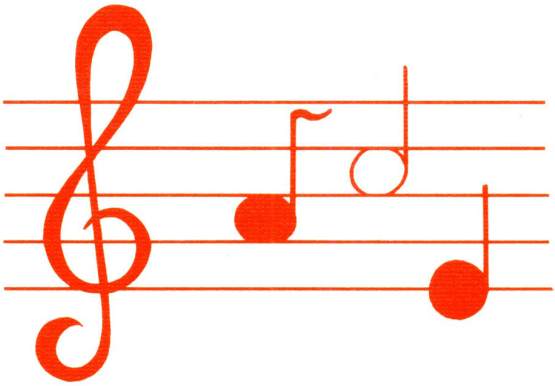
Dachsel

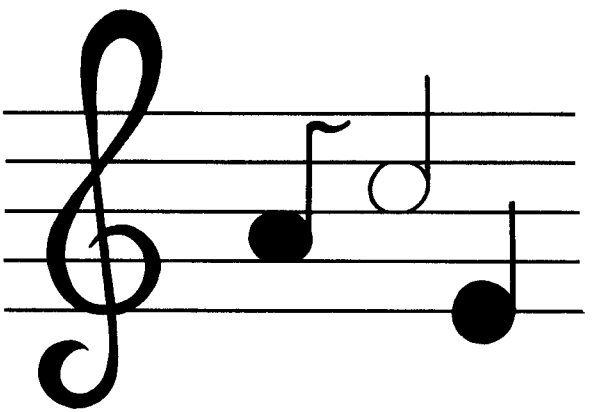
DAS MUSIKBUCH ZUM COMMODORE 64



EIN DATA BECKER BUCH

Dachsel

DAS MUSIKBUCH ZUM COMMODORE 64



EIN DATA BECKER BUCH

ISBN 3-89011-012-6

Wichtiger Hinweis

Die in diesem Buch wiedergegebenen Schaltungen, Verfahren und Programme werden ohne Rücksicht auf die Patentlage mitgeteilt. Sie sind ausschließlich für Amateur- und Lehrzwecke bestimmt und dürfen nicht gewerblich genutzt werden.

Alle Schaltungen, technische Angaben und Programme in diesem Buch wurden von den Autoren mit größter Sorgfalt erarbeitet bzw. zusammengestellt und unter Einschaltung wirksamer Kontrollmaßnahmen reproduziert. Trotzdem sind Fehler nicht ganz auszuschließen. DATA BECKER sieht sich deshalb gezwungen, darauf hinzuweisen, daß weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgen, die auf fehlerhafte Angaben zurückgehen, übernommen werden kann. Für die Mitteilung eventueller Fehler sind die Autoren jederzeit dankbar.

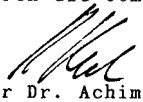
Vorwort

Die überragenden musikalischen Fähigkeiten des Commodore-64 fordern seinen Einsatz zur Musikprogrammierung geradezu heraus. Leider werden diese Möglichkeiten durch das Commodore-BASIC V2 nicht unterstützt.

Daher ist es sehr erfreulich, daß Thomas Dachsel von DATA BECKER als Autor für dieses Buch gewonnen werden konnte. Er hat bereits das Programm SYNTHIMAT geschrieben, dessen unglaubliche Leistungen selbst eingefleischte C-64 Freaks in Erstaunen versetzte. Zudem ist er Musiker, der einige Musikstücke für DATA BECKER neu arrangierte. Sie sind bei den zahlreichen Listings, die im Musikbuch enthalten sind.

In diesem Buch ist von den ersten Schritten der Musikprogrammierung in BASIC bis zu komplexen Musikstücken in Assembler fast alles enthalten, was mit dem C-64 machbar ist. Daneben gewinnt man einen Einblick in die Funktionsweise der Soundregister; Hinweise zum Anschluß des C-64 an Ihre Stereoanlage und ein kleines Lexikon zur Computermusik runden dieses Buch ab.

Und nun viel Spaß bei Ihrer musikalischen Entdeckungsreise durch die Computerwelt !



Ihr Dr. Achim Becker

Inhaltsverzeichnis

1. Einführung in die Computermusik

Lieber Leser,

Sie haben sich dieses Buch gekauft, um etwas über den Commodore 64 und seine Musikmöglichkeiten zu erfahren. Es wird Sie interessieren, wie man den Computer dazu bringt, Melodien zu spielen. Vielleicht wollen Sie auch wissen, wie man Effektklänge erzeugt, die man bei einem Spielprogramm verwenden kann. Sie können aber auch mit diesem Computer komponieren und Klangexperimente anstellen, was kaum ein anderer Computer in dieser Preisklasse kann.

Nun gibt es einige Leute, die gleich sagen: "Aber das ist doch alles nur Computermusik. Viel mehr als ein Spielzeug kann das niemals sein." Woher kommt diese Skepsis gegenüber Musik "aus dem Computer"? Was ist eigentlich so besonders an der "Computermusik"?

Bevor man diese Frage beantworten kann, sollten wir uns einmal vergegenwärtigen, was für Musik wir denn als "normal" ansehen. Jeder von uns wird davon eine andere Vorstellung haben. Gibt es denn eine "normale" Musik, die uns vielleicht vertrauter erscheint als die Musik, die uns ein Computer vorspielt?

Um alle diese Fragen zu beantworten: Musik ist etwas, was von Menschen gemacht wird und uns ein hörbares Resultat liefert. Nach dieser Definition produziert eigentlich auch ein Preßlufthammer Musik, also sollte man vielleicht diese Definition einschränken. Probieren wir das aus: Man könnte sagen: Nur das ist Musik, was für unsere Ohren melodisch klingt. Doch dann darf man einen Großteil der außereuropäischen Musik, zum Beispiel indische Musik, die ja für uns total "schräg" klingt, eigentlich nicht mehr als "Musik" bezeichnen. Man könnte bestimmte Lautstärkegrenzen festsetzen. So könnte man sagen: Was lauter ist als 80 Phon, ist keine Musik mehr.

Wie sieht es aber dann mit einem Hard-Rock-Konzert aus, wo 80 Phon durchaus normal sind? Ältere Menschen sagen dann auch oft: "Das ist ja keine Musik mehr!". Doch für jüngere Menschen gehören solche Lautstärken durchaus zu ihrer Lieblingsmusik.

Um es kurz zu fassen: So etwas wie "normale" Musik gibt es nicht, jedenfalls nicht als allgemeingültigen Begriff. Zwar hat jeder von uns seine Vorstellung von "normaler" Musik, aber, wie wir gesehen haben, ist dies abhängig vom Alter, vom Kulturkreis, in dem man lebt, und nicht zuletzt vom persönlichen Musikgeschmack, über den man ja nicht streiten sollte.

So kommen wir auf unsere Definition zurück, um allgemeine Eigenschaften der Musik zu erkennen und daraus zu ersehen, wieso Computermusik so "besonders" ist. Am Anfang eines jeden Musikstücks steht etwas, was wir direkt nicht sehen, hören oder sonst wahrnehmen können: die Idee. Diese Idee ist es, wonach jeder sucht, der ein Musikstück verfassen will. Fällt einem Komponisten nichts ein, so kann er auch nichts komponieren. Und damit ein Musikstück gespielt werden kann, muß es zuallererst komponiert werden.

Angenommen, ein Komponist sucht nach einer Idee. Er sitzt stundenlang vor einem Klavier und spielt bekannte Melodien, Tonleitern und ähnliches. Aber je krampfhafter er sich bemüht, etwas Neues zu finden, ihm fällt nichts ein. Frustiert hört er auf zu spielen und tut etwas anderes, zum Beispiel betrachtet er ein Gemälde oder macht einen Spaziergang (so etwas nennt man Inspiration). Dann kann es mitunter passieren, daß ihm ein kompletter Satz einer Sinfonie einfällt. Er muß sich nur wieder ans Klavier setzen und das Stück aufschreiben.

Ein Musikstück aufschreiben: das ist etwas, was bei jeder Musikart zwingend notwendig ist. Ohne zumindestens Notizen zu machen, wird kein Komponist ein Stück komponieren können.

Die Art, wie Musik aufgeschrieben wird, ist jedoch von Kulturkreis zu Kulturkreis, ja sogar von Instrument zu Instrument verschieden. Wieso denn auch beim Instrument? Das ist ganz einfach: Jedes Instrument hat einen anderen Tonumfang, d.h. es kann nur eine ganz bestimmte Anzahl von Tönen spielen. Ein Kontrabaß kann eben nur tiefe Töne, eine Violine nur hohe Töne spielen. Beim Klavier haben wir jedoch tiefe und hohe Töne.

Deswegen hat sich für jedes Instrument eine andere Art eingebürgert, Noten dafür aufzuschreiben. Beim Kontrabaß notiert man im Baßschlüssel für die tiefen Töne, bei einer Violine im Violinschlüssel für die hohen Töne. Bei einem Klavier werden sowohl Baß- als auch Violinschlüssel verwendet, da mit einem Klavier sowohl hohe als auch tiefe Töne gespielt werden können.

Noten, Violinschlüssel, Baßschlüssel: das sind Begriffe aus der Notenschrift, die in Europa um 1000 n.Chr. in ihren Frühformen entstand und bis heute immer weiterentwickelt wurde. Das soll aber nicht heißen, daß es nur diese einzige Notenschrift gibt: In anderen Kulturen wurden ähnlich komplexe Systeme entwickelt, um musikalische Informationen schriftlich festzuhalten, genau wie jedes Volk eine eigene Schrift hat, um sprachliche Informationen festzuhalten. Hier sind wir auf eine interessante Parallele zwischen Musik und Sprache gestoßen: Beide menschlichen Errungenschaften bedürfen eines Schriftsystems, um über gewisse Zeitperioden hinweg fixierbar zu sein.

Da jedoch die Notenschrift nie alle Informationen exakt kodieren kann, die ein Musikstück auszeichnen, gibt es verschiedene Arten, es zu spielen. Dies nennt man die Interpretation eines Stückes. Auf die Interpretation wird hauptsächlich in der klassischen Musik geachtet, in der es darauf ankommt, ein Stück so nachzuspielen, wie der Komponist es sich vorgestellt hat. Man versucht so dem Werk "treu zu bleiben". Deswegen spricht man im Bereich der klassischen Musik viel von "Werktreue".

Bei der klassischen Musik haben wir als erstes Bindeglied zwischen dem Komponisten und dem hörbaren Musikstück den Dirigenten. Dieser muß versuchen, die vielen Ungenauigkeiten und Unklarheiten, die sich beim Aufführen eines durch die Notenschrift nur unvollständig beschriebenen Musikstücks ergeben können, in den Griff zu bekommen.

Im Jazz kommt es nicht auf Werktreue an. Hier werden bekannte Stücke oft umarrangiert. Dem Arrangeur kommt im Jazz ein ähnlicher Stellenwert wie bei der klassischen Musik dem Dirigenten. Der Arrangeur übernimmt zwar auch oft bekannte Stücke, verändert sie aber nach seinen Vorstellungen, fügt neue Teile ein und verändert die Instrumentation. Der Arrangeur spielt - im Gegensatz zum Dirigenten - oft in seiner Gruppe mit, meistens spielt er ein markantes Soloinstrument.

In der Pop- und Rockmusik braucht man keinen Dirigenten, denn hier hat jede Gruppe ihre eigenen Stücke, die gemeinsam komponiert werden. Jeder ist für sein eigenes Instrument verantwortlich, so daß jeder Musiker sein eigener Dirigent oder Arrangeur ist. Damit aber alle Instrumente zusammenpassen, müssen sich die Musiker untereinander beim Proben absprechen, was man in der klassischen Musik nicht findet.

Das zweite Bindeglied zwischen Komponist und fertigem Musikstück ist der Interpret, der gemeinhin auch "Musiker" genannt wird, da auf den ersten Blick er die Musik zu machen scheint. Doch der Interpret vollzieht nur nach, was andere vor ihm gedacht haben, er hat also nur eine reproduktive Funktion. Eine Ausnahme hiervon ist die Improvisation, wo der oder die Musiker, die improvisieren, Komponisten, Arrangeure und Musiker in einem sind. Musiker, die gut improvisieren sollen, müssen ihr Instrument perfekt beherrschen, um ihre Inspirationen ohne technische Schwierigkeiten sofort in ein hörbares Ergebnis umsetzen zu können. Wenn mehrere Musiker zusammen improvisieren (sogenannte Kollektivimprovisationen, wie man sie im Free Jazz findet), kommt zusätzlich hinzu, daß die Musiker sich untereinander bestens kennen und aufeinander reagieren können.

Das letzte Bindeglied zwischen Komponist und fertigem Stück ist das Instrument. Bevor ein Musiker sein Instrument spielen kann, muß er erst die technischen Fähigkeiten erwerben, um die Vorgaben, die er durch die Notenschrift erhält, erfüllen zu können. Diese Fähigkeiten muß sich jeder Musiker durch einen jahrelangen, oft sehr mühsamen Lernprozeß erarbeiten. Daher wird man kaum jemanden finden, der gleichermaßen gut komponieren und sein Instrument beherrschen kann. Ein Komponist ist allenfalls ein guter Arrangeur, aber von den technischen Voraussetzungen braucht er immer Musiker, die seine Kompositionen ausführen. Für einen Komponisten der Klassik war es ja unmöglich, sämtliche Instrumente des Orchesters spielen zu können. Sogenannte Multiinstrumentalisten, die mehr als ein Instrument beherrschen, gibt es weniger als Leute, die (mehr oder weniger gut) komponieren können.

Nachdem wir nun das Verhältnis Komponist Arrangeur - Interpret - Instrument - Musikstück in groben Zügen erarbeitet haben, wollen wir uns nun der Frage zuwenden, was denn in der Computermusik daran anders ist.

Egal, ob wir uns in der herkömmlichen Musik oder in der Computermusik befinden, Komponist ist und bleibt der Mensch. Kein Computer würde irgendeinen Mucks von sich geben, würde man ihn in die Wüste stellen und ihm sagen, er solle ein Musikstück spielen. Weiterhin findet der Vorgang des Komponierens nie am Computer selbst statt. Derjenige, der einem Computer ein Musikstück einprogrammiert, nimmt dafür meistens ein ihm schon bekanntes Musikstück. Dies ist aber schon lange komponiert worden, bevor es jemandem einfiel, es auf einen Computer zu übertragen.

An dieser Stelle muß man eine Besonderheit der Computermusik erwähnen: Es gibt Programme, die den Computer Musik "komponieren" lassen. Diese Programme rechnen nach bestimmten Vorgaben, die der Programmierer ausgetüftelt hat, zufällige Melodien aus.

Trotzdem kann man auch in diesem Fall nicht davon sprechen, daß der Computer kreativ tätig ist (was Grundvoraussetzung jedes Komponierens ist). Die kreative Arbeit hat vorher immer noch ein Mensch geleistet, indem er einen sogenannten "Algorithmus" erstellt hat, der, umgesetzt in eine dem Computer verständliche Form, das Komponieren oder Improvisieren von Musik nachahmen kann.

Damit ein Computer ein Musikstück spielen kann, braucht er ein Musikprogramm. Dieses Programm zeichnet sich durch lange Tabellen aus, in denen alle Informationen über Tonhöhe und Tondauer, wie sie in der Notenschrift festgehalten werden, enthalten sind. Es ist Aufgabe des Programmierers, diese Tabellen zu erstellen. Jedoch verfügt ein Computer nicht über 88 verschiedene Töne wie beim Klavier, er kann auch nur eine bestimmte Anzahl von Tönen zur gleichen Zeit spielen. Daher muß der Programmierer arrangieren, er wird sozusagen zum Dirigenten "in Zeitlupe", da er alle musikalischen Informationen minutiös in eine dem Computer verständliche Form bringt.

Während also der Programmierer dem Arrangeur oder Dirigenten gleichzusetzen ist, kann man ein Musikprogramm als den Interpreten eines Musikstücks auffassen. Dieses Musikprogramm "weiß" sozusagen alles, was nötig ist, um ein Musikstück zu spielen.

Doch was ist nun das Instrument in der Computermusik? Es wäre falsch, den Computer das Musikinstrument zu nennen. Es sind ja nur bestimmte Bauteile des Computers, die den hörbaren Ton erzeugen. Beim Commodore 64 ist das hauptsächlich der Sound-Chip 6581, aber auch der Mikroprozessor 6510. Doch ohne die Leitungen ("Busse") dieser Chips untereinander und verschiedener anderer Bauelemente könnte der Commodore 64 keinen Ton erzeugen. Aber Moment mal! Der Commodore 64 selbst erzeugt ja gar keinen Ton! Selbst wenn er gerade ein Musikprogramm abarbeitet, ist der Commodore 64 stumm, falls man keinen Fernseher oder auch eine Stereoanlage angeschlossen hat.

Das gehört aber auch zum Begriffsfeld Instrument dazu, so daß wir sogar das Kabel vom Computer zum Fernseher und den Lautsprecher darin als Teil dieses Musikinstruments bezeichnen müssen.

Und hier sind wir auch dem auf die Schliche gekommen, was uns Computermusik so "fremd" erscheinen läßt: Wir sehen eben keinen Musiker vor uns, der ein Instrument spielt; und was wir als Instrument vor Augen haben, ist uns auch nicht gerade als Musikinstrument vertraut. Der Interpret oder Musiker ist seltsamerweise etwas, was wir nicht sehen können: es ist das Musikprogramm, die "Software". Und das Musikinstrument ist eben der Sound-Chip und alles, was dazu gehört, um einen hörbaren Ton zu produzieren, also die "Hardware".

Aber auch vom Hören her wirkt Computermusik fremdartig auf uns. Es sind nicht nur die synthetischen Klänge, die nicht wie ein "normales" Musikinstrument klingen, es ist auch die Art, wie sie gespielt werden. So exakt und ausdauernd könnte kein Musiker z.B. eine Fuge von Bach spielen wie der Computer. Man kann ein Musikprogramm Tage lang laufen lassen, ohne das es "müde" wird oder gar Fehler macht. Es spielt im Gegenteil die Notenwerte auf die Millisekunde genau und immer gleich.

