	7	SPN,COL,ROW @V1	Sprite wird blockweise auf den Bildschirm bewegt.
PTIF	6	SPN,COL,ROW @V1	Sprite wird 'vor' die Bildschirmdaten gesetzt (PUT).
PTND	7	SPN,COL,ROW @V1	Sprite wird mittels AND mit Bildschirmdaten verknüpft.
PTOR	7	SPN,COL,ROW @V1	Sprite wird mittels OR mit Bildschirmdaten verknüpft.
PTXR	7	SPN,COL,ROW @V1	Sprite wird mittels XOR mit Bildschirmdaten verknüpft.
PWBH	6	SPN,COL,ROW SCL,SRW,HGT LEN,@V1	Spritefenster wird 'hinter' Bildschirmfenster gesetzt (PUT).
PWBL	7	SPN,COL,ROW SCL,SRW,HGT LEN,@V1	Spritefenster wird blockweise in das Bildschirmfenster bewegt.
PWIF	6	SPN,COL,ROW SCL,SRW,HGT LEN,@V1	Spritefenster wird 'vor' Bildschirmfenster gesetzt (PUT).
PWND	7	SPN,COL,ROW SCL,SRW,HGT LEN,@V1	Spritefenster wird mittels AND in das Bildschirmfenster gesetzt.
PWOR	7	SPN,COL,ROW SCL,SRW,HGT LEN,@V1	Spritefenster wird mittels OR in das Bildschirmfenster gesetzt

		SCL,SRW,HGT, LEN,@V1	gesetzt.
RMSK	2	SPN	Ein bereits maskiertes Sprite wird neu MASKiert.
RNUM	3	SP1,SP2	Sprite SP1 wird auf Sprite SP2 umnumeriert.
ROW	1	e1	Der Grafikvariable ROW wird der Wert von e1 zugewiesen.
ROWQ	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert in der Grafikvariable ROW zugeordnet.
RSPR	1	e1	Spriteplatz wird um den Wert e1 versetzt.
SCL	1	e1	Der Grafikvariable SCL wird der Wert von e1 zugewiesen.
SCLQ	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert in der Grafivariable SCL zugewiesen.
SCLS	4		Der Bildschirm wird gelöscht und das laufende Textfenster wird das Textfenster von Laser BASIC.
SCNP	1	SPN,COL,ROW HGT,LEN,@V1	Spritefenster wird auf Daten abgetastet.
SCNS	1	SPN	Das ganze Sprite wird auf Daten abgetastet.
SCNV	1	COL,ROW,HGT LEN	Bildschirmfenster wird auf Daten abgetastet.
SET	1	e1	Der Wert von e1 wird dem laufenden Wert von SET zugeordnet.
SETP	5	SPN,COL,ROW HGT,LEN,IK1 IK2	Das Spritefenster wird umgefärbt.
SETQ	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der laufende SET-Wert zugewiesen.
SETS	5	SPN	Das ganze Sprite wird umgefärbt.
SETV	5	COL,ROW,HGT LEN	Das Bildschirmfenster wird umgefärbt.
SP1	1	e1	Der Grafikvariable SP1 wird der Wert von e1zugeordnet.
SP1Q	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert in der Grafikvariable SP1 zugewiesen.
SP2	1	e1	Der Grafikvariable SP2 wird der Wert von e1 zugewiesen.
SP2Q	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert in der Grafikvariable SP2 zugeordnet.
SP3	1	e1	Der Grafikvariable SP3 wird der Wert von e1 zugewiesen.
SP3Q	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert in der Grafikvariable SP3 zugeordnet.

SP4			Der Grafikvariable SP4 wird der Wert von e1 zugewiesen.
SP4Q	1	@V1	Der ganzzahligen BASIC-Varible V1 wird der Wert in der Grafikvariable SP4 zugeordnet.
SPL1	8	SPN,COL,ROW HGT,LEN,e1 e2	Spritefenster wird um 1 Bildpunkt gerollt, kein Umlauf.
SPL4	9	SPN,COL,ROW HGT,LEN,e1 e2	Spritefenster wird um 1 Byte nach links gerollt, kein Umlauf.
SPL8	9	SPN,COL,ROW HGT,LEN,e1 e2	Spritefenster wird um 2 Bytes nach links gerollt, kein Umlauf
SPN	1	e1	Der Grafikvariable SPN wird der Wert von e1 zugewiesen
SPNQ	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert in de Grafikvariable SPN zugewiesen.
SPNV	2	COL,ROW,HGT LEN,SCL,SRW	Dreht Bildschirmfenster um 90 Grad im Uhrzeigersinn.
SPR1	8	SPN,COL,ROW HGT,LEN,e1 e2	Spritefenster wird um 1 Bildpunkt nach rechts gerollt, kei Umlauf
SPR4	9	SPN,COL,ROW HGT,LEN,e1,e2	Spritefenster wird um 1 Byte nach rechts gerollt, kein Umlau
SPR8	9	SPN,COL,ROW HGT,LEN,e1 e2	Spritefenster wird um 2 Bytes nach rechts gerollt, kein Umlau
SPVN	9	SPN,COL,ROW HGT,LEN,NPX e1,e2	Spritefenster wird um NPX Bildpunkte in vertikaler Richtun gerollt, kein Umlauf.
SRW	1	e1	Der Grafikvariable SRW wird der Wert von e1 zugewieser
SRWQ	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Grafikvariabl SRW zugewiesen.
SSL1	8	SPN,e1,e2	Sprite wird um 1 Bildpunkt nach links gerollt, kein Umlau
SSL4	9	SPN,e1,e2	Sprite wird um 1 Byte nach links gerollt, kein Umlauf.
SSL8	9	SPN,e1,e2	Ssprite wird um 2 Bytes nach links gerollt, kein Umlauf.
SSPR	1	e1,e2	Spriteplatz wird eingestellt.
SSR1	8	SPN,e1,e2	Sprite wird um 1 Bildpunkt nach rechts gerollt, kein Umlau
SSR4	9	SPN,e1,e2	Sprite wird um 1 Byte nach rechts gerollt, kein Umlauf.
SSR8	9	SPN,e1,e2	Sprite wird um 2 Bytes nach rechts gerollt, kein Umlauf.

		e2	Sprite wird um NPX Bildpunkte in vertikaler Richtung gerollt, kein Umlauf.
STCP	4	SPN,COL,ROW HGT,LEN,IK1	Farbe wird für das ganze Spritefenster eingestellt.
STCS	4	SPN	Farbe wird für das ganze Sprite eingestellt.
STCV	4	COL,ROW,HGT LEN	Farbe wird für das ganze Bildschirmfenster eingestellt.
SUMP	1	SPN,COL,ROW HGT,LEN,e1 e2,...,@V1	Das Spritefenster wird summiert und zugewiesen bzw. verglichen.
SUMS	1	SPN,e1,e2 ,...,@V1	Das ganze Sprite wird summiert und zugewiesen bzw. verglichen.
SUMV	1	COL,ROW,HGT LEN,e1,e2, ...,@V1	V1Das Bildschirmfenster wird summiert und zugewiesen bzw. verglichen.
SVL1	8	COL,ROW,HGT LEN,e1,e2	Das Bildschirmfenster wird um 1 Bildpunkt nach links gerollt, kein Umlauf.
SVL4	9	COL,ROW,HGT LEN,e1,e2	Das Bildschirmfenster wird um 1 Byte nach links gerollt, kein Umlauf.
SVL8	9	COL,ROW,HGT LEN,e1,e2	Das Bildschirmfenster wird um 2 Bytes nach links gerollt, kein Umlauf.
SVR1	8	COL,ROW,HGT LEN,e1,e2	Das Bildschirmfenster wird um 1 Bildpunkt nach rechts gerollt, kein Umlauf.
SVR4	9	COL,ROW,HGT LEN,e1,e2	Das Bildschirmfenster wird um 1 Byte nach rechts gerollt, kein Umlauf.
SVR8	9	COL,ROW,HGT LEN,e1,e2	Das Bildschirmfenster wird um 2 Bytes nach rechts gerollt, kein Umlauf.
SVVN	9	COL,ROW,HGT LEN,NPX,e1 e2	Das Bildschirmfenster wird um NPX Bildpunkte in vertikaler Richtung gerollt, kein Umlauf.
SWPS	4	SP1,SP2,SP3 SP4	Rahmenablauf wird umgekehrt.
TMOV	11	KEY,@V1	Eine Anweisung in einem Verfolger wird ausgeführt.
TPUT	1	KEY,SP1,COL ROW,e1	Ein Verfolger wird eingeleitet.
WBNC	11	SP1,SP2,SP3 SP4,HGT,LEN COL,ROW,KEY @V1,e1,e2	Sprungbewegung mit Blocküberschreibung.
WMOV	11	SP1,SP2,SP3 SP4,HGT,LEN COL,ROW @V1,e1,e2	Lineare Bewegung mit Blocküberschreibung.
WMJV	11	SP1,SP2,SP3 SP4,HGT,LEN COL,ROW,KEY @V1,e1,e2	Bewegung unter Tastatur-/Joysticksteuerung mit Blocküberschreibung.
WPL1	8	SPN,COL,ROW HGT,LEN,e1 e2	Spritefenster wird um 1 Bildpunkt nach links gerollt, kein Umlauf.
WPL4	9	SPN,COL,ROW HGT,LEN,e1 e2	Spritefenster wird um 1 Byte nach links gerollt, kein Umlauf.
WPL8	9	SPN,COL,ROW HGT,LEN,e1 e2	Spritefenster wird um 2 Bytes nach links gerollt, kein Umlauf.
WPR1	8	SPN,COL,ROW HGT,LEN,e1 e2	Spritefenster wird um 1 Bildpunkt nach rechts gerollt, kein Umlauf.
WPR4	9	SPN,COL,ROW HGT,LEN,e1 e2	Spritefenster wird um 1 Byte nach rechts gerollt, kein Umlauf.
WPR8	9	SPN,COL,ROW HGT,LEN,e1,e2	Spritefenster wird um 2 Bytes nach rechts gerollt, kein Umlauf.
WPVN	9	SPN,COL,ROW HGT,LEN,NPX e1,e2	Spritefenster wird um NPX Bildpunkte in vertikaler Richtung gerollt, kein Umlauf.
WSL1	8	SPN,e1,e2	Das ganze Sprite wird um 1 Bildpunkt nach links gerollt, mit Umlauf.
WSL4	9	SPN,e1,e2	Das ganze Sprite wird um 1 Byte nach links gerollt, mit Umlauf.
WSL8	9	SPN,e1,e2	Das ganze Sprite wird um 2 Bytes nach links gerollt, mit Umlauf.
WSR1	8	SPN,e1,e2	Das ganze Sprite wird um 1 Bildpunkt nach rechts gerollt, mit Umlauf.
WSR4	9	SPN,e1,e2	Das ganze Sprite wird um 1 Byte nach rechts gerollt, mit Umlauf.
WSR8	9	SPN,e1,e2	Das ganze Sprite wird um 2 Bytes nach rechts gerollt, mit Umlauf.
WSVN	9	SPN,e1,e2 NPX	Das ganze Sprite wird um NPX Bildpunkte in vertikaler Richtung gerollt, mit Umlauf.
WVL1	8	COL,ROW,HGT LEN,e1,e2	Bildschirmfenster wird um 1 Bildpunkt nach links gerollt mit Umlauf.
WVL4	9	COL,ROW,HGT LEN,e1,e2	Bildschirmfenster wird um 1 Byte nach links gerollt, mit Umlauf.
WVL8	9	COL,ROW,HGT	Bildschirmfenster wird um 2 Byte...