Und eben darin liegt die Gefahr der Computermusik. Programmierer lassen sich nämlich leicht dazu verleiten, in einem Computerprogramm nur die exakten Notenwerte zu verwenden. Das ist auch der Grund, warum so viele Stücke von Bach gerne in Musikprogramme umgesetzt werden. Denn Bachs Musik gewinnt dadurch auch einen gewissen Reiz, da sie selbst schon exakt und nach mathematischen Gesichtspunkten komponiert wurde. Daß diese Musik aber steril und unpersönlich klingt, liegt auf der Hand. Um Musikprogramme zu erstellen, die natürlicher und ausdrucksvoller klingen, muß man schon ein guter Programmierer und Musiker zugleich sein.

Es ist eben sehr schwierig, dem Computer uns vertraute Musik beizubringen. Aber darin liegt eigentlich nicht die Aufgabe der Computermusik. Es gibt auch Bereiche, wo uns die ungewohnten Eigenschaften der Computermusik zunutze kommen können und wir musikalisches Neuland betreten.

Durch die Benutzung der Computertechnologie konnten vollkommen neue Instrumente entwickelt werden, die Möglichkeiten erschließen, die kein konventionelles Musikinstrument bietet. In der Hauptsache sind dies Synthesizer und Sequenzer. In neuester Zeit sind aber auch weitere Instrumente entstanden, die auf der neuen Digitaltechnologie beruhen. Eben diese Technologie war die Basis für die Entwicklung des Synthesizer-Chips im Commodore 64. Instrumente, die diesem Chip sehr ähnlich sind, sind die Grundlage eines ganzen Musikbereiches, nämlich der elektronische Musik. Gruppen und Interpreten wie Jean-Michel Jarre, Klaus Schulze, Kraftwerk und Tangerine Dream könnten ohne diese Technologie heute keine Musik machen. Und auch die Klangvielfalt hat mit der Digitaltechnik vorher ungeahnte Ausmaße angenommen. Es können herkömmliche Instrumente ihrem Klang nach in etwa nachgeahmt werden, doch die Stärke der digitalen Instrumente liegt in ihren Variationsmöglichkeiten.

Der Commodore 64 setzt Maßstäbe durch seine musikalischen Eigenschaften. Vor 10 Jahren kam der erste "populäre" Synthesizer von Moog auf den Markt, der "Minimoog". Das war ein Instrument mit vielen Eigenschaften, die der Sound-Chip des Commodore 64 auch hat. Und obwohl dieser Moog-Synthesizer weit weniger konnte als der Sound-Chip 6581 im Commodore 64, kostete er Tausende von Mark. Heute leistet ein briefmarkengroßer Chip ein Vielfaches von dem, was ein Moog-Synthesizer damals konnte:

drei digitale Oszillatoren

acht Oktaven pro Oszillator

vier mischbare Wellenformen pro Oszillator:
Sägezahn, Dreieck, Rechteck/Puls und weißes Rauschen

für jeden Oszillator getrennt einstellbare Pulsbreite

drei Hüllkurvengeneratoren
alle Phasen des Tonablaufs getrennt einstellbar

ein programmierbarer Filter
(Hoch-, Tief- und Bandpass mischbar)
mit einstellbarer Grenzfrequenz und Filterresonanz
auf jeden Oszillator getrennt schaltbar

Ringmodulator und Synchronisationsmöglichkeiten
für Spezialeffekte

Wollen wir nun beginnen, diese Möglichkeiten kennenzulernen
und den Synthesizer im Commodore 64 zum Klingen zu bringen!

2. Die ersten Schritte am Commodore 64

Wir wollen nun einmal versuchen, den Sound-Chip in Ihrem Commodore 64 zum Spielen zu bringen. Wenn Sie Ihren Commodore 64 anschalten, werden Sie im Lautsprecher nichts hören. Auch wenn Sie ein paar Tasten am Computer drücken, wird der Sound-Chip stumm bleiben. Der Schlüssel zum Sound-Chip ist der Befehl "POKE". Deswegen ist es nicht weiter verwunderlich, daß der folgende Dreizeiler zum Großteil aus POKE's besteht:

```
0 A=54272:POKE A+24,15:POKE A+6,240:POKE A+4,33:P=0
1 POKE A+1,P AND 255:P=P+1:IF PEEK(198)=0 THEN 1
2 POKE A+1,0:POKE 198,0
```

Wenn Sie das Programm eingegeben haben, können Sie es mit RUN starten. Sie hören jetzt, welche Tonhöhen der Sound-Chip des Commodore 64 hervorbringen kann. Das Spektrum reicht von ganz tiefen bis zu ganz hohen Tönen. Wenn Sie das Programm beenden wollen, drücken Sie einfach irgendeine Taste. Wenn man die Anweisung POKE A+1,P AND 255 auf POKE A+1,255-(P AND 255) ändert, hört man das Tonspektrum anders herum.

Im folgenden Programm hört man beides zusammen. Ein Ton läuft von den ganz tiefen bis zu den ganz hohen Tonfrequenzen durch und ein zweiter Ton gerade anders herum:

```
0 A=54272:POKE A+24,15
1 POKE A+6,240:POKE A+4,33
2 POKE A+13,240:POKE A+11,33
3 P=0
4 POKE A+1, P AND 255
5 POKE A+8, 255-(P AND 255)
6 P=P+1:IF PEEK(198)=0 THEN 4
7 POKE A+1,0:POKE A+8,0:POKE 198,0
```

PEEK(198) gibt auch hier die Anzahl der gedrückten Tasten an. Das Programm stoppt, sobald man eine Taste drückt.

Wir haben gesehen, daß der Sound-Chip ein sehr breites Frequenzspektrum zur Verfügung stellt. Daraus kann man nun beliebige Frequenzen auswählen. Im folgenden Programm wird die Frequenz aus dem Code der zuletzt gedrückten Taste ermittelt. Sobald man die Space-Taste drückt, stoppt das Programm.

```
0 A=54272:POKE A+24,15
1 POKE A+6,240:POKE A+4,65
2 POKE A+1,PEEK(197):POKE A+3,PEEK(162)/32
3 POKE A+2,(PEEK(162) AND 7)*32
4 IF PEEK(197) <> 60 THEN 2
5 POKE A+1,0:POKE 198,0
```

Der Sound-Chip erzeugt nicht nur musikalische Töne, sondern auch ein Rauschen. Im folgenden Programm wird die Interrupt-Uhr ("Jiffy Clock") des Commodore 64 benutzt, um die Frequenz des Rauschens zu ermitteln. PEEK(162) ist ein Teil der Zeitangabe dieser Uhr.

```
0 A=54272:POKE A+24,15
1 POKE A+6,240:POKE A+4,129
2 POKE A+1,31 - (PEEK(162) AND 31):IF PEEK(198)=0 THEN 2
3 POKE A+1,0:POKE 198,0
```

Hier läuft die Frequenz des Rauschens von hellerem Rauschen zu dunklerem Rauschen. Wenn man in Zeile 2 das "31 -" wegläßt, läuft das ganze umgekehrt.

Neben der Frequenz hat man durch den Sound-Chip noch eine andere Möglichkeit, den Klang dieses Rauschens zu verändern: den Filter. Auch beim Filter kann man eine Frequenz einstellen, die dann angibt, wie hell oder dumpf das gefilterte Rauschen klingen soll. Im folgenden Programm wird die Frequenz des Filters erst von 0 auf ein Maximum erhöht und dann vom Maximum auf 0 erniedrigt. Das Ergebnis ist ein an- und abschwellendes Rauschen, das hier ungefähr an eine Lokomotive erinnert.

```
 0 A=54272:POKE A+24,31
 1 POKE A+6,240:POKE A+4,129
 2 POKE A+1,200:POKE A+23,1
 3 P=(PEEK(162)AND31)*16:IF P>255 THEN P=(P-(P-255)*2)
 4 POKE A+22,P
 5 IF PEEK(198)=0 THEN 3
 6 POKE A+1,0:POKE 198,0
```

Wir haben bisher zwei Möglichkeiten gehabt, um die Tonfrequenz zu ermitteln: einmal durch einen Zähler, zum anderen durch Tastendruck. Doch um Musikstücke mit dem Commodore 64 abzuspielen, muß man von vornherein festlegen, welche Tonfrequenzen gespielt werden sollen. Dazu kommt noch die Dauer, wie lange ein bestimmter Ton gespielt wird. Im folgenden Programm, das den Refrain vom Abba-Lied "Super Trooper" spielt, kann man sehen, wie das gemacht wird:

```
 0 A=54272:POKE A+24,15
 1 POKE A+6,240:POKE A+4,33
 2 READ L,H,D:IF L<0 THEN POKE A+24,0:END
 3 POKE A,L:POKE A+1,H:FOR I=0 TO D:NEXT:GOTO 2
 4 DATA 30,25,200,135,33,200,62,42,200,60,50,300,0,0,100
 5 DATA 60,50,300,0,0,100,193,44,300,0,0,100,193,44,300
 6 DATA 0,0,100,62,42,200,162,37,200,62,42,200,193,44,200
 7 DATA 62,42,300,0,0,100,162,37,300,0,0,100,193,44,300
 8 DATA 0,0,100,193,44,300,0,0,100,62,42,300,0,0,100
 9 DATA 62,42,300,0,0,100,162,37,1100,0,0,100,193,44,300
10 DATA 0,0,100,193,44,300,0,0,100,62,42,300,0,0,100
11 DATA 62,42,300,0,0,100,162,37,1100,0,0,100,162,37,200
12 DATA 135,33,200,162,37,200,62,42,200,162,37,300,0,0,100
13 DATA 135,33,300,0,0,100,135,33,800,-1,0,0
```

In diesem kurzen Basic-Programm kann man schon die grundlegende Struktur aller Musik-Programme erkennen: ein kurzes Programmstück (Zeile 0 bis 3) und ein vergleichsweise sehr langes Datenfeld (DATA's Zeile 4 bis 13). Im nun folgenden Kapitel wollen wir uns damit beschäftigen, wie man in Basic Musikstücke ein-, zwei- und schließlich dreistimmig programmiert.

```
0 REM  *** "LET IT BE"
1 REM
2 REM  *** ALLES AUF 0 SETZEN
3 REM
4 FOR  I = 54272 TO 54296
5 POKE I,0
6 NEXT I
7 REM
8 REM  *** "ANGESTELLTE" KENNZEICHNEN
9 REM
10 ERSTER = 54272
11 LAUTST = ERSTER + 24
12 AN     = ERSTER + 5
13 AUS    = ERSTER + 6
14 H1     = ERSTER
15 H2     = ERSTER + 1
16 DRUECK = ERSTER + 4
17 REM
18 REM *** ANFANGSWERTE SETZEN
19 REM
20 POKE LAUTST,15
21 POKE AN    ,23
23 POKE AUS   ,123
27 REM
28 REM *** EINE NOTE LESEN
29 REM
30 READ HOEHE,DAUER
40 IF   HOEHE=0 THEN END
47 REM
48 REM *** FREQUENZ AUSRECHNEN
49 REM
50 F2=HOEHE/256:F1=HOEHE-F2*256
60 POKE H2,F2  :POKE H1,F1
67 REM
68 REM *** "TASTE DRUECKEN"
69 REM
70 POKE DRUECK,33:FOR I=0 TO DAUER*100
```

```
75 NEXT
77 REM
78 REM *** "TASTE LOSLASSEN"
79 REM
80 POKE DRUECK,32:FOR I=0 TO DAUER*100
85 NEXT
87 REM
88 REM *** NAECHSTEN TON SPIELEN
89 REM
90 GOTO 30
97 REM
98 REM *** LISTE ALLER TONHOEHEN
99 REM             UND TONDAUERN
100 DATA 6430,1
110 DATA 6430,1
120 DATA 6430,3
130 DATA 6430,1
140 DATA 7217,2
150 DATA 5407,2
160 DATA 6430,1
170 DATA 6430,2
180 DATA 8583,3
190 DATA 9634,2
200 DATA 9634,1
210 DATA10814,2
220 DATA10814,3
230 DATA 9634,2
240 DATA 9634,2
250 DATA 8583,1
260 DATA 8583,5
270 DATA10814,1
280 DATA10814,2
290 DATA11457,3
300 DATA10820,2
310 DATA10814,2
320 DATA 9634,4
330 DATA10814,1
340 DATA 9634,1
350 DATA 9634,1
360 DATA 8583,11
370 DATA10814,1
380 DATA 9634,2
390 DATA 8534,5
400 DATA10814,1
410 DATA12860,2
420 DATA14435,5
430 DATA12860,1
440 DATA12860,2
450 DATA10814,3
460 DATA 6439,2
470 DATA 6439,1
480 DATA 6439,2
490 DATA10814,9
500 DATA10814,1
510 DATA10814,2
520 DATA11457,3
530 DATA10820,2
540 DATA10814,2
550 DATA 9634,4
560 DATA10814,1
570 DATA 9634,1
580 DATA 9634,1
590 DATA 8583,15
999 DATA    0,0

READY.
```

3. Musikprogrammierung in Basic

3.1 "Let it be" - Ein einstimmiges Musikprogramm

Haben Sie einmal probiert, das "Let it be"-Programm einzutippen? Falls Sie schon daran gewöhnt sind, in Basic zu programmieren, werden Sie kaum Schwierigkeiten damit haben. Sie werden sicher auch wissen, daß man alle Zeilen auslassen kann, die REM-Befehle enthalten. Dadurch erspart man sich Tipparbeit, das Programm wird aber etwas unübersichtlicher. Haben Sie das Programm in den Computer eingegeben, so können Sie es mit "RUN" starten. Sie wissen ja schon, daß man hinter jedem Basic-Befehl und jeder Basic-Programmzeile einmal die RETURN-Taste drücken muß.

Schauen Sie einmal nach, was für Basic-Befehle Sie in diesem Programm finden! Da gibt es FOR-NEXT-Schleifen, massig POKE-Befehle und jede Menge DATA's ab Zeile 100. Wenn Sie in Basic Musik programmieren wollen, werden Ihnen diese Befehle immer wieder begegnen. Wir wollen daher jetzt versuchen, die Funktionsweise dieser Befehle zu verstehen.

Das Bauteil, das in Ihrem Commodore 64 die Musik macht, ist der Sound-Chip 6581. Wir wollen uns die Funktionen dieses Sound-Chips in Zukunft dadurch veranschaulichen, daß wir uns diesen Chip als eine Art Büro vorstellen. Dieses Büro hat 29 Zimmer, und in jedem Zimmer sitzt eine bestimmte Anzahl von Angestellten. Wenn Sie Ihren Commodore 64 anschalten, so erhalten alle diese Angestellten die Anordnung, einfach nichts zu tun. Sie bleiben solange untätig, bis man Ihnen eine neue Anordnung gibt. Und diese Anordnung beginnt stets mit dem Wort "POKE". Danach kommt immer eine Zimmernummer. Diese Nummer ist eine fünfstellige Zahl von 54272 bis 54300. Nach der Zimmernummer kommt ein Komma als Zeichen dafür, daß die Zimmernummer zu Ende ist. Und nach dem Komma kommt die Anordnung, die an das Zimmer mit der zuvor angegebenen Nummer geht. Diese Anordnung besteht aus einer Zahl zwischen 0 und 255. Das liegt daran, daß ein Computer im Endeffekt ja nur Zahlen verstehen kann.

Auch Ihr Basic-Programm, das im Augenblick im Speicher Ihres Computers steht, steht dort nur als eine (allerdings ziemlich lange) Folgen von Zahlen, die zwischen 0 und 255 liegen. Am Anfang des "Let it be"-Programms (siehe Zeilen 4 bis 6) erhalten alle Zimmernummern die Anordnung "0". Das ist eben eine solche Zahl, die ja nur zwischen 0 und 255 liegen muß. Und "0" bedeutet für alle Angestellten, daß sie mit allem aufhören sollen, was sie bis jetzt machten und sich nur auf das nun folgende konzentrieren sollen.

Da man sich alle diese Zahlen nur schwer merken kann, hat man in Basic die Möglichkeit, stellvertretend für jede Zahl eine Folge aus Buchstaben zu schreiben, eine sogenannte "Variable". Man muß nur dem Computer einmal sagen: "Höre einmal, in Zukunft möchte ich statt der Zahl 54272 einfach "ERSTER" (Angestellter) sagen!" Das würde der Computer natürlich nicht verstehen, er reagiert dann mit einem "Syntax Error". Wenn man jedoch eingibt: "ERSTER = 54272", dann schluckt der Computer das. Er weiß von nun an, daß, wenn ERSTER eingegeben wird, eigentlich die Zahl 54272 gemeint ist (solange man den Computer nicht ausschaltet oder so böswillige Befehle wie CLR oder NEW eingibt. Aber auch, wenn man eine neue Basic-Zeile eingibt oder eine alte verändert, "vergißt" der Computer diese Anweisung). Diese Anweisung steht in dem "Let it be"-Programm in Zeile 10. In allen zukünftigen Beispielprogrammen, die in Basic geschrieben sind, werden wir die Anweisung "ERSTER = 54272" auch weiter verwenden.

"ERSTER" steht also für das erste Zimmer in dem Büro, als das wir uns den Sound-Chip vorstellen. Die Zeilen 11-16 dienen nun dazu, um die anderen "Zimmer" zu bezeichnen, die im Programm verwendet werden. In den Zeilen 20-23 bekommen bestimmte Angestellte in diesen Zimmern Anweisungen, die während des gesamten Programms nicht verändert werden. Das übrige Programm besteht aus einer einzigen Schleife, in der jeweils eine Note aus der Liste am Ende des Programms gelesen wird und dann eine bestimmte Zeit gewartet wird.

Bevor wir auf weitere Details eingehen, sollte uns die Struktur eines Musikprogramms klar sein. Diese Struktur wird Ihnen in allen Musikprogrammen wiederbegegnen, egal, ob dies Programmé aus diesem Buch sind oder andere.