WV...	6	COL,ROW,HGT LEN,e1,e2	Bildschirmfenster wird um * Bildpunkt nach rechts mit Umlauf.
WVR4	9	COL,ROW,HGT LEN,e1,e2	Bildschirmfenster wird um 1 Byte nach rechts gerollt, mit Umlauf.
WVR8	9	COL,ROW,HGT LEN,e1,e2	Bildschirmfenster wird um 2 Bytes nach rechts gerollt, mit Umlauf.
WVVN	9	COL,ROW,HGT LEN,NPX,e1 e2	Bildschirmfenster wird um NPX Bildpunkte in vertikaler Richgung gerollt, mit Umlauf.
XBNC	11	SP1,SP2,SP3 SP4,HGT,LEN COL,ROW,Key @V1,e1,e2	Sprungbewegung mit Exklusiv-ODER.
XCL	1	e1	Der Grafikvariable XCL wird der Wert in e1 zugeordnet.
XCLQ	1	@V1	Der ganzzahligen BASIC-Variable V1 wird der Wert in der Grafikvariable XCL zugeordnet.
XMOV	11	SP1,SP2,SP3 SP4,HGT,LEN COL,ROW,@V1 e1,e2	Lineare Bewegung mit Exklusiv-ODER.
XMVJ	11	SP1,SP2,SP3 SP4,HGT,LEN COL,ROW,KEY @V1,e1,e2	Bewegung unter Tastatur-/Joysticksteuerung mit Exklusiv-Oder.

von Cyclone Software

EINFÜHRUNG

Der Spritegenerator wurde als Ergänzung zur Laser-Reihe entwickelt. Die Laser-Sprachen verfügen über Befehle, mit denen Sprites und Bildschirmdaten manipuliert werden können. Es können mit ihnen aber keine Sprites direkt entworfen werden. Dieses bedeutet, daß sich die Entwicklung von Spielen aus zwei Teilen zusammensetzt. Der erste Teil der Spielentwicklung beschäftigt sich mit dem Entwurf und dem Editieren Ihres Sprites mit Hilfe des Spritegeneratorprogramms, und der zweite Teil umfaßt die eigentliche Spielentwicklung mittels der Laser-Sprachen. In der Praxis erfolgen diese beiden Arbeiten gewöhnlich gleichzeitig. Das Spritegeneratorprogramm wurde so entworfen, daß es in allen drei Bildschirmmodi des Amstrad funktioniert. Um den Spritegenerator so bedienungsfreundlich wie möglich zu gestalten, werden sämtliche Operationen, unabhängig von eingestellten Bildschirmmodus, in der gleichen Art und Weise ausgefuhrt. Die Rotationsoperation sollte jedoch nur im Modus 1 verwendet werden.

DAS HAUPTMENÜ

Das Hauptmenü zeigt Ihnen alle verwendbaren Operationen an. Sie können eine dieser Optionen durch Drücken der entsprechenden Nummer auswählen. Die Optionen 1, 2 und 3 ermöglichen es Ihnen, Sprites zu entwerfen. Mit der Option 4 können Sie das Sprite, das Sie entworfen haben, auf Band bzw. auf Diskette speichern. Mit Option 5 können Sie Sprites, die bereits früher entworfen wurden, wieder in den Spritegenerator zuruckladen. Mit der Option 6 können Sprites, die auf Band bzw. Diskette gespeichert sind, mit Spritegenerator befindlichen Sprites gemischt werden. Wenn Sie eine Diskette verwenden, dann können Sie mit Option 7 sehen, was Sie auf der Diskette haben und, falls Sie es wünschen, eine Datei entfernen. Mit Option 8 können Sie zwischen mehreren Sprites Trickeffekte nach einem bestimmten Muster ausfuhren. Mit Option 9 können Sie zwische Band oder Diskette wählen.

GLOSSAR DER AUSDRÜCKE

DAS ZEICHENQUADRAT

Das Zeichenquadrat bezieht sich auf den 8 x 8 großen Raster, der sich in der linken oberen Bildschirmecke befindete. Dies ist jener Bereich, in dem Sprites Zeichen für Zeichen entworfen und editiert werden.

DER SPRITE-ANZEIGERAUM

Hier handelt es sich um die Fläche unten am Bildschirm, die zur Erstellung, Transformation und zum allgemeinen Arbeiten an Sprites vorgesehen ist.

DER ZEICHENQUADRATCURSOR

Dies ist der zerstörungsfreie, blinkende Cursor, der verwendet wird, um das im Zeichenquadrat gehaltenen Zeichen zu entwerfen und zu editieren

DAS SPRITEANZEIGEFENSTER

Jener Bildschirmbereich, an dem gerade gearbeitet wird, wird als Spriteanzeigefenster bezeichnet. Die Position des Fensters wird durch SX und SY bestimmt, was sich auf die linke obere Ecke des Fensters bezieht. Seine Abmessungen werden durch XS und YS bestimmt. Die obere linke Ecke des Spriteanzeigebereichs hat die Koordinaten SX,0, SY 0. Um das Fenster anzuzeigen, an dem Sie gerade arbeiten, müssen Sie ":" (Doppelpunkt) drucken, und es wird aufblinken.

Dieser Ausdruck bezeichnet jenen Spritesatz, an dem Sie gerade arbeiten. Die Bibliothek enthält 1 bis 255 Sprites. Wenn das Programm das erste Mal abläuft, w[illegible]en Sie aufgefordert, die maximale Spritezahl, die Sie verwenden möchten, einzugeben.

SPRITES

Ein Sprite ist ein programmierbares Grafikzeichen. Das Spritegeneratorprogramm kann bis zu 255 Sprites mit benutzbaren Abmessungen entwickeln. Der Generator verfügt über ca. 6k freien Speicherplatz. Laser BASIC kann aber eine Anzahl von Spritedateien mischen, so daß dies kein Problem darstellt.

INKS UND PAPER (Farben und Papier)

Im Bildschirmmodus 0 stehen 16 INKS (Farben) zur Verfügung, im Bildschirmmodus 1 sind es 4 Farben und im Bildschirmmodus 2 sind es nur 2 Farben. Die Farben dieser INKs werden rechts vom Zeichenquadrat angezeigt. Die erste INK-Farbe (INK 0) wird als Papier bezeichnet. Dies ist jene Farbe, wo keine Bildpunkte eingestellt sind. Wenn die erste INK-Farbe geändert wird, dann wird sich die Farbe des ganzen Bildschirms ändern.

ENTWICKELN VON SPRITES

DIE LEERTASTE

Das Spritegeneratorprogramm arbeitet in zwei Betriebsarten, dem Zeichenmodus und dem Spritemodus. Im Zeichenmodus können Sie Zeichen auf dem Zeichenquadrat entwickeln und dann Zeichen in den Spriteanzeigenbereich setzen oder daraus hervorholen. Im Spritemodus können Sie mit Hilfe des Spritegeneratorprogramms Ihre Sprites im Spriteanzeigebereich entwickeln und transformieren. Mit der LEERTASTE können Sie zwischen Zeichen- und Spritemodus wechseln.

ZEICHENMODUS

Cursorbewegung

Der zerstörungsfreie Cursor kann unter Verwendung der folgenden Tasten über das Zeichenquadrat bewegt werden:

A	für Bewegung nach links.
D	für Bewegung nach rechts.
W	für Bewegung nach oben.
X	für Bewegung nach unten.

Wenn mit einem der obigen Tasten die SHIFT-Taste gedrückt wird, dann wird sich der Cursor im Anzeigebereich bewegen und CX und CY werden angepaßt. 'A', 'D', 'W' und 'X' werden durch den Joystick ersetzt, falls vorhanden.

Die 'S' Taste

Hiermit können Sie ein Zeichen aus dem Zeichenquadrat auf die laufende Cursorposition im Spriteanzeigebereich setzen. Die Cursorposition wird durch CX und CY bestimmt. Die Position von CX und XY im Spritebereich wird durch den blinkenden Cursor angezeigt.

Die 'L'-Taste

Hiermit können Sie aus dem Spriteanzeigebereich ein Sprite hervorholen und es in das Zeichenquadrat erweitern.

Die 'I'-Taste

Wenn Sie die 'I'-Taste drücken, wird INK durch alle erhältlichen Farben laufen, bis 'I' wieder losgelassen wird.

Durch Drücken [illegible]r 'C'-Taste können Sie das Zeichenquadrat löschen. Wenn Sie die 'SHIFT'-Taste im gleichen Au[illegible]blick drücken, dann bleibt das Zeichenquadrat zwar unverändert, der Spriteanzeigebereich wird aber gelöscht werden.

Die LEERTASTE

Durch Drücken der LEERTASTE wird der Spritegenerator auf den Spritemodus eingestellt.

Die 'P'-Taste.

Durch Drücken der 'P'-Taste wird der Pfeil, der die verwendete INK-Farbe anzeigt, nach unten bewegt. Wenn die 'SHIFT-Taste im gleichen Augenblick gedrückt wird, dann bewegt sich der Pfeil nach oben.

Die 'J'-Taste

Wenn Sprites in einem Programm verwendet werden, dann wird die INK-Farbe gewöhnlich auf jene Farbe eingestellt sein, mit der die Sprites entworfen wurden. Wenn Sie die 'J'-Taste drücken, wird die Farbnummer, auf die alle INKs eingestellt sind, angezeigt.

Die 'B'-Taste

Durch Drücken der 'B'-Taste läßt sich die Randfarbe ändern.

Die 'CLR'-Taste

Durch Drücken der 'CLR'-Taste werden beide Cursors auf die Ausgangsstellung zurückgestellt (in die linke obere Ecke).

Die 'ENTER'-Taste

Die 'ENTER'-Taste stellt den Bildpunkt in der laufenden Cursorposition im Zeichenquadratraster auf den laufenden Schreibstift ein. Falls ein Joystick verwendet wird, kann diese Funktion auch mit dem Schußknopf ausgeführt werden.

Die 'DEL'-Taste

Die 'Del'-Taste stellt den Bildpunkt in der laufenden Cursorposition im Zeichenquadrat auf die Papierfarbe ein (INK 0).

Die 'E'-Taste

Durch Drücken der 'E'-Taste können Sie Daten direkt an der laufenden Position im Spriteanzeigebereich eingeben. Wenn Sie die 'E'-Taste drücken, werden Sie aufgefordert, anzugeben, ob Sie die Daten in Dezimal- bzw. in Binärform eingeben möchten. Sie sollten danach entweder 'D' oder 'B' drücken. Danach werden Sie aufgefordert Daten von 8 Bytes einzugeben. Diese werden direkt auf den Spriteanzeigebereich gebracht und verändern das Zeichenquadrat nicht.

Die 'R'-Taste

Wenn Sie die 'R'-Taste gemeinsam mit der 'SHIFT'-Taste drücken, werden Sie zum Hauptmenü zurückkehren.

Bewegung des Spriteanzeigefensters

Das Spriteanzeigefenster kann unter Verwendung der folgenden Tasten bewegt werden:

A	für Bewegung nach links;
D	für Bewegung nach rechts;
W	für Bewegung nach oben;
X	für Bewegung nach unten;

Größe des Spriteanzeigefensters:

Die Größe des Spriteanzeigefensters kann verändert werden, indem Sie die folgenden Tasten in Verbindung mit der 'SHIFT-Taste drücken:

A	zum Verkürzen der Fensterlänge;
D	zum Verlängern der Fensterlänge;
W	zum Verkürzen der Fensterhöhe;
X	zum Verlängern der Fensterhöhe.

Achten Sie darauf, daß der Gebrauch der Tasten 'A', 'D', 'W' und 'X' zur Bewegung und änderung der Abmessungen von Spriteanzeigefenstern vom Joystick ersetzt wird, falls angeschlossen.

Die '@' -Taste

Erhöht die X-Zunahme (in Bytes), um die ein Spriteanzeigefenster bewegt wird. Die Zunahme wird bis zu einem Wert von 8 steigen und dann von 1 neu beginnen.

Die ';' -Taste

Erhöht die Y-Zunahme (in Bildpunkten), um die ein Spriteanzeigefenster bewegt wird. Die Zunahme wird wiederum bis zu einem Wert von 8 steigen un dann von 1 neu beginnen.

Die 'L'-Taste

Durch Drücken der 'L'-Taste kann ein bereits früher erzeugtes Sprite bei der laufenden Cursorposition auf den Spriteanzeigebereich gesetzt werden. Das ganze Sprite wird unabhängig von der laufenden Größe des Spritefensters auf den Spriteanzeigebereich gesetzt werden.

Die 'S'-Taste

Diese Taste ermöglicht es Ihnen, ein Sprite in den Speicher zu holen (GET). Durch Drücken der 'S'-Taste wird es ermöglicht, ein Sprite mit den laufenden Abmessungen des Spriteanzeigefensters und dem laufenden SPN-Wert zu erzeugen (vorausgesetzt es wurde noch nicht zugeteilt). Die Daten im Fenster werden automatisch in das Sprite geholt (GOT). Falls die Spritenummer bereits zugeteilt wurde, wird ein tiefer Piepton erzeugt und es wird keine weitere Aktion stattfinden. Wenn die Operation erfolgreich war, dann wird ein hoher Piepton zu hören sein.

Die 'P'-Taste

Das Spritegeneratorprogramm kann Sprites in einer von sechs verschiedenen Methoden in den Spriteanzeigebereich bringen. Dies wird als Druckmodus bezeichnet. Die sechs Druckmodi werden durch PMD angegeben und sind wie folgt:

BL	Daten werden blockweise auf den Spriteanzeigebereich gebracht (PUT)
OR	Daten werden mittels OR auf den Spriteanzeigebereich gebracht
XOR	Daten werden mittels XOR auf den Spriteanzeigebereich gebracht
AND	Daten werden mittels AND auf den Spriteanzeigebereich gebracht
IF	Daten werden vor laufenden Daten am Spriteanzeigebereich gesetzt
BH	Daten werden hinter laufenden Daten am Spriteanzeigebereich gesetzt

Durch Drücken der 'P'-Taste werden diese Druckmodi durchlaufen.

Die 'N'-Taste

Durch Drücken der 'N'-Taste wird die laufende Spritenummer (dargestellt durch SPN) erhöht. Fa die 'SHIFT'-Taste im gleichen Augenblick gedrückt wird, wird die laufende Spritenummer verkleine

Die 'F'-Taste

Schaltet den speziellen FILL-Cursor ein, der im Spriteanzeigebereich durch Verwendung der Tast 'A', 'D', 'W', und 'X' (oder eines Joysticks) herumbewegt werden kann. Durch Drücken von 'ENTE oder 'FIRE' kann die entsprechende Fläche von der Cursorposition aus mit der laufenden INK-Far gefüllt (FILL) werden.

Die 'E'-Taste

Mit der 'E'-Taste kann ein bereits erzeugtes Sprite wieder gelöscht werden. Bevor Sie das Spr eigentlich löschen, werden Sie gefrage "ARE YOU SURE? (Y/N)" (Sind Sie sich sicher?), damit S ein Sprite nicht zufällig löschen.

Die 'O'-Taste

Mit der 'O'-Taste können Sie 'Wrap' (Umlauf) für die Scrolls (durch WRP dargestellt) ein- u ausschalten. Wenn 'Wrap' eingeschaltet ist, dann werden sämtliche Daten, an denen Scroll-Befehl ausgeführt wird, auf einer Seite des Spriteanzeigefensters verschwinden und auf anderen Seite wieder auftauchen. Wenn 'Wrap' ausgeschaltet ist, geschieht dies nicht.

Die Pfeiltasten

Die Pfeiltasten haben zwei Funktionen. Wenn sie alleine benutzt werden, dann wird Spriteanzeigefenster in die entsprechende Richtung gerollt. Wenn jedoch die 'SHIFT'-Ta gemeinsam mit den Pfeiltasten gedrückt wird:

←	Kippt den Inhalt des Spriteanzeigefensters um die Y-Achse.
↑	Kippt den Inhalte des Spriteanzeigefensters um die X-Achse.
→	Kippt den Inhalt des Spriteanzeigefensters zur rechten Fensterhälfte (erzeugt ein horizontaler Richtung symmetrisches Fenster)
↓	Kippt den Inhalt des Spriteanzeigefensters zur linken Fensterhälfte (erzeugt ein vertikaler Richtung symmetrisches Fenster).

Mit Hilfe der Tasten '→' und '↓' ist es möglich ein symmetrisches Design zu erzeugen und dabei ein Viertel bzw. die Hälfte des kompletten Designs zu entwerfen.

Die 'B'-Taste

Wenn die 'B'-Taste gedrückt wird, dann wird der Inhalt des Spriteanzeigefensters in horizonta Richtung vergrößert.

Wenn die 'SHIFT'-Taste in Verbindung mit der 'B'-Taste verwendet wird, dann wird der Inhalt Spriteanzeigefenster in vertikaler Richtung vergrößert.

Die 'T'-Taste

Mit Hilfe der 'T'-Taste können Sie den Inhalt des Spriteanzeigefensters um 90 Grad im Uhrzeigers in einen anderen Bereich des Bildschirms verdrehen. Wenn die 'T'-Taste gedrückt wird, dann w angenommen, daß die Daten, die im Spriteanzeigefenster enthalten sind, jene Daten sind, gedreht werden sollen. Das Spriteanzeigefenster soll in jene Position gebracht werden, in die

Daten gedreht werden sollen. Wenn sich der Cursor in der entsprechenden Position befindet, dann soll die 'ENTER'-Taste gedrückt werden. Der Bereich, in den die Datengedreht werden, soll den Bereich, von dem die Daten stammen, nicht überlappen. Die Höhe des zu drehenden Bereichs wird auf die nächsten 8 Bildpunkte abgerundet. Siehe SPNV.

Die 'Z'-Taste

Mit Hilfe der 'Z'-Taste kann der Inhalt des Spriteanzeigefensters invertiert werden.

Die 'H'-, 'G'- und 'V'-Tasten

Manchmal kann es notwendig sein, ein Sprite zu entwerfen, das größer ist als der Spriteanzeigebereich. Dies ist möglich, in dem ein großes Sprite im Speicher erzeugt wird und anschließend Stück für Stuck entworfen wird. Durch Drücken der 'H'-Taste läßt sich im Speicher ein großes Sprite erzeugen. Sie werden die Aufforderung erhalten, die Breite des Sprites in Bytes und die Höhe in Bildpunkten einzugeben. Durch Drücken der 'G'-Taste können Sie den Inhalt des Spriteanzeigefensters in das Reihenpositionen innerhalb des Sprites anzugeben. Durch Drücken der 'V'-Taste wird ein Teil des großen Sprites in den Spriteanzeigebereich gebracht. Die entsprechende Position hierfür und die Größe des dafür gewählten Teils wird im Spriteanzeigefenster angegeben werden. Sie werden die Aufforderung erhalten, die Spalten- und Reihenposition innerhalb des Sprites, von dem die Daten genommen werden, anzugeben. Die Spritenummer wird wiederum durch SPN angegeben.

Die '['-Taste

Die '['-Taste kann bei einer horizontalen Scroll-Bewegung der Betrag, um den ein Spriteanzeigefenster gerollt wird, verändert werden.

Die ']'-Taste

Durch Drücken der ']'-Taste kann bei einer vertikalen Scroll-Bewegung der Betrag, um den ein Spriteanzeigefenster gerollt wird, verändert werden.

Die 'K'-Taste

Diese Option bietet Ihnen die Möglichkeit die Farbe eines bestimmten Bildpunktes im Spriteanzeigefenster auf eine andere Farbe zu ändern (siehe SETV). Dies beeinträchtigt die Farbe der INKs, ändert jedoch die Bildpunkte im Spriteanzeigebereich. Wenn die 'K'-Taste gedrückt wird, werden Sie aufgefordert, den ersten INK-Wert einzugeben. Dies ist die INK-Nummer der Bildpunkte, die zu ändern sind. Danach werden Sie aufgefordert, den zweiten INK-Wert einzugeben. Dies ist jene INK-Nummer, auf die die Bildpunkte geändert werden sollen.

Die 'M'-Taste

Die 'M'-Taste bewirkt das Maskieren und Entmaskieren eines Sprites. Wenn die 'M'-Taste gedrückt wird, dann wird das Sprite, dessen Nummer in SPN angegeben ist, maskiert. Wenn die 'SHIFT'-Taste im gleichen Augenblick gedrückt wird wie die 'M'-Taste, dann wird das Sprite, dessen Nummer in SPN angegeben ist, entmaskiert.

Die ':'-Taste

Wenn die ':'-Taste gedrückt wird, dann wird das laufende Spriteanzeigefenster solange aufblinken, bis die Taste losgelassen wird. Diese Option wird geboten, um die Position und die Große des Fensters zu erhalten.

Die 'I'-Taste

Wenn die 'I'-Taste gedruckt wird, dann wird der neue Informationssatz angezeigt.