Wir haben uns den Sound-Chip als eine Art Büro vorgestellt, das viele Zimmer hat. Ein Musikprogramm kann man sich als eine Art Bürobote vorstellen, der wie Superman durch die Zimmer eilt und jedem Angestellten ein Blatt mit den neuesten Anordnungen gibt. Sie als Programmierer sind der Chef des Büros. Sie geben dem Büroboten die Liste der Anordnungen. Das ist der Text, den Sie als Basic-Programm in den Commodore 64 eintippen.

Wenn Sie in Basic programmieren, können Sie sich mit "LIST" jederzeit diese Liste ausgeben lassen. Ist diese Liste fertig, so können Sie dem Büroboten sagen, er soll jetzt losrennen ("RUN"). Sie können aber jederzeit den Büroboten durch Drücken der RUN/STOP-Taste anhalten lassen. Durch den Basic-Befehl CONT können Sie den Boten "weiterrennen" lassen.

Wie sieht nun eine solche Liste aus? Betrachten wir als Beispiel das "Let it be"-Programm. Was einem sofort ins Auge fällt, ist die krasse Zweiteilung Programm - Daten. Diese Zweiteilung, durch die sich die meisten Programme auszeichnen, ist bei einem Musikprogramm am deutlichsten zu erkennen.

Im Beispielprogramm geht der eigentliche Programmteil von Zeile 0 bis Zeile 90. Von Zeile 100 bis Programmende befindet sich der Datenteil. In diesem Fall sind Programmteil und Datenteil in etwa gleich lang. So etwas findet man aber nur bei einfacheren Musikprogrammen. Bei komplexeren Musikprogrammen, wie wir sie später kennenlernen werden, ist der Datenteil um ein Vielfaches länger. Woran liegt das eigentlich?

Ein Musikprogramm besteht zum Großteil aus den musikalischen Informationen des Musikstücks, das das Programm spielen soll. Das Programm selbst ist nur kurz, da es im Grund immer nur einen einzelnen Ton spielt. Sämtliche musikalischen Informationen werden in einem Musikprogramm in sogenannten Datenfeldern gespeichert.

In Basic gibt es hierfür den Befehl DATA, der es erlaubt, solche Daten an beliebiger Stelle im Programm unterzubringen. Meistens findet man aber eine große Anzahl von DATA-Befehlen in einem großen Block zusammen. Im Beispielprogramm wurde für jede Note eine eigene Zeile verwendet. Es wäre auch möglich, die Daten für mehrere Noten in eine Zeile zu schreiben:

```
100 DATA 6430,1,6430,1,6430,3,6430,1,..... usw.
```

Doch dann leidet die Übersichtlichkeit. Was sind denn nun das für Daten? Man findet immer ein Paar von zwei Zahlen. Die erste Zahl, die vier- bis fünfstellig ist, steht für die Tonhöhe (Frequenz). Eine Tabelle für die gebräuchlichen Tonhöhen finden Sie in Kapitel 4.

Die zweite Zahl steht für die Tondauer. Man findet hier Werte von 1 bis 15. "1" steht hier für eine achtel Note, ein ziemlich kurzer Notenwert. Sämtliche anderen Notenwerte erhält man durch einfache Addtion: "2" steht für eine viertel Note, "4" für eine halbe Note usw.

Es ist ohne weiteres möglich, nur durch Ändern der DATA-Anweisungen ein völlig anderes Stück spielen zu lassen. Beispiel hierfür ist das Programm "Hey Jude", dessen Listing Sie auf den nun folgenden Seiten finden.

```
0 REM  *** "HEY JUDE"
1 REM
2 REM  *** ALLES AUF 0 SETZEN
3 REM
4 FOR  I = 54272 TO 54296
5 POKE I,0
6 NEXT I
7 REM
8 REM  *** "ANGESTELLTE" KENNZEICHNEN
9 REM
10 ERSTER = 54272
11 LAUTST = ERSTER + 24
12 AN     = ERSTER + 5
13 AUS    = ERSTER + 6
14 H1     = ERSTER
15 H2     = ERSTER + 1
16 DRUECK = ERSTER + 4
17 REM
18 REM *** ANFANGSWERTE SETZEN
19 REM
20 POKE LAUTST,15
21 POKE AN    ,122
23 POKE AUS   ,120
27 REM
28 REM *** EINE NOTE LESEN
29 REM
30 READ HOEHE,DAUER
40 IF   HOEHE=0 THEN END
47 REM
48 REM *** FREQUENZ AUSRECHNEN
49 REM
50 F2=HOEHE/256:F1=HOEHE-F2*256
60 POKE H2,F2  :POKE H1,F1
67 REM
68 REM *** "TASTE DRUECKEN"
```

```
 69 REM
 70 POKE DRUECK,17:FOR I=0 TO DAUER*50
 75 NEXT
 77 REM
 78 REM *** "TASTE LOSLASSEN"
 79 REM
 80 POKE DRUECK,16:FOR I=0 TO DAUER*50
 85 NEXT
 87 REM
 88 REM *** NAECHSTEN TON SPIELEN
 89 REM
 90 GOTO 30
 97 REM
 98 REM *** LISTE ALLER TONHOEHEN
 99 REM             UND TONDAUERN
100 DATA 8583,4
110 DATA 7217,10
120 DATA 7217,2
130 DATA 8583,2
140 DATA 9634,2
150 DATA 6430,12
160 DATA 6430,2
170 DATA 7217,2
180 DATA 7647,4
190 DATA11457,6
200 DATA11457,2
210 DATA10814,2
220 DATA 8583,2
230 DATA 9634,2
240 DATA 8583,1
250 DATA 7647,1
260 DATA 7217,10
270 DATA 8583,2
280 DATA 9634,2
290 DATA 9634,4
300 DATA 9634,2
310 DATA12860,1
320 DATA11457,2
330 DATA10814,2
```

```
340 DATA11457,1
350 DATA 9634,2
360 DATA 8583,8
370 DATA 5728,2
380 DATA 6430,2
390 DATA 7217,2
400 DATA 9634,4
410 DATA 8583,4
420 DATA 8583,2
430 DATA 7647,2
440 DATA 7217,2
450 DATA 5407,2
460 DATA 5728,4
470 DATA 5728,17
480 DATA 5728,1
490 DATA11457,1
500 DATA10207,2
510 DATA 9634,3
520 DATA 8583,2
530 DATA 8583,2
540 DATA 7647,1
550 DATA 9634,5
560 DATA11457,2
570 DATA 9634,6
580 DATA11457,2
590 DATA 7647,6
600 DATA11457,2
610 DATA 9634,4
620 DATA 8583,2
630 DATA 7647,2
640 DATA 8583,5
650 DATA 9634,3
660 DATA 8583,4
670 DATA 7647,4
680 DATA 7217,2
690 DATA 6430,2
700 DATA 5728,17
710 DATA 5728,1
720 DATA 7647,1
730 DATA 9634,2
740 DATA11457,3
750 DATA 9634,2
760 DATA 9634,2
770 DATA 8583,1
780 DATA 9634,5
790 DATA11457,2
800 DATA 9634,6
810 DATA11457,2
820 DATA 7647,6
830 DATA11457,2
840 DATA 9634,4
850 DATA 8583,2
860 DATA 7647,2
870 DATA 8583,6
880 DATA 9634,2
890 DATA 8583,2
900 DATA 7647,5
910 DATA 7217,4
920 DATA 6430,1
930 DATA 5728,6
940 DATA 5728,2
950 DATA 8583,2
960 DATA 9634,2
970 DATA10207,2
980 DATA 9634,2
990 DATA10207,4
1000 DATA10814,2
1010 DATA11457,2
1020 DATA12860,4
1030 DATA12860,8
1040 DATA    0,0

READY.
```

3.2 "Hey Jude" - Editieren von Basic-Musikprogrammen

Wenn Sie das Programm "Let it be" noch im Speicher haben, ist es sehr einfach, das Programm "Hey Jude" einzugeben. Die Programme sind nämlich von der Struktur her gleich, es wurden nur die Programmdaten verändert.

Wir wollen nun Schritt für Schritt die Unterschiede durchgehen. Die Zeilen von 0 bis 90 sind in beiden Programmen fast identisch, es müssen nur ein paar Zahlenwerte verändert werden.

- Geben Sie "LIST 21" ein.
- Ändern Sie die Zahl "23" auf "122".
- Drücken Sie RETURN.

- Geben Sie "LIST 23" ein.
- Ändern Sie die Zahl "123" auf "120".
- Drücken Sie RETURN.

- Geben Sie "LIST 70" ein.
- Ändern Sie die Zahl "33" auf "17".
- Ändern Sie die Zahl "100" auf "50".
- Drücken Sie RETURN.

- Geben Sie "LIST 80" ein.
- Ändern Sie die Zahl "32" auf "16".
- Ändern Sie die Zahl "100" auf "50".
- Drücken Sie RETURN.

Geben Sie nun einmal "RUN" ein, ohne den Rest des Programms verändert zu haben. Es wird die "Let it be"-Melodie gespielt, aber mit einem anderen Klang und doppelt so schnell! Wir hören einen dumpferen, flötenähnlichen Klang. Wenn die Werte "17" und "16" zurück auf "33" und "32" setzen, hören wir wieder den ursprünglichen Klang. An dieser Stelle des Programms dürfen nur diese Werte erscheinen. Geben Sie nun auch die DATA's von "Hey Jude" ein und probieren Sie das Programm mit RUN aus!

Es gibt noch andere Möglichkeiten, dieses Programm zu verändern (der Computerfachmann sagt hierzu "editieren"). Wenn Sie den Wert hinter der Anweisung "FOR I=0 TO DAUER*..." verändern, verändert sich das Tempo des Stücks. Je kleiner dieser Wert ist, desto schneller wird das Stück gespielt. Die Anweisungen "FOR" und "NEXT" (Zeilennummern 70/75 und 80/85) nennt man hier auch "Verzögerungsschleife". Der Computer tut hier nichts anderes, als von 0 bis zu einer bestimmten Zahl hochzuzählen. Je kleiner diese Zahl ist, desto schneller ist der Computer mit dem Zählen fertig und desto schneller läuft die ganze Sache. Wird die Zahl halbiert, läuft das Programm doppelt so schnell; wird die Zahl verdoppelt, so halbiert sich die Ablaufgeschwindigkeit des Programms.

Auch die Zeilen 20, 21 und 23 brauchen vor uns nicht sicher zu sein! In der Zeile 20 werden die Angestellten des Büros angesprochen, die die Lautstärke der Melodie kontrollieren. Man kann die Lautstärke von "ganz laut" bis "ganz leise" einstellen. Für "ganz laut" steht der Zahlenwert 15, für ganz leise der Zahlenwert 1. Dazwischen kann man beliebige Zwischenwerte einstellen. Geben Sie dazu einfach ein "20 POKE LAUTST,X", wobei X ein Wert zwischen 0 und 15 ist. Bei X=0 hört man jedoch gar nichts mehr. Man kann sich das auch als eine Art Lautstärkeknopf vorstellen, wobei 0 bedeutet, der Lautstärkeknopf steht auf 0. Mit den Werten 1, 2, 3... dreht man die Lautstärke immer mehr auf, um mit 15 den Maximalwert zu erreichen.

In den Zeilen 21 und 23 (Sie können diesen Zeilen auch die Nummern 21 und 22 oder auch 23 und 25 geben, ohne am Programm Grundlegendes zu verändern) wird der sogenannte ADSR-Wert gesetzt. An dieser Stelle soll noch nicht auf die (ziemlich komplizierte) Bedeutung dieses Wertes eingegangen werden. Trotzdem können Sie ja einmal versuchen, hier beliebige Werte zwischen 0 und 255 einzugeben und selbst etwas darüber herauszufinden! Eine genaue Beschreibung der Bedeutung der ADSR-Werte können Sie im vierten Kapitel nachlesen.

```
0 REM  *** "MULL OF KINTYRE"
1 REM
2 REM  *** ALLES AUF 0 SETZEN
3 REM
4 FOR  I = 54272 TO 54296
5 POKE I,0
6 NEXT I
7 REM
8 REM  *** "ANGESTELLTE" KENNZEICHNEN
9 REM
10 ERSTER = 54272
11 LAUTST = ERSTER + 24
12 AN     = ERSTER + 5
13 AUS    = ERSTER + 6
14 H1     = ERSTER
15 H2     = ERSTER + 1
16 DRUECK = ERSTER + 4
17 REM
18 REM *** ANFANGSWERTE SETZEN
19 REM
20 POKE LAUTST,15
21 A=0
22 D=6
23 S=8
24 R=9
25 POKE AN  , A*16 + D
26 POKE AUS , S*16 + R
27 REM
28 REM *** TONFREQUENZEN LESEN
29 REM
30 DIM  FRQ$(59),FL(59),FH(59)
40 FOR  I = 0 TO 59
41 READ FRQ$(I),FRQ
42 FH  (I)= INT (FRQ/256)
```

```
43 FL  (I)= FRQ-FH(I)*256
44 NEXT I
47 REM
48 REM *** NOTENHOEHEN & DAUERN LESEN
49 REM
50 DIM  N1$(100),D1(100)
51 I=0
52 READ N1$(I),D1(I)
53 IF D1(I)>0 THEN I=I+1:GOTO 52
57 REM
58 REM *** NOTENSTRINGS UMRECHNEN
59 REM
60 PRINTCHR$(147)CHR$(13)"NOTENUMRECHNUNG"CHR$(13)
62 DIM L1(100),H1(100)
63 I=0
64 J=59
65 IF FRQ$(J)=N1$(I) THEN L1(I)=FL(J):H1(I)=FH(J):PRINTN1$(I),:GOTO 68
67 J=J-1:GOTO 65
68 I=I+1:IF NOT N1$(I)="" THEN 64
70 REM
71 REM *** STUECK SPIELEN
72 REM
73 PRINT CHR$(147)
74 PRINT "GESCHWINDIGKEIT (1-9)?"
75 POKE 198,0:WAIT 198,1:GET G$
76 IF VAL(G$) < 1 THEN 75
77 SPEED = VAL (G$):FLAG = 0
80 S1 = 0
81 C1 = 0
82 POKE H1,L1(S1):POKE H2,H1(S1):POKE DRUECK,33
84 C1 = C1+1:IF C1>(D1(S1)*SPEED/2) THEN POKE DRUECK,32
85 IF C1<(D1(S1)*SPEED) THEN 90
86 C1 = 0:S1 = S1+1
87 POKE DRUECK,33
88 POKE H1,L1(S1):POKE H2,H1(S1)
90 IF S1=69 THEN IF C1>(D1(S1)*SPEED-2) THEN FLAG=1:GOTO 80
91 IF S1=26 THEN IF C1>(D1(S1)*SPEED-2) THEN IF FLAG=1 THEN 93
92 GOTO 84
93 PRINT:PRINT "NOCHMAL? (J)"
```

```
94 POKE198,0:WAIT198,1:GET A$
95 IF A$="J" THEN 73
96 END
97 REM
98 REM *** DATENFELDER
99 REM
100 REM  FREQUENZEN
101 DATA C-2 ,1072
102 DATA C#-2,1136
103 DATA D-2 ,1206
104 DATA D#-2,1275
105 DATA E-2 ,1351
106 DATA F-2 ,1432
107 DATA F#-2,1517
108 DATA G-2 ,1607
109 DATA G#-2,1703
110 DATA A-2 ,1804
111 DATA A#-2,1911
112 DATA B-2 ,2025
121 DATA C-3 ,2145
122 DATA C#-3,2273
123 DATA D-3 ,2408
124 DATA D#-3,2551
125 DATA E-3 ,2703
126 DATA F-3 ,2864
127 DATA F#-3,3034
128 DATA G-3 ,3215
129 DATA G#-3,3406
130 DATA A-3 ,3608
131 DATA A#-3,3823
132 DATA B-3 ,4050
141 DATA C-4 ,4291
142 DATA C#-4,4547
143 DATA D-4 ,4817
144 DATA D#-4,5103
145 DATA E-4 ,5407
146 DATA F-4 ,5728
147 DATA F#-4,6069
148 DATA G-4 ,6430
149 DATA G#-4,6812
150 DATA A-4 ,7217
151 DATA A#-4,7647
152 DATA B-4 ,8101
161 DATA C-5 ,8583
162 DATA C#-5,9094
163 DATA D-5 ,9634
164 DATA D#-5,10207
165 DATA E-5 ,10814
166 DATA F-5 ,11457
167 DATA F#-5,12139
168 DATA G-5 ,12860
169 DATA G#-5,13625
170 DATA A-5 ,14435
171 DATA A#-5,15294
172 DATA B-5 ,16203
181 DATA C-6 ,17167
182 DATA C#-6,18188
183 DATA D-6 ,19269
184 DATA D#-6,20415
185 DATA E-6 ,21629
186 DATA F-6 ,22915
187 DATA F#-6,24278
188 DATA G-6 ,25721
189 DATA G#-6,27251
190 DATA A-6 ,28871
191 DATA A#-6,30588
192 DATA B-6 ,32407
200 REM  NOTENWERTE UND NOTENDAUERN
201 DATA A-4 ,3,B-4 ,1:REM REFRAIN
202 DATA D-5 ,2,F#-5,4
203 DATA F#-5,2,F#-5,2
204 DATA E-5 ,2,D-5 ,2
205 DATA E-5 ,1,D-5 ,3
206 DATA B-4 ,2,A-4 ,3
207 DATA B-4 ,1,D-5 ,2
208 DATA F#-5,4,F#-5,2
209 DATA F#-5,2,E-5 ,2
210 DATA D-5 ,2,E-5 ,1
```

```
211 DATA D-5 ,3,B-4 ,2
212 DATA A-4 ,3,B-4 ,1
213 DATA D-5 ,1,E-5 ,1
214 DATA D-5 ,9
220 DATA A-4 ,2,D-5 ,2:REM STROPHE
221 DATA B-4 ,2,A-4 ,1
222 DATA F#-4,3,A-4 ,2
223 DATA D-5 ,2,F#-5,2
224 DATA E-5 ,2,D-5 ,6
225 DATA B-4 ,2,D-5 ,2
226 DATA C#-5,2,B-4 ,1
227 DATA A-4 ,3,G-4 ,2
228 DATA A-4 ,2,D-5 ,2
229 DATA B-4 ,2,A-4 ,6
230 DATA A-4 ,2,D-5 ,2
231 DATA B-4 ,2,A-4 ,1
232 DATA F#-4,3,A-4 ,2
233 DATA D-5 ,2,F#-5,2
234 DATA E-5 ,2,D-5 ,2
235 DATA E-5 ,2,F#-5,2
236 DATA G-5 ,3,F#-5,1
237 DATA E-5 ,2,D-5 ,1
238 DATA B-4 ,3,B-4 ,2
239 DATA A-4 ,3,B-4 ,1
240 DATA D-5 ,1,E-5 ,1
241 DATA D-5 ,9
299 DATA ,-1

READY.
```

3.3 "Mull of Kintyre" - Vereinfachte Noteneingabe

Sie haben nun durch das Eintippen der beiden Beispielprogramme "Let it be" und "Hey Jude" einige Erfahrungen mit dem Eingeben von Basic-Musikprogrammen gemacht. Sicher werden Sie auch bemerkt haben, daß es besonders mühselig ist, die vielen DATA's einzutippen. Nun, das kann niemandem erspart bleiben, der ein Musikprogramm schreiben will, egal in welcher Programmiersprache: Irgendwie müssen die musikalischen Informationen ja untergebracht werden. Aber wie man die Daten über Tonhöhe und Tondauer unterbringt, ist Sache des Programmierers. Im Beispielprogramm "Mull of Kintyre" lernen wir eine neue, bequemere Art kennen.