Die 'R'-Taste

Durch gleichzeitiges Drücken der 'SHIFT'- und 'R'-Taste können Sie zum Hauptmenu zurückkehren

Achten Sie darauf, daß alle Daten, die im Zeichenquadrat und im Spriteanzeigefenster enthalten sind, verloren gehen, wenn Sie zum Hauptmenü zurückkehren. Alle Systemwerte werden ebenfalls auf die Standardwerte zurückgesetzt.

TRICKBEWEGUNG VON SPRITES

Sie können zwischen mehreren Sprites Trickbewegungen ausführen, wenn Sie vom Hauptmenü Option 8 wählen. Die Ablauffolge, in der der Spritegenerator eine Trickbewegung ausführen wird, wird an der Bildschirmoberseite eingegeben. Die eigentliche Trickbewegung wird an der Unterseite des Spriteanzeigebereichs erfolgen.

Eingabe eines Trickbewegungsmusters

Um ein Sprite in den Trickbewegungsablauf einzugliedern, müssen Sie den Cursor zu jener Position im Ablauf bewegen, an der das Sprite eingegeben werden soll. Anschließend müssen Sie die Nummer des Sprites eingeben, das Sie an dieser Position eingeben möchten und danach ist die 'ENTER'-Taste zu drücken. Am Ende jedes Trickbewegungsmusters muß 'R' eingegeben werden, und das Bewegungsmuster wird an dieser Stelle wieder von vorne beginnen.

Cursorbewegung

Der Cursor, der die laufende Position im Trickbewegungsablauf anzeigt, kann durch folgende Tasten bewegt werden:

A	für Bewegung nach links;
B	für Bewegung nach rechts;
W	für Bewegung nach oben;
X	für Bewegung nach unten.

Falls mit einer der obigen Tasten die 'SHIFT'-Taste gedrückt wird, dann wird sich der Cursor in der Spriteanzeige bewegen. Dieser Cursor gibt an, in welcher Position der Bewegungsablauf angezeigt wird.

Die LEERTASTE

Durch Drücken der LEERTASTE kann der Trickbewegungsablauf begonnen und angehalten werden.

Die 'R'-Taste

Gleichzeitiges Drücken der 'SHIFT'- und der 'R'-Taste wird Sie wieder zum Hauptmenü zurückbringen.

ZWISCHENSPEICHERN VON SPRITES

Sie können Ihre Sprites auf Bank bzw. auf Diskette zwischenspeichern, indem Sie Option 4 aus dem Hauptmenü wählen. Nachdem Sie diese Option gewählt haben, werden Sie aufgefordert, den Dateinamen einzugeben, unter dem Sie die Sprites speichern möchten. Der Dateiname muß in Großbuchstaben eingegeben werden und soll nicht mehr als 5 Zeichen lang sein. Es werden drei Datensätzen gespeichert. Der erste enthält Systemvariablen, die von Laser-Routinen benutzt werden, der zweite enthält die Spritetabelle, welche Informationen über jedes Sprite beinhaltet, und der dritte Satz sind die eigentlichen Spritedaten.

LADEN VON SPRITES

Option 5 ermöglicht es, bereits gespeicherte Sprites von Band bzw. von Diskette zu laden. Nachdem Sie diese Option gewählt haben, werden Sie die Aufforderung erhalten, den Dateinamen der zu ladenden Sprites einzugeben. Dem Dateinamen, den Sie eingeben, wird automatisch '.SPR' beigefugt. Die neue maximale Spritenummer (von der geladenen Datei) wird angezeigt, und Sie erhalten die Moglichkeit dies zu andern. Falls Sie der Aufforderung mit einem "Y" antworten, sollten Sie ein neues SMAX eingeben. Dieser Wert muß zwischen 1 und 255 liegen. Wenn die Datei nicht gefunden wird, wird die Aufforderung trotzdem erscheinen. Sie werden diesmal wahrscheinlich ein 'N' eingeben, es sei denn, Sie wollen SMAX aus irgendeinem

Spritedateien können auch von Band bzw. von Diskette gemischt werden. Dies kann allerdings nur dann durchgeführt werden, wenn keines der von Band bzw. Diskette gemischten Sprites dieselbe Nummer hat wie irgendein Sprite, das gerade im Spritegenerator gehalten wird. Weiters darf die maximale Spritenummer des Sprites, das gemischt werden soll, nicht die laufende Spritenummer überschreiten, die Sie am Anfang der Sitzung eingegeben haben, als Sie das Programm zum ersten Mal laufen ließen. Nachdem Sie diese Option gewählt haben, werden Sie wiederum die Aufforderung erhalten, den Dateinamen der Sprites einzugeben, die Sie von Band bzw. Diskette mischen möchten.

EINE DEMONSTRATION MIT DEM SPRITE-GENERATOR

Falls Sie Laser BASIC noch nicht geladen haben, sollten Sie es jetzt laden. Das Spritegeneratorprogramm kann jetzt geladen und abgelaufen werden. Kontrollieren Sie ob die Tastatur auf Großbuchstaben eingestellt ist, bevor Sie das Programm laufen lassen – falls nicht, drücken Sie CAPS LOCK. Sie sollten auch darauf achten, daß die Lautstärke auf hoch eingestellt ist, damit Fehlerpieptone zu hören sind.

Für Bandbenutzer: Band 2 mit Seite A einlegen und RUN"SPTGEN eintippen. Für Diskettenbenutzer: Um das Spritegeneratorprogramm zu laden und laufen zu lassen, einfach RUN"SPTGEN eintippen.

Wenn das Programm zum ersten Mal ausgeführt wird, werden Sie die Aufforderung erhalten, die maximale Spritenummer einzugeben. Dies sollte ein Wert zwischen 1 und 255 sein. In diesem Fall geben Sie 120 ein. Die Nachricht "IF YOU HAVE MADE AN ERROR YOU WILL HEAR ..." (Falls Sie einen Fehler gemacht haben, hören Sie ...) wird erscheinen und' zwei tiefe Pieptöne werden erklingen. Dies soll nur demonstrieren, daß er zu hören ist, sollte ein Fehler auftreten. Die Nachricht, "IF THE OPERATION WAS SUCCESSFUL ..." (Wenn die Operation erfolgreich war...) wird erscheinen und zwei hohe Pieptöne werden zu hören sein. Dies oll nur demonstrieren, daß Sie ihn hören können, wenn die Operation erfolgreich war. Drücken Sie jetzt eine beliebige Taste, und das Hauptmenü wird erscheinen.

Sie werden einige Sprites zum Arbeiten benötigen. Wir werden die Sprites SPT1 verwenden (siehe Anhang A).

Die Sprites SPT1 sollten unter Verwendung der folgenden Parameter geladen werden:

1. Wenn Sie mit Band arbeiten, sollten Sie die Kassette mit den Sprites SPT1 in den Kassettenspieler einlegen. Die Sprites SPT1 befinden sich direkt hinter dem Spritegenerator auf Band 2 Seite A.

2. Wählen Sie vom Hauptmenü Option 5. Hiermit können Sie Ihre Sprites laden.

3. Die Aufforderung "ENTER FILENAME" (Dateiname eingeben) sollte jetzt erscheinen. Sie sollten jetzt "SPT1" eingeben und danach die Taste "ENTER" drücken. Beachten Sie, daß "SPR" dem Dateinamen automatisch hinzugefügt wird. Versuchen Sie nicht bei dieser Demonstration die maximale Spritenummber zu ändern (tippen Sie also N).

4. Sie werden jetzt zum Hauptmenü zurückkehren.

Wir werden die Probesitzung im 16-Farbenmodus beginnen (MODE 0). Wählen Sie also Option 1 vom Hauptmenü.

DIE CURSORTASTEN

Mit Hilfe der Tasten 'A', 'D', 'W' und 'X' können Sie den blinkenden Cursor im Zeichenquadrat herumbewegen. Durch Drücken der 'SHIFT'-Taste und der obigen Tasten können Sie auch den Cursor im Spriteanzeigebereich herumbewegen. Benutzen Sie die Tasten, um beide Cursors zu bewegen, damit Sie ein Gefühl für die Sache bekommen. Beachten Sie, daß sich der Cursor mit Umlauf beweg[illegible]. er wird auf der anderen Seite wieder erscheinen, wenn er auf einer Seite des Zeichenquadrats bzw. des Spriteanzeigebereichs abgerollt wird.

Um einen bestimmten Bildpunkt einzustellen, müssen Sie den Zeichenquadratcursor in die gewünschte Position bringen, den Cursor loslassen und 'ENTER' drücken. Bewegen Sie den Cursor zum nächsten Punkt, den Sie einstellen wollen, und drücken Sie wieder "ENTER". Um die Einstellung eines Bildpunktes wieder aufzuheben, brauchen Sie den Cursor nur über den eingestellten Bildpunkt stellen und "DEL" drücken. Wenn keine dieser beiden Tasten gedrückt wird, dann bewegt sich der Cursor zerstörungsfrei. Das heißt er bewegt sich am Bildschirm, ohne die Zellen, über die er sich hinwegbewegt, zu verändern. Versuchen Sie jetzt für ein paar Minuten die Cursortasten kennenzulernen. Probieren Sie zum Beispiel einen außerirdischen Eindringling zu erzeugen. Sie müssen nicht unbedingt ein Kunstwerk schaffen. Mit dieser Übung lassen sich jedoch einige Funktionen dieses Pakets demonstrieren.

Die 'S'- und 'L'-Tasten

Nachdem Sie jetzt ein Zeichen entworfen haben, wird es Zeit, zu sehen, wie es auf die richtige Größe reduziert im Spriteanzeigebereich aussehen wird. Drücken Sie 'S', um Ihren Eindringling in den Spriteanzeigebereich zu bringen. Er wird in der laufenden Cursorposition im Spriteanzeigebereich erscheinen. Drücken Sie jetzt 'C'. Dies wird das Zeichenquadrat löschen. Durch Drücken von 'L' können Sie den Eindringling wieder in das Zeichenquadrat zurücksetzen. Mit Hilfe dieser Methode können Sprites Stück für Stück erzeugt und editiert werden.

Bevor wir weitergehen, möchten wir uns ganz kurz das interne Format der Bildpunktdaten von Amstrad betrachten. Wenn Sie möchten, können Sie diesen Abschnitt überschlagen, um ihn später einmal durchzulesen.

Zeichen werden am Bildschirm entweder in 8 Bytes, 16 Bytes oder 32 Bytes gespeichert, je nach dem Bildschirmmodus, den Sie gerade verwenden. Ein Byte ist eine 8-Bit-Nummer. Die Bits sind, begonnen von rechts, von 0 bis 7 numeriert, und jedes Byte entspricht 2 hoch der Nummer der einzelnen Bits. Wenn also nur die Bits 2 und 5 eingestellt sind, dann würde der Wert des Byte 36 sein. Die entsprechenden Werte von jedem Bit in einem Byte werden nachstehend in Abb. 1 gezeigt. Bit 0 ist das extrem rechte Bit, Bit 7 ist das extrem linke Bit.