Der Anfang dieses Programms wird Ihnen bekannt vorkommen, aber spätestens ab Zeile 30 sieht dann alles ganz anders aus als bisher. Auch wenn Sie sich die DATA's ab Zeile 100 anschauen, werden Sie Unterschiede zu den bisherigen Beispielprogrammen feststellen.

Die Grundidee hinter der neuen Datenstruktur und den zugehörigen neuen Programmteilen ist, daß es eigentlich doch immer ziemlich mühselig war, bei den DATA's vier- bis fünfstellige Zahlen für die Tonhöhe einzugeben. Es wäre eigentlich viel einfacher, wenn man die Tonhöhen direkt (also wie im Beispiel A-4, B-4, D-5 usw.) eingeben könnte. Und genau dies ist beim Beispielprogramm "Mull of Kintyre" möglich.

Doch kaum hat man ein Problem beseitigt, taucht ein neues auf: Diese Tonbezeichnungen, die dem Musiker sicherlich näherkommen, kann der Computer nicht mehr direkt verstehen. Die Tonbezeichnungen müssen vorher umgerechnet werden, und das kostet Zeit. Jedoch rechnet der Computer wesentlich schneller als wir, so daß es im Endeffekt doch bequemer ist, auf die Umrechnung zu warten, als die gesamten Frequenzen mühevoll als vier- und fünfstellige Zahlen einzugeben.

Diese Umrechenroutine wird in allen folgenden Beispielprogrammen weiter verwendet (nur noch an einer Stelle wird sie leicht modifiziert), so daß Sie sie nicht immer neu einzutippen brauchen. Wie sie funktioniert, wollen wir uns nun im einzelnen anschauen.

Die Tonfrequenzen, die wir bisher bei jedem einzelnen Ton der Melodie dazuschreiben mußten, befinden sich im Beispielprogramm "Mull of Kintyre" in einem Datenfeld von Zeile 100 bis Zeile 192. Da das Programm jederzeit über einen Index auf jede Frequenz zugreifen muß, werden von Zeile 30 bis Zeile 44 alle Tonfrequenzen auf einmal in ein Array geschrieben. Wie diese Routine funktioniert, werden Sie als erfahrener Programmierer leicht durchschauen:

Zuerst werden drei Arrays dimensionert, eins für die Strings (FRQ$), die die Töne symbolisieren, und zwei für die Low-und High-Werte der Frequenzen (FL bzw. FH). Da die Frequenzen als 16-Bit Werte in den DATA's erscheinen, werden sie innerhalb der READ-Schleife in Low- und High-Byte aufgespaltet, um diesen zeitaufwendigen Vorgang während des Spielens der Melodie zu umgehen.

Das mag sich alles fürchterlich kompliziert anhören, ist jedoch für reine Musikprogrammierer nicht weiter wichtig. Um die Notenwerte einzugeben, braucht man ja nicht wissen, wie sie nachher verarbeitet werden, ähnlich wie bei einem Klavierspieler, der auch nichts von der Mechanik eines Flügels zu wissen braucht, um auf ihm zu spielen.

Schauen wir uns nun weiter die neuen Programmteile des Programms an! Von Zeile 50 bis 53 finden wir ein Programmstück, das die Notenstrings und Notendauern in ein weiteres Array einliest. Von Zeile 60 bis 68 befindet sich dann die eigentliche Umrechenroutine. Hier wird jeder Notenstring mit den in der Tabelle von Zeile 100 bis 192 befindlichen Strings verglichen.

Wird der String gefunden, handelt es sich also um eine "erlaubte" (d.h. vorgesehene) Note, so werden in die beiden Frequenz-Arrays L1 und H1 die entsprechenden Tonfrequenzen geschrieben. Haben Sie beim Eingeben einen Fehler gemacht, so bleibt hier der Anfangswert der Frequenz (= 0) unverändert, so daß Sie nachher beim Abspielen der Melodie den entsprechenden Ton nicht hören werden. Wenn die Melodie also nicht stimmt, müssen Sie die DATA's kontrollieren.

Von Zeile 73 bis 77 finden wir eine Benutzerabfrage. Hier wird derjenige, der das Programm ausprobiert, gefragt, mit welcher Geschwindigkeit die eingegebene Melodie abgespielt werden soll. Es sind Werte von 1 bis 9 zulässig, wobei 1 ganz schnell und 9 ganz langsam bedeutet. Es können aber auch beliebige Zwischenwerte eingegeben werden, wobei Werte um 5 eine relativ "normale" Geschwindigkeit bewirken.

Von Zeile 80 bis 92 befindet sich die Spielroutine, der Programmteil also, der dafür verantwortlich ist, daß die eingegebene Melodie korrekt abgespielt wird. Wir haben es hier mit einer völlig anderen Routine als bisher zu tun. Das liegt daran, daß die Notenwerte nicht mehr wie bisher mit READ aus dem Datenfeld herausgelesen werden, sondern daß sie sich in drei Arrays befinden. Die Routine ist außerdem so konzipiert, daß sie später auf mehrstimmige Stücke ausgedehnt werden kann.

Hauptprinzip bei dieser Routine ist ein "counter"-System (counter = engl. Zähler). Hierbei wird eine Variable (C1) von 0 bis zu einem bestimmten Wert hochgezählt, der von der Notendauer abhängig ist. Genaueres hierzu finden Sie am Anfang von Kapitel 5, wo auf das counter-Prinzip eingegangen wird.

Die Programme, die wir bisher hatten, liefen nur linear ab, d.h. was einmal gespielt wurde, konnte nicht wieder abgerufen werden. Dies liegt in der READ-DATA-Konzeption von Commodore Basic begründet, die nur eine sequentielle (d.h. ein Wert nach dem anderen) Abarbeitung der gespeicherten Werte zuläßt.

Dadurch, daß wir jetzt mit Arrays arbeiten, können wir bereits gespielte Teile eines Musikstücks wiederholen. Dadurch ist es möglich, den Aufbau Refrain-Strophe-Refrain zu realisieren. Man braucht dazu aber noch ein "Flag" (engl. Flagge), das anzeigt, wie oft ein gewünschter Abschnitt aus einem Musikstück wiederholt wird.

Im Beispiel haben wir den einfachsten Fall einer Flag-Abfrage vor uns: Das Programm soll einmal Refrain und Strophe durchspielen. Danach sind die DATA's eigentlich erschöpft, aber da wir ein Array benutzen, können wir den Index auf 0 zurücksetzen und das Stück nochmal durchlaufen.

Würden wir hierbei kein Flag setzen, so würde das Stück endlos weiterlaufen, da nie eine Abbruchbedingung gegeben wäre. Deswegen erscheint in Zeile 90 die Anweisung "FLAG=1", bevor das Stück mit "GOTO 80" neu gestartet wird. In Zeile 91 erscheint dann die Abbruchbedingung: Falls das Flag auf 1 sitzt und der Refrain somit noch einmal durchlaufen wurde, wird das Stück beendet.

In Zeile 93 bis 95 finden wir eine zweite Benutzerabfrage, die es dem Benutzer erlaubt, sich das Stück noch einmal anzuhören. Diese Abfrage ist nötig, denn sonst mußte man ja immer RUN eingeben, um das Stück noch einmal zu hören, und bei unserem Programm müßte man dann immer wieder auf die Notenumrechnung warten.

Im nun folgenden Programm "Yesterday" haben wir ein zweistimmiges Musikstück vor uns, das auf der Struktur von "Mull of Kintyre" aufbaut.

```
0 REM  *** "YESTERDAY"
1 REM
2 REM  *** ALLES AUF 0 SETZEN
3 REM
4 FOR  I = 54272 TO 54296
5 POKE I,0
6 NEXT I
7 REM
8 REM  *** "ANGESTELLTE" KENNZEICHNEN
9 REM
10 ERSTER = 54272
11 LAUTST = ERSTER + 24
12 AN     = ERSTER + 5
13 AUS    = ERSTER + 6
14 H1     = ERSTER
15 H2     = ERSTER + 1
16 DRUECK = ERSTER + 4
17 REM
18 REM *** ANFANGSWERTE SETZEN
19 REM
20 POKE LAUTST,15:POKE ERSTER+3,3
21 POKE AN    , 6*16 +  8
22 POKE AUS   , 7*16 + 10
23 POKE AN +7 , 2*16 + 11
24 POKE AUS+7 ,11*16 + 11
25 W1 = 64 :REM RECHTECK
26 W2 = 32 :REM SAEGEZAHN
27 REM
28 REM *** TONFREQUENZEN LESEN
29 REM
30 DIM  FRQ$(59),FL(59),FH(59)
40 FOR  I = 0 TO 59
41 READ FRQ$(I),FRQ
42 FH  (I)= INT (FRQ/256)
43 FL  (I)= FRQ-FH(I)*256
44 NEXT I
```

```
47 REM
48 REM *** NOTENHOEHEN & DAUERN LESEN
49 REM
50 DIM  N1$(100),D1(100),N2$(100),D2(100)
51 I=0
52 READ N1$(I),D1(I):IF D1(I)>0 THEN I=I+1:GOTO 52
53 I=0
54 READ N2$(I),D2(I):IF D2(I)>0 THEN I=I+1:GOTO 54
57 REM
58 REM *** NOTENSTRINGS UMRECHNEN
59 REM
60 PRINTCHR$(147)CHR$(13),"  NOTENUMRECHNUNG"CHR$(13)
61 PRINT,"1.STIMME","2.STIMME"CHR$(13)
62 DIM L1(100),H1(100),L2(100),H2(100)
63 I=0
64 FOR J=0 TO 59
65 IF FRQ$(J)=N1$(I) THEN L1(I)=FL(J):H1(I)=FH(J):PRINT,N1$(I)
66 IF FRQ$(J)=N2$(I) THEN L2(I)=FL(J):H2(I)=FH(J):PRINT,,N2$(I)
67 NEXT
68 I=I+1:IF NOT (N1$(I)="" AND N2$(I)="") THEN 64
70 REM
71 REM *** STUECK SPIELEN
72 REM
73 PRINT CHR$(147)
74 PRINT "GESCHWINDIGKEIT (1-5)?"
75 POKE 198,0:WAIT 198,1:GET G$
76 IF VAL(G$) < 1 OR VAL(G$) > 5 THEN 75
77 SPEED = VAL (G$):FLAG = 0:GOTO 80
78 FOR X=1 TO SPEED*80:NEXT :GOTO 81
79 FOR X=1 TO SPEED*80:NEXT
80 S1 = 0 :S2 = 0
81 C1 = 0 :C2 = 0
82 POKE H1,L1(S1):POKE H2,H1(S1):POKE DRUECK,W1 OR 1
83 POKE H1+7,L2(S2):POKE H2+7,H2(S2):POKE DRUECK+7,W2 OR 1
84 C1 = C1+1:IF C1>(D1(S1)*SPEED/2) THEN POKE DRUECK,W1
85 IF C1<(D1(S1)*SPEED) THEN 87
86 C1 = 0:S1 = S1+1:POKE DRUECK,W1 OR 1:POKE H1,L1(S1):POKE H2,H1(S1)
87 C2 = C2+1:IF C2>(D2(S2)*SPEED/2) THEN POKE DRUECK+7,W2
88 IF C2<(D2(S2)*SPEED) THEN 90
```

```
89 C2 = 0:S2 = S2+1:POKE DRUECK+7,W2 OR 1:POKE H1+7,L2(S2):POKE H2+7,H2(S2)
90 IF S1=28 THEN IF C1>(D1(S1)*SPEED/2) THEN IF FLAG=0 THEN FLAG=1:GOTO 79
91 IF S1=55 THEN IF C1>(D1(S1)*SPEED/2) OR SPEED=1 THENIFFLAG=1THENFLAG=2:GOTO80
92 IF S1=28 THEN IFC1>(D1(S1)*SPEED/2)THENIFFLAG=2THENSP=SP+1:S1=56:S2=50:GOTO78
93 IF S1=62 THEN IF C1>(D1(S1)*SPEED/2) THEN 95
94 GOTO 84
95 PRINT "NOCHMAL? (J)":POKE198,0:WAIT198,1:GET A$:IF A$="J" THEN GOTO 73
96 END
97 REM
98 REM *** DATENFELDER
99 REM
100 REM  FREQUENZEN
101 DATA C-2 ,1072
102 DATA C#-2,1136
103 DATA D-2 ,1206
104 DATA D#-2,1275
105 DATA E-2 ,1351
106 DATA F-2 ,1432
107 DATA F#-2,1517
108 DATA G-2 ,1607
109 DATA G#-2,1703
110 DATA A-2 ,1804
111 DATA A#-2,1911
112 DATA B-2 ,2025
121 DATA C-3 ,2145
122 DATA C#-3,2273
123 DATA D-3 ,2408
124 DATA D#-3,2551
125 DATA E-3 ,2703
126 DATA F-3 ,2864
127 DATA F#-3,3034
128 DATA G-3 ,3215
129 DATA G#-3,3406
130 DATA A-3 ,3608
131 DATA A#-3,3823
132 DATA B-3 ,4050
141 DATA C-4 ,4291
142 DATA C#-4,4547
143 DATA D-4 ,4817
144 DATA D#-4,5103
145 DATA E-4 ,5407
146 DATA F-4 ,5728
147 DATA F#-4,6069
148 DATA G-4 ,6430
149 DATA G#-4,6812
150 DATA A-4 ,7217
151 DATA A#-4,7647
152 DATA B-4 ,8101
161 DATA C-5 ,8583
162 DATA C#-5,9094
163 DATA D-5 ,9634
164 DATA D#-5,10207
165 DATA E-5 ,10814
166 DATA F-5 ,11457
167 DATA F#-5,12139
168 DATA G-5 ,12860
169 DATA G#-5,13625
170 DATA A-5 ,14435
171 DATA A#-5,15294
172 DATA B-5 ,16203
181 DATA C-6 ,17167
182 DATA C#-6,18188
183 DATA D-6 ,19269
184 DATA D#-6,20415
185 DATA E-6 ,21629
186 DATA F-6 ,22915
```

```
187 DATA F#-6,24278
188 DATA G-6 ,25721
189 DATA G#-6,27251
190 DATA A-6 ,28871
191 DATA A#-6,30588
192 DATA B-6 ,32407
199 REM  NOTENWERTE UND NOTENDAUERN
200 REM  1. STIMME
201 DATA G-4 ,2,F-4 ,1
202 DATA F-4 ,7,A-4 ,1
203 DATA B-4 ,1,C#-5,1
204 DATA D-5 ,1,E-5 ,1
205 DATA F-5 ,1,E-5 ,2
206 DATA D-5 ,1,D-5 ,7
207 DATA D-5 ,1,D-5 ,1
208 DATA C-5 ,1,A#-4,1
209 DATA A-4 ,1,G-4 ,1
210 DATA A#-4,2,A-4 ,1
211 DATA A-4 ,3,G-4 ,2
212 DATA F-4 ,2,A-4 ,1
213 DATA G-4 ,3,D-4 ,2
214 DATA F-4 ,2,A-4 ,1
215 DATA A-4 ,5
220 DATA A-4 ,4,A-4 ,4
221 DATA D-5 ,2,E-5 ,2
222 DATA F-5 ,2,E-5 ,1
223 DATA D-5 ,1,E-5 ,3
224 DATA D-5 ,1,C-5 ,2
225 DATA D-5 ,2,A-4 ,8
226 DATA A-4 ,4,A-4 ,4
227 DATA D-5 ,2,E-5 ,2
228 DATA F-5 ,2,E-5 ,1
229 DATA D-5 ,1,E-5 ,3
230 DATA D-5 ,1,C-5 ,2
231 DATA E-5 ,2,F-5 ,2
232 DATA C-5 ,2,A#-4,2
233 DATA A-4 ,2
240 DATA F-4 ,2,A-4 ,2
241 DATA G-4 ,2,D-4 ,2
242 DATA F-4 ,2,A-4 ,1
243 DATA A-4 ,5
299 DATA ,-1
300 REM  2. STIMME
301 DATA F-2 ,2,F-3 ,2
302 DATA F-2 ,2,F-3 ,1
303 DATA F-2 ,1,E-2 ,2
304 DATA E-3 ,2,E-2 ,2
305 DATA E-3 ,1,E-2 ,1
306 DATA D-2 ,2,D-3 ,2
307 DATA D-2 ,2,D-3 ,1
308 DATA D-2 ,1,A#-2,2
309 DATA A#-3,2,C-2 ,2
310 DATA C-3 ,2,F-2 ,6
311 DATA E-2 ,2,D-2 ,2
312 DATA D-3 ,2,C-2 ,2
313 DATA C-3 ,2,A#-2,2
314 DATA F-2 ,1,F-2 ,5
320 DATA C#-3,4,A-2 ,4
321 DATA D-3 ,2,C-3 ,2
322 DATA A#-2,2,G-2 ,2
323 DATA C-3 ,4,C-2 ,4
324 DATA F-2 ,2,A-2 ,2
325 DATA C-3 ,2,A-2 ,1
326 DATA G-2 ,1
327 DATA A-2 ,4,C#-3,4
328 DATA D-3 ,2,C-3 ,2
329 DATA A#-2,2,G-2 ,2
330 DATA C-3 ,4,C-2 ,4
331 DATA F-2 ,8
340 DATA D-3 ,4,C-3 ,4
341 DATA A#-2,2,F-2 ,1
342 DATA F-2 ,5
399 DATA ,-1

READY.
```

3.4 "Yesterday" - Ein zweistimmiges Musikprogramm

Beim Programm "Yesterday" haben wir es mit einer wichtigen Erweiterung zu tun: Bisher waren die Programme stets einstimmig, d.h. es war immer nur die Melodie zu hören. Auf die Dauer wirkt dies recht monoton, und dem ganzen fehlt eigentlich irgendetwas.