ABB.1	Bit	0	=	2	to	the	power	of	0	= 1
	Bit	1	=	2	to	the	power	of	1	= 2
	Bit	2	=	2	to	the	power	of	2	= 4
	Bit	3	=	2	to	the	power	of	3	= 8
	Bit	4	=	2	to	the	power	of	4	= 16
	Bit	5	=	2	to	the	power	of	5	= 32
	Bit	6	=	2	to	the	power	of	6	= 64
	Bit	7	=	2	to	the	power	of	7	=128

Im Bildschirmmodus 0 werden 32 Bytes benötigt, um ein Zeichen zu erzeugen. Das erste Byte erzeugt die Farbe der ersten zwei Bildpunkte in der obersten Reihe des Zeichens. Das zweite Byte erzeugt die beiden zweiten Bildpunkte in der obersten Reihe des Zeichens. Das dritte Byte erzeugt die nächsten zwei Bildpunkte, das vierte Byte die letzten zwei Bildpunkte. Das fünfte Byte wird die Farbe der ersten zwei Bildpunkte in der zweiten Reihe des Zeichens erzeugen. Jedes Byte entspricht 2 Bildpunkten, die wie folgt codiert sind:

Rechter Bildpunkt - Bits 0,4,2,6
Linker Bildpunkt - Bits 1,5,3,7

Im Bildschirmmodus 1 sind 16 Bytes erforderlich, um ein Zeichen zu speichern. Das erste Byte erzeugt die Farbe der ersten vier Bildpunkte in der obersten Reihe des Zeichens. Das zweite Byte erzeugt die Farbe der letzten vier Bildpunkte in der obersten Reihe des Zeichens. Das dritte Byte erzeugt die Farbe der ersten vier Bildpunkte in der zweiten Reihe des Zeichens usw. Jedes Byte entspricht 4 Bildpunkten, die wie folgt codiert sind:

Linker Bildpunkt - Bits 0,4
Zweiter Bildpunkt - Bits 1,5

Dritter Bildpunkt - Bits 2,6
Rechter Bildpunkt - Bits 3,7

Im Bildschirmmodus 2 sind nur 8 Bytes erforderlich, um ein Zeichen zu speichern. In diesem Bildschirmmodus gibt jedes Bit an, ob der entsprechende Bildpunkt in der Zeichenreihe ein- oder ausgeschaltet ist. Drücken Sie für unseren Eindringling erstmals die 'E'-Taste und beantworten Sie danach die Frage "Decimal or Binary" (Dezimal- oder Binärform), indem Sie die 'D'-Taste drücken. Sie werden darauf die Aufforderung erhalten, den ersten Bytewert, der in Abb. 2 zu finden ist, einzugeben. Geben Sie danach das zweite Byte, das dritte Byte usw. ein, bis alle 8 Bytes eingegeben wurden. Bewegen Sie dann den Cursor im Spriteanzeigebereich um eine Stelle nach rechts und geben Sie die 8 Bytes in Abb. 3 ein. Verwenden Sie dabei dasselbe Verfahren wie bei der Eingabe der Daten in Abb. 2. Bewegen Sie den Cursor anschließend um eine weitere Stelle nach rechts und geben Sie die 8 Bytes in Abb. 4 ein. Bewegen Sie den Cursor dann um eine weitere Stelle nach rechts und geben Sie die letzten 8 Bytes in Abb. 5 ein.

	Abb.2	Abb.3	Abb.4	Abb.5
1. Byte =	17	0	0	34
2. Byte =	34	0	0	17
3. Byte =	17	51	51	34
4. Byte =	51	17	34	51
5. Byte =	51	17	34	51
6. Byte =	51	51	51	51
7. Byte =	17	34	17	34
8. Byte =	34	0	0	17

Nun zurück zur Probesitzung ....

Versuchen wir jetzt die INK-Farben zu ändern. Drücken Sie die 'P'-Taste, um den Pfeil bei den INKs hinunter zu bewegen (SHIFT 'P' wird den Pfeil hinauf bewegen). Wenn Sie jetzt 'ENTER' drücken, dann wird ein Bildpunkt auf dem Zeichenquadrat auf jene INK-Farbe eingestellt werden, die vom Pfeil anzeigt wird – versuchen Sie es. Obwohl in diesem Modus nur 16 INK-Werte Verfügung stehen, können Sie durch Drücken der 'I'-Taste den laufenden INK-Wert durch alle erhältlichen Farben durchlaufen lassen. Sie werden bemerken, daß sämtliche Bildpunkte am Zeichenquadrat und im Spriteanzeigebereich mit dem vom Pfeil angezeigten INK-Wert die Farbe ändern werden. Wenn Sie jetzt die 'J'-Taste drücken, dann werden alle INK-Werte angezeigt. Sie sollten sich diese notieren, bevor Sie die Sprites speichern, so daß Sie die INKs in Ihrem Laser-BASIC-Programm auf die die gewünschten Werte einstellen können. Drücken Sie 'J' noch einmal, um zur ursprünglichen Bildschirm zurückzukehren.

Drücken Sie jetzt die LEERTASTE, um den Spritemodus einzugeben. Beachten Sie, daß die Cursorindikatoren (CX und CY) durch die Indikatoren des Spriteanzeigefensters (SX und SY) ersetzt wurden. Um das Fenster zu betrachten, müssen Sie 'I' gedruckt halten, und es wird blinken, um seine Große und Position anzuzeigen. Das Fenster kann durch Drücken der Tasten 'A', 'D', 'W' und 'X' bewegt werden (nach links, nach rechts, nach oben und nach unten), oder durch Verwendung eines Joysticks. Die Breite des Fensters kann erhöht bzw. verringert werden, indem Sie 'SHIFT' und 'W' oder 'SHIFT' und 'X' drücken. SHIFT kann auch wieder gemeinsam mit dem Joystick verwendet werden, um dasselbe Ergebnis zu erzielen.

ERZEUGUNG EINES SPRITES

Bewegen Sie das Fenster zur oberen linken Ecke des Spriteanzeigebereichs, so daß SX und SY 0 sind. Drücken Sie jetzt die 'I'-Taste, um einen neuen Informationssatz anzuzeigen. Drücken Sie die 'N'-Taste (oder SHIFT 'N'-Taste) solange, bis der SPN-Wert in der oberen rechten Ecke 5 ist. Drücken Sie jetzt 'L', und Sprite 5 wird erscheinen. Drücken Sie wieder 'I', und ein weiterer Informationssatz wird angezeigt werden, der Ihnen sämtliche Informationen über Sprite 5 angeben wird. Wenn Sie jetzt wieder 'I' drücken, werden Sie zur ursprünglichen Information zurückkommen.

Wir werden annehmen, daß dieses Sprite gerade entworfen wurde. Sie konnten, falls Sie es wünschen in den Zeichenmodus zurückgehen ('LEERTASTE') und die Farbe der INKs ändern, falls Sie dies wünschen, drücken Sie anschließend wieder die 'LEERTASTE', um den Spritemodus zuge-

Der folgende Ablauf muß eingehalten werden, um ein Sprite zu erzeugen. Verstellen Sie das Spritefenster so lange, bis es alle Bildpunktdaten enthält, die für das Sprite erforderlich sind. Sie werden bemerken, daß sich die Werte für LEN und HGT ändern werden; für dieses Beispiel sollte LEN ca. 7 und HGT ca. 23 sein. Drücken Sie 'I' und stellen Sie SPN auf 26 ein (ein undefiniertes Sprite).

Drücken Sie 'S' (Sie werden einen hohen Piepton vernehmen) und das Sprite wird erzeugt werden. Drücken Sie 'I' und sämtliche Informationen über das Sprite werden angezeigt. Bewegen Sie das Spritefenster zu einer leeren Bildschirmstelle. Drücken Sie 'L', und wenn alles gut funktioniert hat, wird das Sprite auf den Bildschirm gebracht.

ANMERKUNG: Falls Sie versuchen sollten, ein Sprite zu erzeugen, das bereits gespeichert ist, dann wird ein tiefer Piepton zu hören sein. Falls dies jemals der Fall ist, sollten Sie versuchen, eine andere SPN-Nummer zu wählen.

Sie können ein Sprite auch löschen, indem Sie SPN auf den entsprechenden Wert einstellen und die 'E'-Taste drücken. Die Nummer des soeben gelöschten Sprite kann jetzt wieder neu verteilt werden.

Die Sprites SPT1, die Sie in den Speicher geladen haben, bestehen aus Sprites im 16-Farbenmodus und Sprites im 4 Farbenmodus. Es werden im laufenden Modus allerdings nur Sprites im 16-Farbenmodus korrekt angezeigt werden (Hauptmenuoption 1).

Wir werden jetzt auch eines der Sprites SPT1 anzeigen. Verringern Sie vorerst unter Verwendung von 'SHIFT' und 'N' den SPN-Wert auf 6, drücken Sie anschließend 'L' und Sprite 6 wird erscheinen. Bewegen Sie das Spriteanzeigefenster mit Hilfe der Tasten 'A', 'D', 'W' und 'X', so daß es über das Sprite paßt.

SPRITEBILDSCHIRMTRANSFORMATIONEN

Wir werden jetzt ein paar Scroll- und Kippbewegungen ausführen. Drücken Sie die rechte Pfeiltaste und das Sprite wird nach rechts rollen. Beachten Sie, daß WRAP im Augenblick ausgeschaltet ist (WRP ist auf 0 eingestellt); falls WRAP in Betrieb ist, wird der Teil des Sprites, das verschwunden ist, auf der gegenüberliegenden Bildschirmseite wieder erscheinen. Drücken Sie jetzt die Taste mit dem nach unten zeigenden Pfeil, und das Sprite wird zur unteren Bildschirmseite rollen. WRAP ist immer noch ausgeschaltet. Bringen Sie das Sprite mit Hilfe der 'L'-Taste wieder in den Anzeigbereich zurück. Drücken Sie die '0'-Taste, und WRAP wird in Betrieb genommen (WRP ist auf 1 eingestellt). Beobachten Sie jetzt, was passiert, wenn Sie das Sprite rollen. Drücken Sie wieder '0', um WRP auf 0 einzustellen.

Versuchen wir jetzt eine Kipp-Bewegung auszuführen. Drücken Sie 'SHIFT' und 'UP ARROW', und der Inhalt des Spriteanzeigefensters wird auf den Kopf gestellt (vertikale Spiegelung).