Es ist nun möglich, basierend auf dem "Mull of Kintyre"-Programm, zwei Stimmen parallel ablaufen zu lassen. Auf die dazu nötigen Veränderungen wollen wir im folgenden eingehen.

Im Grund besteht der Unterschied zwischen den beiden Programmen darin, daß beim "Yesterday"-Programm viele Programmzeilen einfach doppelt vorkommen. (Wenn wir später das Programm auf drei Stimmen erweitern, werden diese Zeilen dann eben dreifach erscheinen.) Nehmen wir einmal die Zeilennummern 52 und 54 als Beispiel! Sie gleichen sich fast bis auf Haar, nur daß statt N1$ N2$ verwendet wird und statt D1 D2. Versuchen Sie einmal, ähnliche Parallelen zu entdecken! Ihnen werden die Zeilen 65 und 66 auffallen, aber ab Zeile 80 wird das ganze ziemlich undurchschaubar. Während die Notenumrechnungs-Routine noch recht einfach aussieht, ist nämlich die Spielroutine ziemlich komplex geworden (bei drei Stimmen wird sie noch komplexer).

Das Hauptproblem bei der Spielroutine ist, daß zwei Stimmen normalerweise völlig unabhängig voneinander laufen können. Das hat zur Folge, daß FOR-NEXT-Schleifen, die viele Programmierer, die gewohnt sind, in Basic zu programmieren, jetzt gerne anwenden würden, hier total versagen.

Es hilft hier nur ein counter-System, das es erlaubt, zwei Zählvariablen unabhängig voneinander laufen zu lassen. So können unterschiedliche Notendauern voneinander getrennt verarbeitet werden.

Die eigentlich Spielroutine finden Sie von Zeile 84 bis 89. Hier entspricht Zeilennummer 84 der Nummer 87, Zeilennummer 85 der Nummer 88 und Zeilennummer 86 der Nummer 89. Wo Sie in der ersteren die Arrays L1, H1, D1 und die Variablen W1, C1 und S1 finden, finden Sie in der letzteren die Arrays L2, H2, D2 und die Variablen W2, C2 und S2. Für jede der beiden Stimmen laufen also voneinander unabhängige Variablen.

Auch bei diesem Programm findet man ein Flag-System zur Ablaufsteuerung, das aber hier etwas komplizierter ist. Mit diesem Flag-System wird die Abfolge Hauptteil-Mittelteil-Hauptteil-Coda realisiert.

Was ist denn eine Coda? Das ist ein musikalischer Fachbegriff, der einen Abschnitt eines Musikstücks bezeichnet, der zu Schluß gespielt wird und vorher noch nicht vorkam. Das Gegenteil von der Coda ist der Fade-Out. Dies ist der (eigentlich ziemlich einfallslose) Schluß, mit dem die meisten Musikstücke der Pop-Musik aufhören. Bei Fade-Out wird zu Schluß des Lieds der Refrain immer und immer wieder wiederholt, wobei die Lautstärke immer mehr herabgesetzt wird, bis am Schluß nichts mehr zu hören ist. Man benutzt also eine fallende Dynamik (= Lautstärke), um das Stück zu beenden. Oft wird, wie auch hier, zu Schluß des Stücks das Tempo herabgesetzt (= "ritardando"), um das Stück ausklingen zu lassen.

Um nun diese Abfolge zu steuern, nimmt das Flag drei verschiedene Werte an. Zum Übergang vom Hauptteil auf den Mittelteil wird es von 0 auf 1 gesetzt (Zeile 90). Zum Übergang vom Mittelteil zurück auf den Hauptteil wird es von 1 auf 2 gesetzt (Zeile 91). Wenn der Hauptteil beendet wurde und das Flag auf 2 sitzt, werden die beiden Notenindizes auf die Notenwerte der Coda gesetzt, das Tempo verlangsamt (= "ritardando") und die Coda gespielt. Danach kommt wieder die Abfrage, ob das Stück noch einmal gespielt werden soll.

```
0 REM  "STRAWBERRY FIELDS FOREVER"
1 REM  ARRANGED BY  THOMAS DACHSEL
2 REM
3 REM
4 FOR  I=54272 TO 54296
5 POKE I,0
6 NEXT I
7 REM
8 REM   DEFINITIONEN
9 REM
10 ER=54272
11 TL=ER
12 TH=ER+1
13 PL=ER+2
14 PH=ER+3
15 WE=ER+4
16 AD=ER+5
17 SR=ER+6
18 LF=ER+21
19 HF=ER+22
20 RE=ER+23
21 LA=ER+24
27 REM
28 REM  FREQUENZEN LESEN
29 REM
30 DIM FR$(11),FL(11,4),FH(11,4)
31 FOR I=0 TO 11
32     READ FR$(I)
33     FOR J=0TO 4
34         READ  FR
35         FH(I,J)=INT(FR/256)
36         FL(I,J)=FR-FH(I,J)*256
37     NEXTJ
38 NEXTI
47 REM
48 REM  NOTENDATEN LESEN
49 REM
```

```
50 DIMN1$(200),O1(200),D1(200)
51 DIMN2$(200),O2(200),D2(200)
52 DIMN3$(200),O3(200),D3(200)
53 I=0
54 READN1$(I),O1(I),D1(I):IFD1(I)>0THENI=I+1:GOTO54
55 I=0
56 READN2$(I),O2(I),D2(I):IFD2(I)>0THENI=I+1:GOTO56
57 I=0
58 READN3$(I),O3(I),D3(I):IFD3(I)>0THENI=I+1:GOTO58
59 REM
60 REM  UMRECHNUNGSROUTINE
61 REM
62 DIML1(200),H1(200),L2(200),H2(200),L3(200),H3(200)
63 I=0:PRINTCHR$(147)"1.STIMME:"
64 FOR J=0TO11
65 IF FR$(J)=N1$(I)THENL1(I)=FL(J,O1(I)-2):H1(I)=FH(J,O1(I)-2):PRINTN1$(I)
66 NEXT
67 I=I+1:IFD1(I)>0THEN64
68 I=0:PRINT:PRINT"2.STIMME:"
69 FOR J=0TO11
70 IF FR$(J)=N2$(I)THENL2(I)=FL(J,O2(I)-2):H2(I)=FH(J,O2(I)-2):PRINTN2$(I)
71 NEXT
72 I=I+1:IFD2(I)>0THEN69
73 I=0:PRINT:PRINT"3.STIMME:"
74 FOR J=0TO11
75 IF FR$(J)=N3$(I)THENL3(I)=FL(J,O3(I)-2):H3(I)=FH(J,O3(I)-2):PRINTN3$(I)
76 NEXT
77 I=I+1:IFD3(I)>0THEN74
87 REM
88 REM  ANFANGSKLANG EINSTELLEN
89 REM
90 POKEAD,41:POKEAD+7,41:POKEAD+14,41
91 POKESR,121:POKESR+7,121:POKESR+14,121
92 POKELA,31:POKERE,0
93 W1=16:W2=16:W3=16
94 FLAG=0
95 S1=0:S2=0:S3=0:REM INDIZES
96 C1=0:C2=0:C3=0:REM COUNTER
100 REM
```

```
101 REM STUECK SPIELEN
102 REM
103 POKETL,L1(S1):POKETH,H1(S1):POKEWE,W1OR1
104 POKETL+7,L2(S2):POKETH+7,H2(S2):POKEWE+7,W2 OR 1
105 POKETL+14,L3(S3):POKETH+14,H3(S3):POKEWE+14,W3 OR 1
110 C1=C1+1:IFC1>(D1(S1)*.5)THENPOKEWE,W1
111 IFC1<(D1(S1))THEN120
112 C1=0:S1=S1+1:POKETL,L1(S1):POKETH,H1(S1):POKEWE,W1OR1
120 C2=C2+1:IFC2>(D2(S2)*.7)THENPOKEWE+7,W2
121 IFC2<(D2(S2))THEN130
122 C2=0:S2=S2+1:POKETL+7,L2(S2):POKETH+7,H2(S2):POKEWE+7,W2OR1
130 C3=C3+1:IFC3>(D3(S3)*.8)THENPOKEWE+14,W3
131 IFC3<(D3(S3))THEN140
132 C3=0:S3=S3+1:POKETL+14,L3(S3):POKETH+14,H3(S3):POKEWE+14,W3OR1
140 IFS1=14 THEN W1=64:POKE PH,3
141 IFS2=2THENIFC2=9THENW2=32:W3=32:POKEAD+14,7:POKESR+14,169:POKESR+7,139
142 IFS2=22 THEN IFC2=5 THEN POKEHF,210:POKERE,7:IF FLAG=2 THEN 151
143 IFS1=96 THEN IFC1=15THEN POKERE,0:FLAG=FLAG+1:S1=14:S2=3:S3=9:GOTO 96
150 IFD1(S1)>0ORD2(S2)>0ORD3(S3)>0THEN110
151 PRINTCHR$(147)"NOCHMAL? (J)":POKE198,0:WAIT198,1:GETC$:IFC$<>"J"THENEND
152 C1=0:C2=0:C3=0:S1=0:S2=0:S3=0:GOTO90
200 REM FREQUENZEN
201 DATAC,1072,2145,4291,8583,17167
202 DATAC#,1136,2273,4547,9094,18188
203 DATAD,1206,2408,4817,9634,19269
204 DATAD#,1275,2551,5103,10207,20415
205 DATAE,1351,2703,5407,10814,21629
206 DATAF,1432,2864,5728,11457,22915
207 DATAF#,1517,3034,6069,12139,24278
208 DATAG,1607,3215,6430,12860,25721
209 DATAG#,1703,3406,6812,13625,27251
210 DATAA,1804,3608,7217,14435,28871
211 DATAA#,1911,3823,7647,15294,30588
212 DATAB,2025,4050,8101,16203,32407
1000 REM 1.STIMME
1001 DATAB,4,5
1002 DATAB,4,5
1003 DATAB,4,5
1004 DATAB,4,5

READY.
```

```
1005 DATAB,4,5
1006 DATAB,4,5
1007 DATAA,4,5
1008 DATAG#,4,7
1009 DATAF#,4,3
1010 DATAA,4,6
1011 DATAE,4,3
1012 DATAG#,4,4
1013 DATAF#,4,5
1014 DATAE,4,10
1015 DATA,0,4
1016 DATAC#,5,2
1017 DATAC#,5,2
1018 DATAD,5,2
1019 DATAC#,5,4
1020 DATAA,4,6
1021 DATAE,4,2
1022 DATAF#,4,2
1023 DATAB,4,2
1024 DATAA,4,4
1025 DATAG,4,9
1026 DATAG,4,3
1027 DATAA,4,3
1028 DATAB,4,3
1029 DATAE,4,8
1030 DATA,0,15
1031 DATAG,4,3
1032 DATAA,4,3
1033 DATAA#,4,3
1034 DATAE,4,8
1035 DATA,0,6
1036 DATAE,4,2
1037 DATAE,5,2
1038 DATAD,5,2
1039 DATAC#,5,2
1040 DATAB,4,2
1041 DATAA#,4,1
1042 DATAB,4,1
1043 DATAC#,5,4
1044 DATA,0,10
1045 DATAC#,5,2
1046 DATAA,4,2
1047 DATAF#,4,2
1048 DATAC#,5,2
1049 DATAA,4,2
1050 DATAE,4,2
1051 DATAB,4,2
1052 DATAA,4,10
1053 DATA,0,6
1054 DATAB,4,3
1055 DATAB,4,3
1056 DATAB,4,3
1057 DATAB,4,3
1058 DATAB,4,3
1059 DATAB,4,3
1060 DATAB,4,6
1061 DATAB,4,11
1062 DATA,0,2
1063 DATAA,4,2
1064 DATAA,4,2
1065 DATAA,4,2
1066 DATAC#,5,2
1067 DATAB,4,2
1068 DATAA,4,2
1069 DATAG#,4,2
1070 DATAA,4,1
1071 DATAG#,4,1
1072 DATAF#,4,4
1073 DATA,0,12
1074 DATAA,4,2
1075 DATAA,4,2
1076 DATAA,4,2
1077 DATAG#,4,3
1078 DATAF#,4,1
1079 DATAE,4,2
1080 DATAC#,5,2
1081 DATAA,4,2
1082 DATAA,4,1
```

```
1083 DATAA,4,1
1084 DATAA,4,1
1085 DATAB,4,1
1086 DATAA,4,1
1087 DATAG#,4,1
1088 DATAF#,4,4
1089 DATA,0,6
1090 DATAA,4,2
1091 DATAA,4,2
1092 DATAA,4,2
1093 DATAB,4,2
1094 DATAA,4,2
1095 DATAG#,4,2
1096 DATAA,4,2
1097 DATAA,4,16
1999 DATA,-1,-1
2000 REM 2.STIMME
2001 DATA,0,58
2002 DATAD,4,5
2003 DATAE,4,10
2004 DATAA,3,32
2005 DATAB,3,8
2006 DATAB,3,8
2007 DATAB,3,8
2008 DATA,0,8
2010 DATAC#,3,2
2011 DATAF#,3,2
2012 DATAA#,3,12
2013 DATAA#,3,16
2014 DATAF#,3,6
2015 DATAG#,3,2
2016 DATAA#,3,8
2017 DATAA#,4,2
2018 DATAF#,4,2
2019 DATAE,4,2
2020 DATAF#,4,2
2021 DATAF#,3,6
2022 DATAF#,3,6
2023 DATAC#,3,12
2024 DATA,0,6
2025 DATAG#,4,9
2026 DATAG#,4,9
2027 DATAG#,4,9
2028 DATAA,4,4
2029 DATAG#,4,4
2030 DATAF#,4,8
2031 DATAE,4,8
2032 DATAD,4,8
2033 DATAF#,3,4
2034 DATAE,3,4
2035 DATAF#,4,8
2036 DATAD,4,6
2037 DATAE,4,4
2038 DATAC#,4,6
2039 DATAA,3,4
2040 DATA,0,4
2041 DATAF#,4,8
2042 DATAE,4,8
2043 DATAF#,4,8
2044 DATAC#,4,8
2999 DATA,-1,-1
3000 REM 3.STIMME
3001 DATAE,4,10
3002 DATAD#,4,10
3003 DATAD,4,10
3004 DATAF#,4,5
3005 DATAF,4,5
3006 DATAC#,4,9
3007 DATAB,3,9
3008 DATAA,3,5
3009 DATAC#,4,10
3010 DATAA,2,32
3011 DATAE,2,6
3012 DATAE,2,2
3013 DATAE,2,8
3014 DATAE,2,2
3015 DATAG,2,2
3016 DATAB,2,2
```

```
3017 DATAG,2,2
3018 DATAE,2,2
3019 DATAG,2,2
3020 DATAB,2,2
3021 DATAG,2,2
3022 DATAF#,2,16
3023 DATAF#,2,6
3024 DATAF#,2,2
3025 DATAF#,2,8
3026 DATAD,2,6
3027 DATAE,2,2
3028 DATAF#,3,8
3029 DATAF#,3,6
3030 DATAE,3,2
3031 DATAD,3,6
3032 DATAD,3,4
3033 DATAE,3,2
3034 DATAA,2,18
3035 DATAE,4,9
3036 DATAD#,4,9
3037 DATAD,4,9
3038 DATAF#,4,4
3039 DATAF,4,4
3040 DATAC#,4,16
3041 DATA,0,8
3042 DATAF#,2,4
3043 DATAE,2,4
3044 DATAD,2,8
3045 DATAE,2,8
3046 DATAA,2,8
3047 DATAF#,2,4
3048 DATAE,2,4
3049 DATAD,2,8
3050 DATAE,2,8
3051 DATAD,2,8
3052 DATAA,2,8
3999 DATA,-1,-1

READY.
```

3.5 Die Struktur eines dreistimmigen Musikprogramms in Basic

Die Struktur des Programms "Strawberry Fields Forever" stellt die Grenze des Machbaren dar, was man in Commodore Basic an Musikprogrammierung fertigbringen kann. Bei den früheren Programmen hatten wir immer eine Abfrage, wie schnell das Stück abgespielt werden soll. Es war also immer noch eine Art "Geschwindigkeitsreserve" vorhanden. Aber sobald man alle drei Stimmen verwenden möchte und dann noch nicht-lineare Abfolgen programmieren will (wie z.B. der Wechsel Strophe - Refrain - Strophe - Refrain), ist es mit dieser Reserve aus und vorbei.

Die Befehle zum Bearbeiten von Arrays, gekoppelt mit IF-THEN-Abfragen, um die man bei mehrstimmiger Programmierung nicht herumkommen wird, sind in Commodore Basic recht langsam. Wenn man sich vorstellt, daß nach wirklich jeder Note des Stücks solche Abfragen erfolgen müssen - das ist programmiertechnisch gar nicht anders machbar - , dann kann man sich leicht denken, was für Geschwindigkeitseinbußen das mit sich bringt.

Nichtsdestoweniger ist es möglich, dreistimmige Stücke in Basic zu programmieren. Wie, das wollen wir uns nun einmal anschauen.

Es dreistimmiges Musikprogramm ist ziemlich lang, daß werden Sie beim Eintippen sehr schnell feststellen. Und beim Eingeben langer Programme sollte man auf zwei Dinge besonders achten: 1. die Übersicht im Programm zu behalten und 2. Zeit einzusparen, wo auch immer es geht.

Wer Erfahrung im Umgang mit Commodore Basic hat, wird hier schon einige Tricks kennen. Fangen wir mit der Übersichtlichkeit an. Das ist in Commodore Basic ein sehr heikles Thema, denn die Übersichtlichkeit steht hier im offenen Kampf mit der Speicherplatzersparnis. Jeder Space (Leerzeichen) schluckt Speicherplatz, macht das Programm länger und langsamer.

Schauen Sie sich einmal die Zeilen 110 bis 132 vom Programm "Strawberry Fields Forever" an! Keinen einzigen Space werden Sie entdecken. Für einen Informatiker ist so etwas ein Greuel, aber der Benutzer von Commodore Basic muß damit leben. Nehmen wir als Gegenbeispiel die Zeilen 31 bis 38. So sieht eine sauber eingerückte Schleifenschachtelung aus. Allerdings machen die zusätzlichen Spaces auch wieder das Programm länger und langsamer.

Sie werden übrigens enorme Schwierigkeiten haben, diese Zeilen genauso einzugeben, wie sie im Listing stehen. Normale führende Spaces werden vom Editor einfach ignoriert! Probieren Sie es aber einmal mit einem geSHIFTeten Space direkt nach der Zeilennummer aus.

Zur Übersichtlichkeit verhelfen auch ein paar REM's, aber die sollte man wirklich sehr spärlich säen. Schon die Zeilennummer und der Basic-Befehl schlucken fünf Bytes, von den "ausführlichen Kommentaren" in den REM-Zeilen ganz zu schweigen.

Kommen wir nun zur Zeitersparnis beim Eingeben, die oft mit der Speicherplatzersparnis einhergeht. Nehmen wir die Zeilen 10 bis 21 als Beispiel. Früher haben wir ausführlich geschrieben "ERSTER = 54272". Hier kann man zuerst die Spaces weglassen, aber es reicht auch, wenn man "ER" statt "ERSTER" schreibt. In Commodore Basic sind nämlich nur die ersten beiden Zeichen, also in diesem Falle "ER", signifikant. Das bedeutet, daß Variablen nur anhand der ersten beiden Zeichen unterschieden werden. Für den Commodore Basic-Interpreter sind also "EISKREM" und "EIFFELTURM" dasselbe. Aber hier haben wir auch wieder den Fall, daß die Speicherplatzersparnis auf Kosten der Übersichtlichkeit geht.

Wo kann man beim Eintippen von Programmen noch Zeit sparen? Man sollte einmal nachschauen, ob man Paare von Zeilen findet, in denen fast das gleiche steht und die sich nur um wenige Zeichen unterscheiden.

Im Programm "Strawberry Fields Forever" findet man z.B. bei den Zeilen 50,51 und 52, daß sie sich bis auf drei Zahlen genau gleichen. Man kann nun beim Eintippen folgendermaßen vorgehen: Man gibt ein "50 DIMN1$(200),O1(200),D1(200)" wie im Listing angegeben und drückt RETURN. Nun drückt man die Taste "Cursor nach unten" und die SHIFT-Taste gleichzeitig. Dadurch gelangt man mit dem Cursor zurück in die soeben eingegebene und abgeschickte Zeile. Nun benutzt man die Taste "Cursor nach rechts", um die Zeichen zu übergehen, die in beiden Zeilen, also in der schon eingegebenen und in der jetzt einzugebenden Zeile, gleich sind. Man braucht dann nur noch die unterschiedlichen Zeichen einzugeben. Im Beispiel müßte man dann nur die Zeilennummer von 50 auf 51 und dreimal die "1" in eine "2" umändern. Wenn man danach RETURN drückt, hat man beide Zeilen im Speicher. Um nach aus der Zeile 51 die Zeile 52 zu einzugeben, geht man genauso vor.

Dieses Verfahren läßt sich auch auf mehrere Zeilen ausdehnen, man braucht also nicht immer die Zeilen einzeln umändern. Nehmen wir die nun folgenden Zeilen 53 und 54 als Beispiel. Diese beiden Zeilen gibt man nacheinander ein, so daß sie auf dem Bildschirm direkt untereinander stehen. Danach geht man mit dem Cursor zwei Zeilen nach oben (man muß also "Cursor nach unten" und "SHIFT" zweimal drücken). Da die Zeilen 53 und 55 identisch sind, braucht man zunächst nur die Zeilennummer 55 einzugeben und danach RETURN zu drücken. In der nächsten Zeile muß man dann zusätzlich zur Zeilennummer unter Zuhilfenahme der "Cursor nach rechts"-Taste insgesamt viermal aus einer "1" eine "2" machen. An der Anweisung "I=I+1" muß man nichts ändern. Zuletzt muß noch "GOTO54" in "GOTO56" geändert werden. Um die Zeilen 57 und 58 einzugeben, geht man analog vor.

Wenn man dieses Verfahren einmal in den Griff bekommen hat und ein bißchen Geschick im Umgang mit den Cursorsteuerungstasten entwickelt hat (ein großer Nachteil des Commodore 64 ist, daß man immer die SHIFT-Taste mit drücken muß, wenn man mit dem Cursor nach links oder nach oben fahren will), kann man sich so einige Zeit sparen.

Mit der Zeit wird einem auch das Auge dafür geschärft, wie man Basic-Zeilen erkennt, die man so eingeben kann. Beim Programm sind dies noch die Zeilen 63-67 (ändern auf 68-72 und 73-77) und die Zeilen 110-112 (ändern auf 120-122 und 130-132).

Es ist sogar zeitsparender, nach diesem Verfahren zu arbeiten, als diese Zeilen mit dem AUTO-Befehl eines Basic-Toolkits einzugeben. Bei den vielen DATA's ab Zeile 200 ist jedoch ein Basic-Toolkit unbedingt erforderlich. Es ist hier zwar auch möglich, mit der "Cursor nach oben"-Taste zu arbeiten, aber man muß immer die Zeilennummer ändern, was auf die Dauer sehr langweilig wird. Mit einem Basic-Toolkit hat man es da viel einfacher: Man legt einfach den Befehl "DATA" auf irgendeine Funktionstaste, z.B. F1, gibt AUTO 201,1 ein und legt los. Wenn man eine Zeile fertig hat, drückt man RETURN und F1. Darauf erscheint die nächste Zeilennummer und "DATA". So erspart man sich eine Menge Zeit.

Nachdem wir besprochen haben, wie man das dreistimmige Programm "Strawberry Fields Forever" eingibt, kommen wir nun auf die Struktur dieses Programms zu sprechen.

Wenn man durchschaut hat, wie das "Yesterday"-Programm arbeitet, hat man keine Schwierigkeiten, die dort angewendete Arbeitsweise des Programms von zwei auf drei Stimmen zu übertragen. Wie wir schon beim Eingeben von "Strawberry Fields Forever" gesehen haben, gibt es einige Programmzeilen, die fast identisch sind. Diese Zeilen traten immer in Dreiergruppen auf (da wir es ja mit drei Stimmen zu tun haben). Man sieht also, daß ein Musikprogramm, daß drei Stimmen benutzt, wenigstens dreimal so viel zu tun hat wie eins, daß nur eine Stimme ansteuert. Allerdings ist der Speicherplatzbedarf auch dreimal so hoch, und zwar nicht nur der vom Programm selbst (jede der drei Stimmen hat ja ihre eigenen Tonhöhen und Tondauern), sondern auch von den verwendeten Arrays.

Man stößt hier leicht an eine Grenze. Im Beispiel findet man 15 Arrays mit 200 Elementen, und jedes Arrayelement hat um die 5 Byte. Das sind dann schon dicke 15000 Byte, und um die zu verwalten, hat der Basic-Interpreter einiges zu schaffen.

Na, und da denkt man doch nur daran, irgendwie sonst Speicherplatz zu sparen bzw. den noch vorhandenen möglichst effektiv zu nutzen. Das geschah auch bei den Tonfrequenzen. Im "Yesterday"-Programm hatte noch jede Tonfrequenz ihre eigene Zeilennummer, und von Zeile 100 bis 192 war alles in Beschlag genommen. Im jetzigen Programm haben wir alle Tonfrequenzen von Zeile 201 bis 212 dicht an dicht, wobei eine Menge Speicherplatz gespart wird (jede Zeilennummer mit zugehörigem DATA-Befehl benötigt ja fünf Byte). Außerdem werden nunmehr die Oktaven (also C2, C3, C4 usw.) nicht mehr innerhalb der Noten-Strings angegeben, sondern als Indizes. Deswegen hat das Array für die Tonfrequenzen zwei Dimensionen, eine für die zwölf Halbtöne und eine zweite für die fünf möglichen Oktaven.

In den Zeilen 30 bis 38 werden die Tonfrequenzen aus den DATA's von Zeile 201 bis 212 in dieses Array eingelesen. Die Zeilen 50 bis 58 lesen die Tonhöhen und Tondauern von allen drei Stimmen in die zugehörigen Arrays. Von Zeile 62 bis 77 haben wir dann die Umrechnungsroutine, die aus den Tonhöhen in der üblichen Notation die eigentlichen Tonfrequenzen ermittelt. Ab Zeile 90 befindet sich dann die Spielroutine für alle drei Stimmen.

Wir haben es bei dieser Routine mit einem 3-Counter-System zu tun. Jede Stimme hat ihren eigenen Counter (C1, C2 und C3) und einen eigenen Index für die drei Arrays pro Stimme (S1, S2 und S3 für die Arrays L1, H1 und D1; L2, H2 und D2; L3, H3 und D3), so daß die Tondauern der drei Stimmen vollkommen unabhängig voneinander gesteuert werden können.

Das Flag-System, was wir von Zeile 140 bis 143 finden, ermöglicht auch eine Sound-Umschaltung. Nach dem Intro des Stücks, das mit einem dumpfen, flötenähnlichen Klang gespielt wird (Dreieck-Wellenform), wird auf einen helleren, brillanten Klang umgeschaltet. Nach dem Hauptteil des Stücks wird auf einen dumpferen, weicheren Klang umgeschaltet (Tiefpass-Filter) und ein Zwischenteil gespielt.

Die Gesamtabfolge des Stücks Intro - Hauptteil - Zwischenteil - Hauptteil - Zwischenteil - Hauptteil wird über ein Flag gesteuert. Das Flag erhält in Zeile 94 den Anfangswert 0. Jedesmal, wenn der Zwischenteil einmal abgelaufen ist, wird das Flag um 1 erhöht (das steht in Zeile 143). Jedesmal, wenn der Hauptteil einmal abgelaufen ist, wird geprüft, ob das Flag auf 2 sitzt (das steht in Zeile 142). Ist das der Fall, so wird das Stück beendet.

3.6 Hinweise zum Erstellen eigener Basic-Musikprogramme

Es ist ohne weiteres möglich, das "Strawberry Fields Forever"-Programm so zu ändern, das es ein beliebiges anderes Musikstück spielt. Wie man diese Änderungen vornimmt, soll im folgenden besprochen werden.

Die Programmzeilen, die immer beibehalten werden müssen, werden im folgenden fix genannt. Diejenigen Zeilen, die von Stück zu Stück verändert werden müssen, heißen variabel.

Gehen wir das Programm durch: Von Zeile 0 bis 89 haben wir fast durchgehend fixe Programmzeilen. Ausnahmen sind hier nur diejenigen Programmzeilen, die DIM-Befehle enthalten. Hier kann es unter Umständen notwendig sein, den Dimensionierungsindex höher zu setzen. Wie hoch dieser Dimensionierungsindex sein darf, hängt von der Größe des Programms selbst ab. Bei eventuellen Änderungen müssen Sie darauf achten, daß die Dimensionierungsindizes zusammengehöriger Arrays gleich sind. Zusammengehörige Arrays sind N1$, O1, D1, L1 und H1; N2$, O2, D2, L2 und H2 sowie N3$, O3, D3, L3 und H3.

Die Programmzeilen 90 bis 93 sind variabel. Hier wird der Anfangsklang der drei Stimmen eingestellt. Welche Anweisungen und Werte hier möglich sind, wird im vierten Kapitel behandelt.

Die Programmzeilen 94 bis 132 sind fix. Hier handelt es sich um die Spielroutinen.

Die Programmzeilen 140 bis 149 sind variabel. Hier können beliebige Flagabfragen programmiert werden. Falls Sie keine Flagabfage benötigen, muß trotzdem eine Zeile 140 vorhanden sein (z.B. 140 REM), da Zeile 131 in diese Zeile springt.

Die Programmzeilen 150 bis 212 sind fix. Hier befindet sich die Abfrage, ob das Stück nochmal gespielt werden soll, und die Frequenztabelle.

Die Programmzeilen 1000 bis 1998, 2000 bis 2998 und 3000 bis 3998 sind frei für die Tonhöhen und Tondauern der einzelnen Stimmen. Es ist hier ohne weiteres auch möglich, die Daten zu mehreren Tönen in eine Zeile zu schreiben. Für einen Ton stehen hier jeweils drei Werte, die durch Kommata getrennt sind:

Der erste Wert ist ein String, der den zu spielenden Halbton symbolisiert (C,C#,D,D#,E,F,F#,G,G#,A,A#,B). Als erster Wert kann auch der Leerstring stehen (DATA"",... oder auch DATA,... ergibt bei READ A$ beides den Leerstring). Falls dies der Fall ist, wird für die gegebene Dauer (dritter Wert) die Frequenz 0 gespielt. Dies ist ein nicht hörbarer Ton. Das kann man dazu benutzen, um Pausen im Stück zu spielen.

Der zweite Wert ist die Oktave und muß zwischen 2 und 6 liegen. Der dritte Wert ist die Tondauer. Eine "1" steht hier für ein Achtel, eine "2" für ein Viertel, eine "3" für ein punktiertes Viertel (das sind 3 Achtel), eine "4" für eine halbe Note usw.

Die Programmzeilen 1999, 2999 und 3999 sind fix. Sie dienen zur Trennung der Tonhöhen und Tondauern der einzelnen Stimmen. Wenn in Programmzeile 54 (bzw. 56 und 58) ein Wert kleiner gleich 0 gelesen wird, geht das Programm davon aus, das alle Töne der betreffenden Stimme gelesen wurden. Die -1, die hier im Programm erscheint, nennt man auch Begrenzer oder Delimiter (das ist ein Wert, der einen Datenblock abschließt).

3.7 Anmerkung zu den Beispielprogrammen

Wieso kommen in diesem Buch eigentlich so viele Beatles-Lieder als Beispielprogramme? Es soll deutlich werden, daß der Commodore 64 mit seinem Sound-Chip mehr kann als "nur" Bach-Fugen spielen oder Effektsounds erzeugen. Ihm stehen praktisch alle Bereiche der Musik offen (sieht man einmal von den "nur" drei Stimmen ab) - also auch Pop- und Rockmusik. Ein gutes Beispiel hierfür sind die Lieder einer Gruppe, die die Musik der sechziger Jahre entscheidend mitbestimmte und deren Lieder heute immer noch gespielt werden: die Beatles.

John Lennon, Paul McCartney, George Harrison und Ringo Starr: das sind Namen, die heute auf der ganzen Welt bekannt sind. Es wurden schon so viele Bücher über den Beatles-Mythos geschrieben, daß an dieser Stelle nur einige Stationen ihrer Karriere genannt werden sollen.

Eine Stadt wird mit den Beatles oft in einem Atemzug genannt: Liverpool in England. Alle vier Beatles stammen aus dieser Stadt. Die erste wichtige Station in ihrer Karriere waren ihre Auftritte im Star-Club in Hamburg. 1963 war das "Jahr der Beatles". Die wichtigsten Singles, die die Beatles in diesem Jahr herausbrachten, hießen "Love Me Do", "Please Please Me", und die wohl bekannteste "She Loves You". Das "Yeah-Yeah-Yeah" aus diesem Lied sollte zur Parole einer Begeisterungswelle werden, die 1963 und 1964 die ganze Welt ergriff: die "Beatlemania" ("Beatles-Manie").

Die LPs und Singles, die die Beatles in dieser Zeit herausbrachten, verkauften sich in Millionenhöhe, und Tourneen, Filme und Skandale der Gruppe taten ihr übriges, um ihre Popularität immer weiter zu steigern.

Doch kein Mensch, nicht einmal die Beatles, kann der Showmaschinerie auf Dauer standhalten. Nach 1966 traten die Beatles nicht mehr öffentlich auf, sondern arbeiteten nur noch in den Plattenstudios. Private und auch wirtschaftliche Probleme traten auf, als die Gruppe ihr eigenes Management übernehmen wollte. Und diese Probleme sind auch schuld daran, daß die Beatles sich im Jahr 1970 auflösten. Jeder der vier versuchte sich an Soloprojekten, aber diese kamen sehr unregelmäßig und hatten sehr unterschiedlichen Erfolg.

Gerüchte, die Beatles würden wieder zusammen auftreten, wurden endgültig Anfang der Achtziger zunichte gemacht, als John Lennon von einem Fan in New York erschossen wurde. Der einzige der Beatles, der heute noch Erfolg hat, ist Paul McCartney, der von Zeit zu Zeit, kürzlich sogar zusammen mit Michael Jackson ("Say Say Say"), noch in den Hitparaden auftaucht.