Wir werden jetzt eine Fläche mit einem bestimmten INK-Wert füllen (FILL). Für dieses Beispiel müssen Sie im Spriteanzeigebereich eine geschlossene Kontur zeichnen. Nachdem Sie gemacht haben, können Sie den INK-Wert ändern und den Spritemodus eingeben. Drücken Sie danach 'F'. Sie werden jetzt einen kleinen blinkenden Bildpunkt steuern können. Bewegen Sie den blinkenden Bildpunkt mit Hilfe der Tasten 'A', 'D', 'W' und 'X' innerhalb der Kontur und drücken Sie 'ENTER' oder 'FIRE'. Die ganze Innenfläche sollte jetzt mit der laufenden INK-Farbe gefüllt (FILL) werden. Falls in Ihrer Kontur Lücken vorhanden sind, wird die INK-Farbe den ganzen Spriteanzeigebereich bedecken!!

Bewegen Sie das Spriteanzeigefenster zu einem leeren Abschnitt im Spriteanzeigebereich, geben Sie den Zeichenmodus ein und bewegen Sie den Pfeil zu einem bestimmten INK-Wert. Gehen Sie jetzt wieder in den Spritemodus zurück ('LEERTASTE') und drücken Sie dann 'SHIFT' und 'F'. Das ganze Spritefenster wird jetzt mit der von Ihnen gewählten INK-Farbe gefüllt.

Drücken Sie die 'L'-Taste, um Sprite 6 auf den Bildschirm zu bringen. Setzen Sie das Fenster über das Sprite.

Versuchen wir jetzt alle Bildpunkte im Spriteanzeigefenster mit einem bestimmten INK-Wert auf einen anderen INK-Wert einzustellen. Drücken Sie 'K' und Sie werden die Aufforderung erhalten, d

Nummer ... wollen, einzugeben. Tippen Sie also Nummer 3 ein und drücken Sie 'ENTER'. Sie werden jetzt die Aufforderung erhalten, jene INK-Nummer einzugeben, auf die Sie den Bildpunkt ändern wollen. Tippen Sie also 14 und drücken Sie 'ENTER'. Dies wird jeden auf INK 3 eingestellten Bildpunkt auf INK 14 einstellen.

Kehren wir an dieser Stelle zum Hauptmenü zurück. Stellen Sie durch Drücken der LEERTASTE den Zeichenmodus ein. Bevor Sie zum Hauptmenü zurückkehren, sollten Sie sich vergewissern, ob alle Sprites im Spriteanzeigebereich gespeichert wurden, da der Spriteanzeigebereich gelöscht werden wird. Drücken Sie jetzt 'SHIFT' und 'R', um zum Hauptmenü zurückzukehren.

MODE 1 SPRITES

Wählen Sie Option 2, um den Spritegenerator im Grafikmodus 1 zu verwenden. Die Anzeige ist ähnlich wie bei dem durch Option 1 gewählten.

Drücken Sie die LEERTASTE und anschließend 'L'. SPN wird auf eingestellt sein (der Standardwert beim Wählen der Option). Sprite 1 wird also auf dem Bildschirm erscheinen (Sprite 1 ist ein MODE 1 Sprite). Sie können, falls Sie es wünschen, zum Zeichenmodus zurückkehren und die INKs auf Ihre gewünschten Farben einstellen.

ROTATION

Wir werden jetzt demonstrieren, wie eine Rotation (funktioniert nur in diesem Modus) vor sich geht. Stellen Sie zuerst das Spriteanzeigefenster so ein, daß es den Panzer abdeckt (Achten Sie darauf, daß HGT durch 8 teilbar sein muß). LEN sollte 10 sein und HGT 24.

Sie können kontrollieren, ob das Spriteanzeigefenster den Panzer abdeckt, indem Sie ':' gedrückt halten oder 'Z' invertieren sollten, müßten Sie 'Z' nochmals drücken, bevor Sie fortsetzen.

Drücken Sie 'T' und bewegen Sie das Spriteanzeigefenster zu einem freien Bereich des Bildschirms (weg vom Panzer) und drücken Sie 'ENTER'. Ein um 90 Grad rotierter Panzer wird erscheinen.

NOCHMALIGE BETRACHTUNG DES SPRITEANZEIGEFENSTERS UND DES SPRITECURSOR

Sie werden wahrscheinlich bereits bemerkt haben, daß es ziemlich lange dauert, das Spriteanzeigefenster bzw. den Spritecursor im Spriteanzeigebereich zu bewegen. Der Spritegenerator ist mit einer Möglichkeit ausgestattet, die es erlaubt, die X- und Y-Zunahmen, welche die Geschwindigkeit der Bewegung des Anzeigefensters und das Cursors bestimmen, zu ändern. Im Zeichenmodus und auch im Spritemodus kann der Wert von XS durch die @-Taste und der Wert von YS durch die ';'-Taste geändert werden. Die Werte für die beiden Modi sind jedoch voneinander unabhängig. Wenn Sie also XS und YS im Zeichenmodus ändern, bleiben die Werte, die im Spritemodus eingegeben werden, davon unberührt. Drücken Sie die @-Taste solange, bis XS 8 ist. Drücken Sie die ';'-Taste solange, bis YS ebenfalls 8 ist. Sie können nun in großen Schritten Bewegungen ausführen.

Die Scrollgeschwindigkeit des Spriteanzeigefensters kann durch die '[' und ']'-Tasten verändert. Drücken Sie im Spritemodus die 'I'-Taste Sie werden die Werte XSC und YSC zu sehen bekommen. XSC ist die Auflösung für die seitliche Scrollbewegung. Diese ist beim erstmaligen Laden des Spritegenerators auf 'PIX' eingestellt. Dies verleiht der Scrollbewegung eine Auflösung von 1 Bildpunkt

Wenn Sie jetzt '[' drücken, wird eine Auflösung von 1 Bildpunkt eingestellt sein. Drücken Sie '[noch einmal, und eine Auflösung von 2 Bildpunkten wird eingestellt. Drücken Sie '[' und probieren Sie einen seitliche Scrollbewegung.

YSC zeigt die Auflösung für die vertikale Scrollbewegung an. Durch Drücken von ']' können Sie den Wert von 1 bis auf 8 erhöhen. Versuchen Sie ein paar Scroll-Bewegungen mit underschiedlichen XSC und YSC-Werten auszuführen, und betrachten Sie sich das Ergebnis.

WEITERE BILDSCHIRMTRANSFORMATIONEN

Bringen Sie das Spriteanzeigefenster in die linke obere Ecke des Spriteanzeigbereichs, d.h. SX und SY sollten auf 0 eingestellt sein. Drücken Sie 'L', um das Sprite 1 auf den Bildschirm zu setzen Bewegen Sie das Spriteanzeigefenster jetzt so, daß es das Sprite (ein Panzer) abdeckt. Kontrollieren Sie seine Position mit Hilfe der ':' – bzw. der 'Z'-Taste.

Drücken Sie die 'B'-Taste, und der Panzer wird in horizontaler Richtung 2fach vergrößert. Sie bemerken, daß der LEN Wert jetzt doppelt so groß wie sein ursprünglicher Wert ist.

Drucken Sie 'SHIFT' und 'B' und der Panzer wird in vertikaler Richtung 2fach vergroßert. Wenn durch die vergroßerung das Ergebnis großer als der Spriteanzeigebereich sein sollte, dann wird keine Operation stattfinden.

TRICKBEWEGUNG

Die meisten Arkadenspiele benötigen eine Reihe von beweglichen Sprites. Das Spritegeneratorprogramm bietet eine Möglichkeit, den Ablauf einer Trickbewegung zu betrachten.

Kehren Sie in das Hauptmenü zurück, indem Sie in den Zeichenmodus gehen und SHIFT 'R' drücken.

Im Demo sind viele Beispiele zur Trickbewegung von Sprites vorhanden. Wir werden jetzt die Sprites SPT3 laden.

Bandbenutzer werden das Demo-Band benutzen müssen.

Wählen Sie Option 5 und tippen Sie SPT3. Drücken Sie 'N' nach der Aufforderung 'change sprite max value' (Spritehöchstwert ändern). Wenn Sie wieder im Hauptmenü sind, sollen Sie Option 8 wählen.

Sie werden sehen, daß sich der Cursor ('>') in der linken oberen Bildschirmecke befindet. Die Sprites, die wir fur die Trickeffekte verwenden sind die Sprites 30 bis 33 (die 'Augäpfel').

Tippen Sie 30 gefolgt von ENTER: Bewegen Sie jetzt den Cursor mit der 'D'-Taste zur Position 2 und tippen Sie 31 gefolgt von ENTER. Bewegen Sie den Cursor mit 'A' und anschließend mit der 'X'-Taste zur Position 3 und tippen Sie 32. Geben Sie zum Schluß 33 in Position 4 ein.

Bewegen Sie den Cursor zur Position 5 und drücken Sie 'R'.

Die Daten für die Trickbewegung sind jetzt eingegeben und können nun verwendet werden. Drücken Sie die LEERTASTE, um die Trickbewegung auszuführen.

Tippen Sie SHIFT 'R', um zum Hauptmenü zurückkehren.

NOCH MEHR FORTGESCHRITTENE FUNKTIONEN

Kehren Sie zum Hauptmenü zurück und wählen Sie Option 2.

MASKIEREN UND UMMASKIEREN

Sprites die mit FMOV bzw. BMOV bewegt werden sollen mussen MASKiert werden. Sie können dies in Ihrem Laser-BASIC-Programm bzw. im Spritegeneratorprogramm durchfuhren

Stellen Sie SPN auf 10 und drücken Sie 'L', um Sprite 10 in den Spriteanzeigebereich zu bringen Sie werden sofort eine Reihe von vertikalen Linien bemerken konnen, die darauf hindeuten, daß das Sprite bereits MASKiert ist. Wenn Sie SHIFT 'M' drücken, sollten Sie einen hohen Piepton vernehmen können, was bedeutet, daß die Operation erfolgreich war. Sie haben jetzt das Sprite im Speicher entMASKiert. Drücken Sie 'L', um das Sprite zuruckzubringen und das Ergebnis zu sehen

Piepton sollte andeuten, daß die Operation erfolgreich war. Drücken Sie jetzt 'L', und das MASKierte Sprite wird angezeigt werden.

NUMERISCHE DATENEINGABE

Wie bereits in einem früheren Abschnitt dieses Handbuchs erwähnt wurde, können Sprites Daten für mehrere Anwendungsfälle enthalten. Wenn Sie die LEERTASTE drücken, um in den Zeichenmodus zurückzukehren, und danach 'E' drücken, dann wird 'DECIMAL OR BINARY' (Dezimal-oder Binärform) erscheinen. Drücken Sie 'D' für DECIMAL und 'Byte 1' wird angezeigt. Tippen Sie also 1 ein und drücken Sie ENTER, gefolgt von 2, 4, 8, 16, 32, 64 und 128.

Die Daten, die Sie soeben eingegeben haben, werden an der Spritebildschirmcursorposition erscheinen.

SPEICHERN IHRER ENDGÜLTIGEN SPRITES

Kehren Sie zum Hauptmenü zurück, nachdem Sie Ihre Sprites erzeugt haben. Schalten Sie Option 9 ein und wählen Sie je nach System entweder 'DISC' (Diskette) oder 'TAPE'. Dies kann durch Drücken der Taste '9' erzielt werden.

Drücken Sie jetzt '4', um die Option 4 zu wählen und den Dateinamen einzugeben. Denken Sie daran, daß Sie Ihrem Dateinamen 'SPR' nicht hinzuzufügen brauchen, da dies für Sie gemacht wird. Z.B.: Wenn Sie 'FRED' eingeben wird eine Spritesdatei mit dem Namen 'FREDSPR' gespeichert werden.

Wenn Sie zufällig das Spritegeneratorprogramm verlassen, sollten Sie die Sprites manuell mittels PSPR abspeichern, bevor Sie RUN tippen. Wenn Sie das Programm laufen lassen, werden sämtliche Sprites vom Speicher gelöscht

FUNKTIONSTASTENÜBERBLICK

HAUPTMENÜOPTIONEN:

OPTIONEN	FUNKTIONEN
1,2,3	Ermöglichen die Erzeugung von Sprites.

Zeichenmodus

TASTEN

A	für Bewegung nach links.
D	für Bewegung nach rechts.
W	für Bewegung nach oben.
X	für Bewegung nach unten.
SHIFT mit (A,D,W,X)	Wird den Cursor mit Spriteanzeigebereich bewegen.
S	Bring ein Zeichen in den Spriteanzeigebereich
L	Nimmt ein Zeichen aus dem Spriteanzeigebereich
I	Ändert die Farbe des laufenden INK-Wertes
C	Löscht das Zeichenquadrat

LEERTASTE	lt das Spritegeneratorprogramm auf Spritemodus.
P	Bewegt den Pfeil, der den INK-Wert anzeigt, nach unten.
SHIFT und P	Bewegt den Pfeil, der den INK-Wert anzeigt, nach oben.
J	Zeigt den Farbwert sämtlicher INKs an.
B	Ändert die Randfarbe.
CLR	Bringt beide Cursors in die Grundstellung zurück.
ENTER	Stellt einen Bildpunkt auf den laufenden INK-Wert ein.
DEL	Stellt einen Bildpunkt auf Papierfarbe ein.
E	Eintritt von Sprite is den Spriteanzeigebereich.
R und SHIFT	Bringt Sie zum Hauptmenü zurück.

Spritemodus

KEYS

A	für Bewegung nach links.
D	für Bewegung nach rechts.
W	für Bewegung nach oben.
X	für Bewegung nach unten.
SHIFT und A	zur Verringerung der Fensterlänge.
SHIFT und D	zur Erhöhung der Fensterlänge.
SHIFT und W	zur Verringerung der Fensterhöhe.
SHIFT und X	zur Erhöhung der Fensterhöhe.
@	Ändert die X-Zunahme für die Bewegung des Spriteanzeigefensters.
;	Ändert die Y-Zunahme für die Bewegung des Spriteanzeigefensters.
L	Bringt das Sprite in den Spriteanzeigebereich.
S	Speichert ein Sprite in den Speicher.
P	Ändert Druckmodus.
N	Erhöht laufende Spritenummer.
SHIFT und N	Verringert laufende Spritenummer.
F	Füllt einen von Bildpunkten umgebenen Bereich.
SHIFT und F	Bedeckt das Spriteanzeigefenster mit dem laufenden Schreiber.
SHIFT und E	Löscht ein Sprite.
O	Schaltet Wrap (Umlauf) ein und aus
↓	Rollt nach unten
→	Rollt nach rechts.
←	Rollt nach links.
↑	Rollt nach oben

SHIFT und ←	Kippt den Inhalt des Fensters um die Y-Achs
SHIFT und ↑	Kippt den Inhalt des Fensters um die X-Achse.
SHIFT und →	Kippt den Inhalt des Fenster in die rechte Fensterhälfte.
SHIFT und ↓	Kippt den Inhalt des Fensters in die untere Fensterhälfte.
B	Vergrößert das Fenster in horizontaler Richtung.
SHIFT und B	Vergrößert das Fenster in vertikaler Richtung.
T	Dreht das Fenster um 90 Grad.
Z	Invertiert den Fensterinhalt.
H	Erzeugt im Speicher ein großes Sprite.
G	Speichert den Fesnterinhalt in das große Sprite.
V	Ladet einen Teil des großen Sprite vom Speicher den Spriteanzeigebereich.
[	Schrittänderung bei horizontalen Rollen.
]	Schrittänderung beim vertikalen Rollen.
K	ändert einen bestimmten INK-Wert in einem Spriteanzeigefenster auf eine andere Farbe.
M	Maskiert ein Sprite.
SHIFT und M	Entmaskiert ein Sprite.
:	Läßt das laufende Spriteanzeigefenster aufblinken.
I	Zeigt Informationen an.
SHIFT und R	Kehrt zum Hauptmenü zurück.

Trickbewegung Ihres Sprites

OPTION	FUNKTION
8	
TASTEN	
A	für Bewegung nach links.
D	für Bewegung nach rechts.
W	für Bewegung nach oben.
X	für Bewegung nach unten.
SHIFT und (A,D,W,X)	Bewegt Cursor in Spriteanzeigebereich.
LEERTASTE	Beginnt und stoppt Trickbewegungsablauf.
SHIFT und R	Kehrt zum Hauptmenü zurück.

ANHANG A- SPRITES SPT1

Um die Sprites in LASER-BASIC zu laden, tippen Sie: **A$="SPT1SPR":|GSPR,@A$**

Sprite Nr.	Bezeichnung	HGT	LEN	Empfohlene Farbwerte für INKS 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Leichte Panzer	19	9	0	13	12	-	-	-	-	-	-	-	-	-	-	-	-
2	Zero fighter – 2. Weltkrieg	13	8	0	23	6	-	-	-	-	-	-	-	-	-	-	-	-
3	Sturzkampfflugzeug-2. Weltkrieg	16	8	0	9	11	-	-	-	-	-	-	-	-	-	-	-	-
4	Armeehubschrauber	15	11	0	9	13	-	-	-	-	-	-	-	-	-	-	-	-
5	Gespenst	22	7	0	18	9	14	6	0	0	0	0	0	0	0	0	0	0
6	Erdbeervariante	20	5	0	16	6	26	3	9	18	2	0	0	0	0	0	0	0
7	Orangenvariante	16	6	0	14	15	26	3	9	0	0	0	0	0	0	0	0	0
8	Weltraumkreuzer	19	10	0	25	15	-	-	-	-	-	-	-	-	-	-	-	-
9	Weltraumaufklärungsschiff	19	9	0	14	23	-	-	-	-	-	-	-	-	-	-	-	-
10	Weltraumtransporter	14	11	0	0	8	-	-	-	-	-	-	-	-	-	-	-	-
11	Tiefsee-U-Boot	18	15	0	24	15	-	-	-	-	-	-	-	-	-	-	-	-
12	Mondrover	15	10	0	10	15	-	-	-	-	-	-	-	-	-	-	-	-
13	Mondtransporter	18	14	0	13	15	-	-	-	-	-	-	-	-	-	-	-	-
14	Jet-Interpreter	16	13	0	12	13	-	-	-	-	-	-	-	-	-	-	-	-
15	Außerirdischer	22	6	0	2	11	26	6	-	-	-	-	-	-	-	-	-	-
16	Sportauto	15	13	0	2	11	-	-	-	-	-	-	-	-	-	-	-	-
17	Mondforscher	21	10	0	5	12	-	-	-	-	-	-	-	-	-	-	-	-
18	Marsforscher	27	16	0	13	14	-	-	-	-	-	-	-	-	-	-	-	-
19	Marsprobe	35	10	0	3	24	-	-	-	-	-	-	-	-	-	-	-	-
20	Hubschrauber-transporter	19	15	0	9	12	-	-	-	-	-	-	-	-	-	-	-	-
21	Hubschrauber-Kanonenboot	17	15	0	12	11	-	-	-	-	-	-	-	-	-	-	-	-
22	Düsenhubschrauber	16	15	0	13	11	-	-	-	-	-	-	-	-	-	-	-	-
23	Düsenjäger	16	13	0	9	12	-	-	-	-	-	-	-	-	-	-	-	-
24	Salonwagen	16	14	0	6	13	-	-	-	-	-	-	-	-	-	-	-	-
25	Shuttle	20	13	0	26	13	-	-	-	-	-	-	-	-	-	-	-	-

ANHANG B SPRITES SPT2

Um die Sprites in Laser-BASIC zu laden, tippen Sie: A$="SPT2SPR":|GSPR,@A$

Sprite Nr.	Bezeichnung	HGT	LEN	Empfohlene Farbwerte für INKs 1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	Spinne	16	4	14	6	0	-	-	-	-	-	-	-	-	-	-	-	-
2	Droid	16	4	15	18	0	-	-	-	-	-	-	-	-	-	-	-	-
3	Herz	16	8	0	0	6	-	-	-	-	-	-	-	-	-	-	-	-
4	In der Probesitzung definiert	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
5	In der Probesitzung definiert	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
6	OCEAN IQ LOGO	32	28	2	26	11	-	-	-	-	-	-	-	-	-	-	-	-
7	Ball	8	2	18	0	9	-	-	-	-	-	-	-	-	-	-	-	-
8	Joystick	16	4	6	0	13	-	-	-	-	-	-	-	-	-	-	-	-
9	Gesicht	16	4	16	15	0	-	-	-	-	-	-	-	-	-	-	-	-
10	Teddybär	40	8	15	17	0	-	-	-	-	-	-	-	-	-	-	-	-
11	Männchen	16	4	16	7	9	-	-	-	-	-	-	-	-	-	-	-	-
12	Oldtimer	16	16	0	13	14	-	-	-	-	-	-	-	-	-	-	-	-
13	Dragster	16	16	0	15	18	-	-	-	-	-	-	-	-	-	-	-	-
14	Doppeldecker	16	3	13	0	0	-	-	-	-	-	-	-	-	-	-	-	-
15	Blindsprite	18	9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
16	In der Probesitzung definiert	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
17	Pflanzenvariante #1	22	7	19	0	9	-	-	-	-	-	-	-	-	-	-	-	-
18	Pflanzenvariante #2	22	7	19	0	9	-	-	-	-	-	-	-	-	-	-	-	-
19	Pflanzenvariante #3	22	7	19	0	9	-	-	-	-	-	-	-	-	-	-	-	-
20	Pflanzenvariante #4	22	7	19	0	9	-	-	-	-	-	-	-	-	-	-	-	-
21	Gehender Roboter #1	16	6	0	26	10	-	-	-	-	-	-	-	-	-	-	-	-
22	Gehender Roboter #2	16	6	0	26	10	-	-	-	-	-	-	-	-	-	-	-	-