Wenn man sich mit der rein musikalischen Seite der Beatles befaßt, so wird man feststellen, daß der Großteil aller der Lieder, die zu Evergreens geworden sind, vom Autorengespann Lennon-McCartney stammt, wobei die Lieder mit mehr psychedelischem Charakter ("Strawberry Fields Forever") von John Lennon stammen und Lieder wie "Yesterday" oder "Let it be", also Pop-Songs schlechthin, von Paul McCartney.

Zu den Liedern der Beatles gibt es eine dermaßen große Anzahl von Cover-Versionen, daß selbst die Tantiemen davon in die Millionen gehen. Sinfonieorchester, Unterhaltungsbands (z.B. James Last) und andere Rock- und Pop-Gruppen (z.B. The Beatles Revival Band) spielen ihre Lieder nach. Nun gibt es ihre Lieder auch auf dem Commodore 64, und vielleicht wird irgendjemand, der dieses Buch liest, noch weitere Lieder von ihnen eingeben.

4. Die Soundregister des Commodore 64

4.1 Allgemeines

Im vorangegangenen Kapitel haben wir uns den Sound-Chip als ein Büro mit vielen Zimmern und Angestellten vorgestellt. Ein Musikprogramm dachten wir uns als Büroboten, der diese Zimmer in bestimmter Reihenfolge besucht und den Angestellten darin immer neue Anordnungen bringt. Im folgenden wollen wir uns damit beschäftigen, was für Angestellte es denn nun alles gibt und was ihre Aufgaben sind.

Die 215 Angestellten im Sound-Chip-"Büro" des Commodore 64 sind auf 29 Zimmer verteilt. Jedes dieser Zimmer hat eine Nummer von 54272 bis 54300. Um uns die Sache einfacher zu machen, numerieren wir sie einfach von 0 bis 28 durch. Wir müssen uns diese Zimmer als ziemlich große Räume vorstellen.

In jedem Raum sind acht Schreibtische, die von 0 bis 7 durchnumeriert sind. An jedem dieser acht Schreibtische sitzt ein Angestellter. Wenn wir also einen bestimmten Angestellten ansprechen wollen, müssen wir neben der Zimmernummer auch die Nummer des Schreibtisches nennen, an dem dieser Angestellte sitzt.

Was kann man denn nun diesen Angestellten sagen? Um das herauszukriegen, sollte man sich daran erinnern, daß wir es ja in Wirklichkeit immer noch mit einem Computer zu tun haben. Und jeder weiß ja, daß ein Computer nur Nullen und Einsen versteht. So ist es auch mit unseren Angestellten. Jeder versteht nur "0" oder "1". Wir sehen also, daß ein Angestellter allein ziemlich wenig kann. Um einen Ton zu spielen, müssen wir mehrere Angestellte zugleich ansprechen.

Nun haben wir das Problem, daß wir den Sound-Chip nicht direkt mit den Nullen und Einsen "füttern" können, von denen wir wollen, daß sie die gewünschten Angestellten erreichen. Wir müssen das von Basic aus machen. Und Basic rechnet ja nicht mit Nullen und Einsen, sondern im Dezimalsystem.

Es gibt aber eine Basic-Anweisung, mit der wir alle Angestellten eines Zimmers direkt erreichen können. In dieser Anweisung steht A für die gewünschte Zimmernummer. X steht für 0 oder 1. Wenn X=0 ist, so kann man den entsprechenden Summanden ganz weglassen. Wenn X=1 ist, kann man das "X*" weglassen.

```
POKE A, X*1  + X*2  + X*4  + X*8
        ^      ^      ^      ^
        0      1      2      3

      + X*16 + X*32 + X*64 + X*128
        ^      ^      ^      ^
        4      5      6      7
```

Die Zahlen unter den X'en stehen für den Angestellten, den wir ansprechen wollen. Für ganz Schlaue: da wir es hier ausschließlich mit Zweierpotenzen zu tun haben, kann man das "+" (das ist eine arithmetische Operation) auch durch ein "OR" (das ist eine logische Operation) ersetzen, ohne daß sich etwas am Ergebnis ändert.

Doch es ist ziemlich umständlich, sich diese Formel zu merken. Viel leichter ist es, sich einige Standardwerte zu merken. Wenn man alle Angestellten auf "0" setzen will, braucht man nur die Anweisung POKE A,0 einzugeben. Falls alle Angestellten auf "1" gesetzt werden sollen, sollte man die Anweisung POKE A,255 gebrauchen.

Nun sind die verschiedenen Funktionen des Sound-Chips nicht willkürlich auf die Angestellten verteilt. In den meisten Zimmern des Büros arbeiten die Angestellten des Büros eng zusammen, so daß es ganz natürlich ist, daß man sie zusammen anspricht.

Wollen wir nun zusammen durchgehen, wie die einzelnen Angestellten zusammenarbeiten. Dazu braucht man nicht unbedingt ein komplettes Basic-Programm, wenn man nur einmal einige Dinge ausprobieren will.

Man kann die nun folgenden Anweisungen direkt eintippen. Es sollte jedoch kein Basic-Programm, das vielleicht gerade im Speicher steht, editiert, d.h. verändert werden, sonst werden die Variablen gelöscht.

Geben Sie ein: ERSTER = 54272. Vielleicht erinnern Sie sich, mit dieser Anweisung begannen immer die Basic-Programme aus dem dritten Kapitel. Geben Sie nun ein: LAUTST = ERSTER+24 und POKE LAUTST,15. Auch diese Anweisung kennen Sie bereits, hiermit drehen wir den "Lautstärkeknopf" des Sound-Chips auf.

Wenn man genau hinhört, hört man einen Knacks im Lautsprecher. Einen Ton hört man jedoch noch nicht. Wir müssen erst angeben, was für eine Tonhöhe wir gerne hätten. Hier brauchen wir eine Tabelle, in der die Frequenzen drinstehen, die wir musikalisch verwerten können. Auf den nächsten Seiten finden Sie diese Tabelle der Tonfrequenzen.

4.2 Tabelle der Tonfrequenzen

Note	Dezimal	Low	High	Hexadezimal	Low	High
C -0	268	12	1	$010C	$0C	$01
C#-0	284	28	1	$011C	$1C	$01
D -0	301	45	1	$012D	$2D	$01
D#-0	318	62	1	$013E	$3E	$01
E -0	337	81	1	$0151	$51	$01
F -0	358	102	1	$0166	$66	$01
F#-0	379	123	1	$017B	$7B	$01
G -0	401	145	1	$0191	$91	$01
G#-0	425	169	1	$01A9	$A9	$01
A -0	451	195	1	$01C3	$C3	$01
B -0	477	221	1	$01DD	$DD	$01
H -0	506	250	1	$01FA	$FA	$01
C -1	536	24	2	$0218	$18	$02
C#-1	568	56	2	$0238	$38	$02
D -1	602	90	2	$025A	$5A	$02
D#-1	637	125	2	$027D	$7D	$02
E -1	675	163	2	$02A3	$A3	$02
F -1	716	204	2	$02CC	$CC	$02
F#-1	758	246	2	$02F6	$F6	$02
G -1	803	35	3	$0323	$23	$03
G#-1	851	83	3	$0353	$53	$03
A -1	902	134	3	$0386	$86	$03
B -1	955	187	3	$03BB	$BB	$03
H -1	1012	244	3	$03F4	$F4	$03
C -2	1072	48	4	$0430	$30	$04
C#-2	1136	112	4	$0470	$70	$04
D -2	1204	180	4	$04B4	$B4	$04
D#-2	1275	251	4	$04FB	$FB	$04
E -2	1351	71	5	$0547	$47	$05
F -2	1432	152	5	$0598	$98	$05
F#-2	1517	237	5	$05ED	$ED	$05
G -2	1607	71	6	$0647	$47	$06

Note	Dezimal	Low	High	Hexadezimal	Low	High
G#-2	1703	167	6	$06A7	$A7	$06
A -2	1804	12	7	$070C	$0C	$07
B -2	1911	119	7	$0777	$77	$07
H -2	2025	233	7	$07E9	$E9	$07
C -3	2145	97	8	$0861	$61	$08
C#-3	2273	225	8	$08E1	$E1	$08
D -3	2408	104	9	$0968	$68	$09
D#-3	2551	247	9	$09F7	$F7	$09
E -3	2703	143	10	$0A8F	$8F	$0A
F -3	2864	48	11	$0B30	$30	$0B
F#-3	3034	218	11	$0BDA	$DA	$0B
G -3	3215	143	12	$0C8F	$8F	$0C
G#-3	3406	78	13	$0D4E	$4E	$0D
A -3	3608	24	14	$0E18	$18	$0E
B -3	3823	239	14	$0EEF	$EF	$0E
H -3	4050	210	15	$0FD2	$D2	$0F
C -4	4291	195	16	$10C3	$C3	$10
C#-4	4547	195	17	$11C3	$C3	$11
D -4	4817	209	18	$12D1	$D1	$12
D#-4	5103	239	19	$13EF	$EF	$13
E -4	5407	31	21	$151F	$1F	$15
F -4	5728	96	22	$1660	$60	$16
F#-4	6069	181	23	$17B5	$B5	$17
G -4	6430	30	25	$191E	$1E	$19
G#-4	6812	156	26	$1A9C	$9C	$1A
A -4	7217	49	28	$1C31	$31	$1C
B -4	7647	223	29	$1DDF	$DF	$1D
H -4	8101	165	31	$1FA5	$A5	$1F
C -5	8583	135	33	$2187	$87	$21
C#-5	9094	134	35	$2386	$86	$23
D -5	9634	162	37	$25A2	$A2	$25
D#-5	10207	223	39	$27DF	$DF	$27

Note	Dezimal	Low	High	Hexadezimal	Low	High
E -5	10814	62	42	$2A3E	$3E	$2A
F -5	11457	193	44	$2CC1	$C1	$2C
F#-5	12139	107	47	$2F6B	$6B	$2F
G -5	12860	60	50	$323C	$3C	$32
G#-5	13625	57	53	$3539	$39	$35
A -5	14435	99	56	$3863	$63	$38
B -5	15294	190	59	$3BBE	$BE	$3B
H -5	16203	75	63	$3F4B	$4B	$3F
C -6	17167	15	67	$430F	$0F	$43
C#-6	18188	12	71	$470C	$0C	$47
D -6	19269	69	75	$4B45	$45	$4B
D#-6	20415	191	79	$4FBF	$BF	$4F
E -6	21629	125	84	$547D	$7D	$54
F -6	22915	131	89	$5983	$83	$59
F#-6	24278	214	94	$5ED6	$D6	$5E
G -6	25721	121	100	$6479	$79	$64
G#-6	27251	115	106	$6A73	$73	$6A
A -6	28871	199	112	$70C7	$C7	$70
B -6	30588	124	119	$777C	$7C	$77
H -6	32407	151	126	$7E97	$97	$7E
C -7	34334	30	134	$861E	$1E	$86
C#-7	36376	24	142	$8E18	$18	$8E
D -7	38539	139	150	$968B	$8B	$96
D#-7	40830	126	159	$9F7E	$7E	$9F
E -7	43258	250	168	$A8FA	$FA	$A8
F -7	45830	6	179	$B306	$06	$B3
F#-7	48556	172	189	$BDAC	$AC	$BD
G -7	51443	243	200	$C8F3	$F3	$C8
G#-7	54502	230	212	$D4E6	$E6	$D4
A -7	57743	143	225	$E18F	$8F	$E1
B -7	61176	248	238	$EEF8	$F8	$EE
H -7	64814	46	253	$FD2E	$2E	$FD

4.3 Wie man diese Tabelle verwendet

Die "Tabelle der Tonfrequenzen" enthält alle Tonfrequenzen, die man für Musikstücke braucht. Die Tabelle ist folgendermaßen aufgebaut: In der ersten Spalte findet man die musikalische Tonbezeichnung von C bis H. Wenn man Musikstücke umsetzt, sollte man hier auf die sogenannte "enharmonische Verwechslung" achten. Dieses Wortungetüm stammt aus der Musiktheorie und bedeutet, daß man in der wohltemperierten Stimmung die Halbtöne Cis und Des, Dis und Es, Fis und Ges, Gis und As sowie Ais und B gleichsetzen kann. Wenn man also vor das Problem gestellt wird, die Frequenz für ein "Es" herauszufinden, schaut man einfach bei "Dis" nach. Hier sind alle "enharmonischen Verwechslungen" in einer Tabelle:

```
Ich suche den Halbton:  Ich schaue bei:
---------------------------------------
...His,   C, Deses.........C...........
......Cis, Des.............C#..........
...Cisis, D, Eses..........D...........
.......Es, Dis.............D#..........
...Disis, E, Fes...........E...........
...Geses, F, Eis...........F...........
......Fis, Ges.............F#..........
...Fisis, G, Ases..........G...........
......Gis, As..............G#..........
...Gisis, A, Heses.........A...........
......Ais, B...............A#..........
...Aisis, H, Ces...........H...........
```

Hier sollte man darauf achten, daß in der englischen Sprache "B" statt "H" steht. Auch in einigen Musikprogrammen taucht das immer wieder auf. Wenn man sich bei irgendeinem Notentext nicht im klaren ist, ob nun "B" oder "H" gemeint ist, sollte man darauf achten, ob die Schreibweise "Bb" vorkommt (also B und das "be" (engl. flat)-Symbol). Falls ja, bedeutet das, daß wann immer ein "B" alleine auftaucht, wohl ein "H" gemeint ist.

Zurück zur Frequenztabelle. Hinter dem Notennamen steht, getrennt durch einen Bindestrich, die Oktave des Tons. Hier ist "0" die tiefste und "7" die höchste Oktave. Alle Halbtöne der "0"-ten Oktave klingen sehr tief und brummig, die Halbtöne der "7"-ten Oktave klingen sehr piepsig und hoch. Musikalisch verwendbare Töne liegen ungefähr von der "1"-ten bis zur "6"-ten Oktave.

In den nächsten Spalten findet man die Dezimalwerte der Tonfrequenzen. Diese Werte werden Sie in jedem Basic-Musikprogramm in einer der beiden angegebenen Formen wiederfinden. Die erste Form ist die vollständige Frequenz, die zunächst noch umgerechnet werden muß. Die Werte unter "Low" und "High" können direkt über POKE eingegeben werden. Die hexadezimalen Werte, die Sie in den rechten Spalten finden, werden in Musikprogrammen in Assembler verwendet.

Wollen Sie nun eine bestimmte Frequenz einstellen, z.B. C-4, geben Sie ein POKE ERSTER, 195 und POKE ERSTER+1, 16. Man kann natürlich auch jede andere Frequenz aus der Tabelle eingeben, wenn man folgendes Format einhält: POKE ERSTER, <Low> und POKE ERSTER+1, <High> mit den entsprechenden Low-/High-Werten aus der Tabelle.

Man hört jetzt aber immer noch keinen Ton! Dem können Sie abhelfen, indem Sie eingeben: POKE ERSTER+6,240:DRUECK = ERSTER+4 und noch zum Einschalten POKE DRUECK,33. Man hört jetzt einen ziemlich lauten, gleichbleibenden Ton. Durch POKE DRUECK, 32 wird der Ton wieder abgeschaltet.

Stellen Sie sich den Befehl POKE DRUECK, 33 wie das Drücken einer Taste an einem Klavier vor und den Befehl POKE DRUECK,32 wie das Loslassen. Probieren Sie diese beiden Anweisungen ein paarmal aus und verändern Sie einmal die Tonfrequenz!

4.4 Ein kurzes Basic-Programm

Vielleicht empfanden Sie es auch als störend, daß die ganzen Befehle, die Sie im sogenannten "Direkt-Modus" eingegeben haben, immer wieder vom Bildschirm verschwanden. Nun, das liegt daran, daß der Computer einen Befehl wie POKE DRUECK,33 ohne vorausgehende Zeilennummer zwar einmal ausführt, aber dann wieder "vergißt".

Solange dieser Befehl noch auf dem Bildschirm steht, kann man mit dem Cursor hochfahren und durch Drücken der Return-Taste den Befehl nochmals ausführen lassen, aber irgendwann verschwindet die entsprechende Zeile durch das "Scrolling" (Bildschirmrollen) vom Bildschirm. Geben Sie einmal folgendes kurze Basic-Programm ein:

```
  0 WELLE  = 32
 10 ERSTER = 54272
 20 LAUTST = ERSTER+24
 30 DRUECK = ERSTER+4
 40 POKE LAUTST, 15
 50 INPUT F
 60 F2=F/256
 70 F1=F-INT(F2)*256
 80 POKE ERSTER,F1
 90 POKE ERSTER+1,F2
100 POKE ERSTER+6,240
110 IF PEEK (197)=64 THEN 140
120 POKE DRUECK, WELLE + 1
130 GOTO 110
140 POKE DRUECK, WELLE
150 GOTO 110
```

Wenn Sie das Programm starten, erscheint ein Fragezeichen. Nun können Sie eine beliebige Frequenz aus der Tabelle eingeben. Nehmen Sie hier aber die erste Zahl unter "Dezimal", denn Sie wird im Programm in die Low-/High-Werte umgerechnet. Darauf können Sie durch Drücken einer Taste den Ton spielen lassen und durch Loslassen den Ton ausschalten. Wenn Sie einen neuen Ton eingeben wollen oder das Programm anhalten wollen, drücken Sie die RUN/STOP-Taste.

4.5 Das Gate-Signal

Ändern Sie im Basic-Programm einmal Zeile 0 auf POKE WELLE, 16 und starten Sie das Programm neu! Sie werden einen dumpferen Ton hören. Wenn Sie in Zeile 0 POKE WELLE, 128 eingeben, hören Sie ein Rauschen. Hier haben wir einige der "Wellenformen", die der Sound-Chip produzieren kann.