23	Gehender Roboter #3	16	6	0	26	10	-	-	-	-	-	-	-	-	-	-	-	-
24	Gehender Roboter #4	16	6	0	26	10	-	-	-	-	-	-	-	-	-	-	-	-
25	Schere #1	20	8	13	0	0	-	-	-	-	-	-	-	-	-	-	-	-
26	Schere #2	20	8	13	0	0	-	-	-	-	-	-	-	-	-	-	-	-
27	Schere #3	20	8	13	0	0	-	-	-	-	-	-	-	-	-	-	-	-
28	Schere #4	20	8	13	0	0	-	-	-	-	-	-	-	-	-	-	-	-
29	Undefiniert	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
30	In der Probesitzung definiert	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
31	Backsteine	25	10	0	0	6	-	-	-	-	-	-	-	-	-	-	-	-
32	Blindsprite	25	10	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
33	Der Buchstabe 'A'	8	2	24	0	0	-	-	-	-	-	-	-	-	-	-	-	-
34	Blindsprite	8	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
35	Der Buchstabe 'a'	8	2	24	0	0	-	-	-	-	-	-	-	-	-	-	-	-
36	Blindsprite	2	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
37	Der Buchstabe 'a'	8	2	24	0	0	-	-	-	-	-	-	-	-	-	-	-	-
38	Leerer Block	8	2	24	0	0	-	-	-	-	-	-	-	-	-	-	-	-
39	Banane	17	5	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0
40	Datensprite	1	13	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
41	Datensprite	1	11	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
42	Datensprite	1	170	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
43	Datensprite	1	170	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
44	Datensprite	1	19	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
45	Datensprite	1	7	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
46	Planet mit Mond #1	16	4	11	26	6	-	-	-	-	-	-	-	-	-	-	-	-
47	Planet mit Mond #2	16	4	11	26	6	-	-	-	-	-	-	-	-	-	-	-	-
48	Planet mit Mond #3	16	4	11	26	6	-	-	-	-	-	-	-	-	-	-	-	-
49	Planet mit Mond #4	16	4	11	26	6	-	-	-	-	-	-	-	-	-	-	-	-
50	Ort	13	3	23	0	0	-	-	-	-	-	-	-	-	-	-	-	-
51	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
52	Pfeilleck	24	6	25	6	0	-	-	-	-	-	-	-	-	-	-	-	-
53	Katze	22	7	0	12	0	-	-	-	-	-	-	-	-	-	-	-	-
54	Vogel	25	5	0	0	4	-	-	-	-	-	-	-	-	-	-	-	-

ANHANG C (i) – SPRITES SPT3 (Sprites, die im Demo, Teil 1, verwendet werden).

n die Sprites in Laser BASIC zu laden, tippen Sie: A$="SPT3SPR":|GSPR,@ A$

.en Sie darauf, daß alle Sprites, die ein (M) nachgestellt haben, MASKiert sind und mittels DMSK entMASKiert werden können.

Sprite Nr.	Bezeichnung	Empfohlene, alternative Farbwerte für die INKs															
		HGT	LEN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1.	Schildkröte	17	8	0	6	12	-	-	-	-	-	-	-	-	-	-	-	-
2.	Ratte	17	9	0	6	10	-	-	-	-	-	-	-	-	-	-	-	-
3.	Hase	17	8	0	20	15	-	-	-	-	-	-	-	-	-	-	-	-
4.	Steinbodenteil	8	7	0	11	9	-	-	-	-	-	-	-	-	-	-	-	-
5.	Flaschenzug	23	7	0	11	7	-	-	-	-	-	-	-	-	-	-	-	-
7.	Würfel	8	2	0	16	20	-	-	-	-	-	-	-	-	-	-	-	-
8	Weltraumaufklärugsschiff	23	6	0	10.	16	-	-	-	-	-	-	-	-	-	-	-	-
9	Außerirdischer	24	10	0	8	0	-	-	-	-	-	-	-	-	-	-	-	-
10	Hupfender Außerirdischer #1(M)	33	10	0	8	0	-	-	-	-	-	-	-	-	-	-	-	-
11	Hupfender Außerirdischer #2(M)	33	10	0	8	0	-	-	-	-	-	-	-	-	-	-	-	-
12	Hupfender Außerirdischer #3(M)	33	10	0	8	0	-	-	-	-	-	-	-	-	-	-	-	-
13	Hupfender Außerirdischer #4(M)	33	10	0	8	0	-	-	-	-	-	-	-	-	-	-	-	-
14	Quadrat	16	4	0	0	5	-	-	-	-	-	-	-	-	-	-	-	-
15	Steinmauerteil	24	10	0	0	9	-	-	-	-	-	-	-	-	-	-	-	-
16	Leiterteil	24	10	19	0	0	-	-	-	-	-	-	-	-	-	-	-	-
17	Datensprite	8	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
18	Datensprite	8	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
19	Datensprite	10	18	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
20	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
21	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
23	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
24	Jagender Außerirdischer #1(M)	33	14	0	0	14	-	-	-	-	-	-	-	-	-	-	-	-
25	Jagender Außerirdischer #2(M)	33	14	0	0	14	-	-	-	-	-	-	-	-	-	-	-	-
26	Jagender Außerirdischer #3(M)	33	14	0	0	14	-	-	-	-	-	-	-	-	-	-	-	-
27	Jagender Außenirdischer #4(M)	33	14	0	0	14	-	-	-	-	-	-	-	-	-	-	-	-
28	Langer Würfel	15	5	0	3	6	-	-	-	-	-	-	-	-	-	-	-	-
29	Datensprite	3	2	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
30	Augapfel #1	16	6	0	16	9	-	-	-	-	-	-	-	-	-	-	-	-
31	Augapfel #1	16	6	0	16	9	-	-	-	-	-	-	-	-	-	-	-	-
32	Augapfel #1	16	6	0	16	9	-	-	-	-	-	-	-	-	-	-	-	-
33	Augapfel #1	16	6	0	16	9	-	-	-	-	-	-	-	-	-	-	-	-
34	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
35	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
36	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
37	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
40	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
41	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
42	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
43	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
44	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
45	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
46	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
47	Datensprite	1	4	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
49	Kleiner Leiterteil	8	3	0	11	0	-	-	-	-	-	-	-	-	-	-	-	-
50	Korkenzieher #1	20	4	0	7	3	-	-	-	-	-	-	-	-	-	-	-	-
51	Korkenzieher #2	20	4	0	7	3	-	-	-	-	-	-	-	-	-	-	-	-
52	Korkenzieher #3	20	4	0	7	3	-	-	-	-	-	-	-	-	-	-	-	-
53	Korkenzieher	20	4	0	7	3	-	-	-	-	-	-	-	-	-	-	-	-
55	Bewegliches Klosett #1	19	7	14	23	24	-	-	-	-	-	-	-	-	-	-	-	-
56	Bewegliches Klosett #2	19	7	14	23	24	-	-	-	-	-	-	-	-	-	-	-	-
57	Bewegliches Klosett #3	19	7	14	23	24	-	-	-	-	-	-	-	-	-	-	-	-
58	Bewegliches Klosett #4	19	7	14	23	24	-	-	-	-	-	-	-	-	-	-	-	-
63	Datensprite	8	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
64	Datensprite	8	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
65	Datensprite	8	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
66	Datensprite	8	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
67	Datensprite	8	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
68	Datensprite	8	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
69	Datensprite	8	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

...ie Sprites in Laser-BASIC zu laden, tippen Sie: A$= S...

Sprite-Nr.	Bezeichnung	Empfohlene, alternative Farbwerte für die INKs															
		HGT	LEN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	OCEAN IQ Logo	33	28	9	21	23	-	-	-	-	-	-	-	-	-	-	-	-
70	Steinziegel	16	8	0	0	13	12	0	0	0	0	0	0	0	0	0	0	0
71	Steinziegel mit Unkraut	16	8	0	24	13	12	1	0	0	0	0	0	0	0	0	0	0
72	Gehender Mönch #1(M)	26	10	0	0	0	0	0	12	25	15	0	13	0	0	0	0	0
73	Gehender Mönch #2(M)	26	10	0	0	0	0	0	12	25	15	0	13	0	0	0	0	0
74	Gehender Mönch #3(M)	26	12	0	0	0	0	0	12	25	15	0	13	0	0	0	0	0
75	Kletternder Mönch #1(M)	27	12	0	0	0	0	0	12	25	15	0	13	0	0	0	0	0
76	Kletternder Mönch #2(M)	27	10	0	0	0	0	0	12	25	15	0	13	0	0	0	0	0
77	Kletternder Mönch #3(M)	26	12	0	0	0	0	0	12	25	15	0	13	0	0	0	0	0
78	Steinfenster	36	7	0	0	12	25	0	0	0	0	0	0	0	0	0	0	0
79	Lanzenträger #1	38	5	0	0	15	3	12	9	0	0	19	16	0	0	0	0	0
80	Lanzenträger #2	38	5	0	0	15	3	12	9	0	0	19	16	0	0	0	0	0
81	Lanzenträger #3	40	5	0	0	15	3	12	9	0	0	19	16	0	0	0	0	0
82	Zeigender Mönch	29	5	0	0	0	0	0	12	25	15	0	13	0	0	0	0	0
83	Mönchshand	6	5	0	0	0	0	0	12	25	15	0	13	0	0	0	0	0
84	Frau	35	3	0	11	0	0	2	1	26	15	21	0	0	0	0	0	0
96	Raketenflugzeug #1(M)	14	24	0	0	0	0	14	24	25	0	16	0	0	0	0	0	0
97	Raketenflugzeug #2(M)	14	24	0	0	0	0	14	24	25	0	16	0	0	0	0	0	0
98	"Beliebige Taste drücken" (M)	8	50	14	0	-	-	-	-	-	-	-	-	-	-	-	-	-
100	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
101	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
102	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
103	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
104	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
105	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1[illegible]	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
1[illegible]	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
108	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
109	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
110	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
111	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
112	Datensprite	1	20	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

ANHANG C(iii) SPRITES SPT5 (Sprites, die im Demo, Teil 5, verwendet werden)

Um die Sprites in LASER BASIC zu laden, tippen Sie: A$="SPT5SPR":|GSPR,@A$

Sprite Nr.	Bezeichnung	Empfohlene, alternative Farbwerte für die INKs															
		HGT	LEN	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
60	Plattformspiel Kanal A Musiksprite	7	49	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Plattformspiel Kanal B Musiksprite	1	128	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
62	Plattformspiel Kanal C Musiksprite	1	152	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
86	Ladebildschirm Kanal A Musiksprite	2	234	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
87	Ladebildschirm Kanal B Musiksprite	1	94	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
88	Ladebildschirm Kanal C Musikksprite	1	94	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
90	Ocean IQ Bildschirm Kanal A Musiksprite	3	119	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
91	Ocean IQ Bildschirm Kanal B Musiksprite	2	128	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
92	Oceam IQ Bildschirm Kanal C Musiksprite	1	75	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
93	Klöckner-Bildschirm Kanal A Musiksprite	3	91	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
94	Klöckner-Bildschirm Kanal B Musiksprite	1	237	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
95	Klöckner-Bildschirm Kanal C Musiksprite	2	122	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-