Betrachten Sie einmal die Anweisungen POKE DRUECK, WELLE + 1 und POKE DRUECK, WELLE! Löschen Sie in Zeile 120 das "+1" und fügen Sie es in Zeile 140 ein. Starten Sie nun das Programm neu! Jetzt ist es gerade umgekehrt: Jedesmal, wenn eine Taste gedrückt wird, verstummt der Ton. Läßt man die Taste los, so erklingt der Ton wieder.

Was ist denn nun mit dem "DRUECK" los? Ganz offensichtlich haben wir es hier mit einem Zimmer des Sound-Chip-"Büros" zu tun, in dem die Angestellten unterschiedliche Aufgaben zu bewältigen haben. Einen Angestellten können wir direkt durch das "+1" steuern. Dies ist das sogenannte "Gate-Signal".

Jedesmal, wenn wir dem Angestellten durch das "+1" eine "1" übermitteln, heißt das, der Ton wird gespielt (die "Taste" wird gedrückt). Falls das "+1" fehlt, so wird diesem Angestellten eine "0" übermittelt (die "Taste" wird losgelassen).

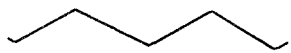
Es gehört also zu jedem neu gespielten Ton dazu, daß das Gate-Signal gesetzt wird und nach einer gewissen Zeit, die mit der Tondauer zusammenhängt, wieder gelöscht wird. Andererseits wird durch das DRUECK auch der Klang des Tons gesteuert. Welche Klänge es nun alles gibt, wollen wir im folgenden betrachten.

4.6 Die Wellenformen des Sound-Chips

Welche Werte die Variable WELLE im Beispielprogramm annehmen darf, können Sie folgender Tabelle entnehmen:

WELLE	Wellenform
16	Dreieck
32	Sägezahn
64	Rechteck
128	Rauschen
(16 OR 32)	Dreieck + Sägezahn
(32 OR 64)	Sägezahn + Rechteck
(16 OR 64)	Dreieck + Rechteck
(16 OR 32 OR 64)	Dreieck + Sägezahn + Rechteck

Die Namen der Wellenformen wie "Dreieck", "Sägezahn" usw. stammen aus der Physik. Diese Namen stehen für das Aussehen dieser Wellenformen auf einem Oszilloskop. Die folgenden Zeichnungen sollen Ihnen einen Eindruck vermitteln, wie das aussieht:

Wellenform	Oszilloskop
Dreieck	
Sägezahn	
Rechteck	
Rauschen	
Dreieck + Sägezahn	
Sägezahn + Rechteck	
Dreieck + Rechteck	
Dreieck + Sägezahn + Rechteck	

Jeder dieser Wellenformen ist ein Gemisch aus bestimmten Sinustönen in einem genau festgelegten Verhältnis. Den tiefsten Sinuston bezeichnet man als Grundton. Dieser Grundton wird durch die Tonfrequenz festgelegt. Zu dem Grundton kommt (außer bei einem reinen Sinuston) ein Gemisch aus Obertönen, das für den "Klang" eines Tones charakteristisch ist.

Bei der Sägezahnschwingung ist der n-te Oberton 1/n mal so laut wie der Grundton. Es sind also alle Obertöne vertreten, und zwar in einem Lautstärkeverhältnis 1 : 1/2 : 1/3 : 1/4 : 1/5 ..., wobei 1 die Lautstärke des Grundtons ist. Die Sägezahnschwingung klingt aufgrund der Obertonreichheit "strahlend", "grell", "trompetenhaft", "glänzend", um nur einige der Adjektive zu nennen, die man als Beschreibung anführen könnte.

Jedoch muß betont werden, daß eine Sägezahnschwingung, wie sie der Commodore 64 produziert, nie in dieser reinen Gestalt in der Natur auftritt. Es handelt sich hierbei um eine nur elektronisch erzeugbare Schwingung. Dies gilt übrigens für alle durch den Sound-Chip erzeugten Wellenformen.

Bei der Dreieckschwingung haben wir wesentlich weniger Obertöne. Daraus resultiert ein dumpfer, flötenähnlicher Klang.

Bei der Rechteckschwingung, auch Pulswelle genannt, sind alle ungeradzahligen Obertöne vertreten, so daß wir einen hohlen, leicht "näselnden" Klang ähnlich wie bei einer Oboe haben. Das Lautstärkenverhältnis der einzelnen Obertöne kann man bei der Rechteckschwingung durch die sogenannte Pulsbreite einstellen.

Einen besonderen Effekt erreicht man dadurch, daß man die Pulsbreite kontinuierlich ändert. Dies nennt man dann PWM (Pulse Width Modulation) oder Pulsbreitenmodulation. Wie sich das anhört, können Sie sich durch Einfügen der folgenden Programmzeilen in das Beispielprogramm und durch Ändern der Zeile 0 auf POKE WELLE,64 anhören:

```
115 A = A + 16: IF A=256 THEN A=16
116 POKE ERSTER + 2, (A*16) AND 255
117 POKE ERSTER + 3, A/16
```

Die "Zimmer" des Sound-Chips ERSTER + 2 und ERSTER + 3 steuern hierbei die Pulsbreite. Wenn man die Rechteck-Wellenform verwendet, muß man diesen Zimmern einen Wert zuweisen, sonst ist kein Ton zu hören. Im Zimmer ERSTER+3 sitzen übrigens nur vier Angestellte, die übrigen vier "Schreibtische" (Bits) in diesem Zimmer sind leer. Man sollte also nur Werte zwischen 0 und 15 hineinPOKEn.

Weiterhin haben wir noch das "weiße" Rauschen als Wellenform. Hierbei haben alle Obertöne die gleiche Lautstärke, so daß ein kontinuierliches Frequenzspektrum entsteht, was sich wie ein Rauschen anhört.

Die Wellenformen, die sich durch Verknüpfung der bereits vorgegebenen Wellenformen durch den Sound-Chip erzeugen lassen, entstehen durch AND-Verknüpfung innerhalb des Sound-Chips. Die meisten dieser Wellenformen sind von der Lautstärke her sehr leise und lassen sich musikalisch nur begrenzt einsetzen.

4.7 Wie man die "ADSR-Werte" programmiert

Wenn wir bei unserem Beispielprogramm auf eine Taste drücken, hört es sich an, als sei der Ton sofort da. Wenn wir die Taste loslassen, scheint der Ton sofort zu verschwinden. In Wirklichkeit brauchen diese Vorgänge "Ton an" und "Ton aus" eine bestimmte Zeit. Und genau diese Zeiten lassen sich durch den Sound-Chip genau kontrollieren.

Betrachten Sie im Beispielprogramm, das wir die ganze Zeit benutzt haben, einmal die Zeile 100! Dies ist eins der "Zimmer" des Sound-Chip-"Büros", das diese Zeiten kontrolliert. Fügen Sie nun die Programmzeilen 91 bis 95 ein und verändern Sie die Zeile 100, wie es auf der nächsten Seite oben abgedruckt ist.

```
 91 A=0
 92 D=0
 93 S=15
 94 R=0
 95 POKE ERSTER+5, A * 16 + D
100 POKE ERSTER+6, S * 16 + R
```

Wenn Sie das Programm nun erneut starten, werden Sie keinen Unterschied feststellen. Hier haben wir die vollständige Darstellung der Zeiten, die bestimmen, wie schnell ein Ton ein- und ausschwingt. Bisher konnten wir ohne weiteres schreiben: POKE ERSTER+6,240, da 15*16 = 240. Doch um schnell Änderungen vornehmen zu können, empfiehlt sich für die Zeilen 91 bis 94 auch folgende Schreibweise:

```
91 INPUT A
92 INPUT D
93 INPUT S
94 INPUT R
```

Wenn Sie das Programm jetzt starten, so muß man nach der Tonfrequenz vier Werte eingeben. Jeder dieser vier Werte muß zwischen 0 und 15 liegen. Falls es Ihnen lästig erscheint, bei jedem Programmstart eine neue Frequenz einzugeben, können Sie die Zeile 50 auch ändern auf: 50 F = 4291 z.B. für den Ton C4.

Sie sehen also: der Basic-Befehl INPUT erlaubt es, bei jedem Programmablauf neue Werte festzulegen. Wenn man einen bestimmten Wert jedoch für jeden Programmablauf festlegen will, benutzt man das Gleichheitszeichen ("="), um den gewünschten Wert ein für allemal zu definieren.

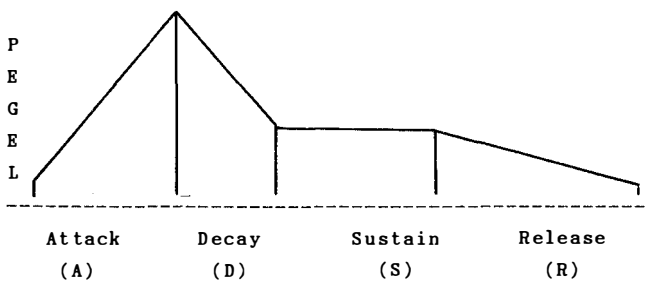
Die vier Werte A, D, S und R heißen nicht ohne Grund so: Wir haben es hier mit der bei Synthesizern üblichen Form ("ADSR-Werte") zu tun, wie man die Zeiten festlegt, die ein Ton braucht, um ein-und auszuschwingen.

Da die exakte Beschreibung in Worten ziemlich kompliziert ist, wollen wir zunächst anhand unseres Beispielprogramms mit den zusätzlichen INPUT-Anweisungen für die vier Werte A, D, S und R einige Beispiele durchgehen. Setzen Sie in Zeile 0 WELLE auf 16, 32 oder 64.

Geben Sie ein:				Wenn Sie eine Taste drücken, hören Sie:
für A:	D:	S:	R:	
1	9	0	0	einen klavierähnlichen, kurzen Ton
5	10	5	10	einen geigenähnlichen, längeren Ton
13	8	8	13	einen langsam ein-/ausschwingenden Ton
0	5	0	0	einen äußerst kurzen Ton
0	0	0	0	gar keinen Ton

Die zweite und dritte Einstellung ist gut dazu geeignet, die einzelnen Phasen des Tonablaufs zu verfolgen. Wenn man eine Taste drückt, schwingt der Ton langsam ein (Attack), erreicht ein Lautstärkemaximum und wird leiser (Decay), verweilt dann auf einem bestimmten Lautstäkepegel (Sustain), um nach Loslassen der Taste endgültig auszuklingen (Release). Wenn man diesen Lautstärkenverlauf einmal aufzeichnet, so könnte das so aussehen:

Die ADSR-Hüllkurve



Man nennt das ganze "Hüllkurve", weil der Ton durch diese Kurve "eingehüllt" wird. Die ADSR-Werte sind für die Form dieser Kurve verantwortlich. Durch Verändern der Werte kann man hunderte verschiedener Hüllkurven erzeugen.

Probieren Sie einmal soviel Kombinationen aus, wie Sie Lust haben! Man wird hier durch Einfallsreichtum immer etwas Neues finden. Gerade darin liegt eine der Stärken eines Synthesizers wie der Sound-Chip des Commodore 64. Man wird immer neue Kombinationen der einstellbaren Werte entdecken.

Manche der obigen Werte werden einen Ihnen vertrauten Lautstärkeverlauf bewirken, z.B. ein kleiner Wert für A. Hier schwingt der Ton sehr schnell ein, wie etwa bei einem Spinett oder Cembalo. Es gibt aber auch eine Fülle von Einstellungen, die sehr fremdartige Ergebnisse liefern.

4.8 Was die ADSR-Werte genau bedeuten

Wir wollen uns nun die Bedeutung der ADSR-Werte ansehen. "ADSR" steht für die englischen Begriffe Attack, Decay, Sustain und Release. Jeder dieser Begriffe steht für eine bestimmte Phase im Tonablauf, die durch den zugehörigen Wert charakterisiert wird. Im einzelnen bedeuten diese Werte folgendes:

1. Attack. Dieser Wert bestimmt die Einschwingzeit des Tons.

Das englische Wort "Attack" bedeutet soviel wie "Anschlag". Wir haben es hier mit einem Wert zu tun, der bestimmt, wie schnell ein Ton einschwingt, d.h. seine Maximallautstärke erreicht.

Attack-Werte, wie sie bei "natürlichen" Musikinstrumenten vorkommen, liegen im Zeitbereich von einigen Millisekunden bis zu mehreren Sekunden. Kurze Attack-Werte finden wir hauptsächlich bei Saiten-Instrumenten, wo die Saiten gezupft oder angeschlagen werden. Dies sind zum Beispiel Instrumente wie Harfe, Gitarre oder die meisten Tasteninstrumente.

Längere Attack-Werte haben wir bei durchweg allen Blasinstrumenten. Hier muß der Spieler ja Luft in das Instrument hineinblasen, bevor der Ton sich aufbaut.

Noch längere Attack-Werte findet man bei allen Streichinstrumenten. Bedingt durch den "Anstrich", das ist die Art, wie der Spieler mit dem Bogen die Saiten "anstreicht", braucht der Ton vergleichsweise lange, bis er sein Lautstärkemaximum erreicht hat.

Die längsten Attack-Werte haben Geräusche, wie z.B. der Wind, der ja auch in seiner Lautstärke ganz langsam an- und abschwellen kann. Je kürzer der Attack-Wert, desto prägnanter ist auch der sogenannte "Peak" am Anfang eines Tons, durch den sich viele Tasteninstrumente charakterisieren lassen.

Äußerst kurze Attack-Werte, die nur um ein paar Millisekunden liegen, lassen sich nur auf elektronischem Weg erzeugen. Beispiel sind hier die elektronischen Orgeln, wo nach Drücken einer Taste der Ton sofort "da ist".

2. Decay. Dieser Wert bestimmt die Abklingzeit des Tons.

Das englische Wort "decay" steht für Absinken. Der Decay-Wert beeinflußt, wie schnell der Ton in seiner Lautstärke auf ein bestimmtes Niveau absinkt, nachdem er nach der Attack-Phase sein Lautstärkemaximum erreicht hat. Man spricht auch von einer Decay-Phase. Diese Phase kann quasi als das Spiegelbild der Attack-Phase gesehen werden.

Beide Phasen bestimmen den zeitlichen Verlauf von der Lautstärke 0 zum Lautstärkemaximum bis zu einem bestimmten Lautstärkewert, dem Sustain-Wert. Meistens liegt der Fall vor, daß die Decay-Werte länger sind als die Attack-Werte, denn nachdem einmal der Ton angeregt wurde, kann es eine Weile dauern, ehe er wieder leiser wird.

3. Sustain. Dieser Wert bestimmt den Haltepegel des Tons.

Der Sustain-Wert bestimmt die Lautstärke eines Tons nach Ablaufen der Attack- und Decay-Phasen. Man spricht oft von einem "Sustain-Level", zu deutsch "Haltepegel". Wichtig ist, daß man den Sustain-Wert immer im Zusammenhang mit der Gesamtlautstärke sieht. Der Sustain-Level ist nämlich kein absoluter Lautstärkepegel, sondern ein Lautstärkeverhältnis.

Für die Beschreibung des Lautstärkenverlaufs eines Tons durch die ADSR-Werte braucht man neben den Attack- und Decay-Werten, die ja Zeitperioden angeben, die Sustain-Lautstärke und die Gesamtlautstärke. Letztere gibt die Maximallautstärke an, die der Ton nach Ablauf der Attack-Phase und vor Beginn der Decay-Phase hat. Der Sustain-Wert gibt nun an, um wieviel leiser der Ton nach Ablauf der Decay-Phase ist.

Ein Wert von 0 gibt hierbei an, das nach Ablauf der Decay-Phase überhaupt kein Ton mehr zu hören ist. Ein Wert von 8 gibt an, daß nach Ablauf der Decay-Phase der Ton um die Hälfte leiser wurde. Ein Wert von 15 bedeutet, daß nach Ablauf der Attack-Phase der Ton konstant auf dem Pegel der Gesamtlautstärke verbleibt. Der Decay-Wert hat dann keinen Einfluß mehr auf das Lautstärkeverhalten eines Tons.

Wie lange der Ton nun auf der Sustain-Lautstärke verweilt, wird nicht durch die ADSR-Werte gesteuert. Hierfür ist das Gate-Signal verantwortlich. Solange gate=1 ist, bleibt der Ton auf dem Sustainpegel. Ist gate=0, so geht der Ton in die letzte und abschließende Phase, die Release-Phase, über.

Bei den meisten Tasteninstrumenten haben wir einen Sustain-Wert von 0. Grund hierfür ist, daß man durch "Liegenlassen" einer Taste den Ton nicht auf einer bestimmten Lautstärke verweilen lassen kann. Der Ton schwingt ein (Attack) und klingt aus (Decay), und fertig.

Ausnahme sind die elektronischen Orgeln. Hier haben wir einen Sustainwert von 15, alle anderen Werte sind 0. Grund hierfür ist, daß bei der elektronischen Orgel keine Attack- oder Decay-Phasen vorliegen. Ist gate=1, befindet sich der Ton sofort auf dem Sustain-Pegel, der mit der Gesamtlautstärke identisch ist. Ist gate=0, so ist der Ton sofort weg. Es gibt also keine Lautstärkeabstufungen im Tonverlauf, man kann immer nur die Gesamtlautstärke mit einem Lautstärkepedal ändern.

4. Release. Dieser Wert bestimmt die Ausklingzeit des Tons.

Der Release-Wert bestimmt, wie lange nach Löschen des Gate-Signals ein Ton braucht, vom Sustain-Lautstärkepegel auf die Lautstärke 0 abzufallen. Im Beispielprogramm bedeutet ein großer Release-Wert, daß der Ton nach Loslassen der Taste noch lange nachklingt (engl. release = "loslassen"). Bei den meisten natürlichen Instrumenten sind die Release-Werte ziemlich kurz, da nur sehr wenige Instrumente nach Beendigung der Tonerzeugung noch lange weiterklingen.

4.9 Tabelle der Attack-, Decay- und Release-Zeiten

Aus der nun folgenden Tabelle können Sie entnehmen, welche Attack-, Decay- und Release-Zeiten Sie mit dem Sound-Chip erzeugen können. "ms" steht hier für Millisekunden (tausendstel Sekunden), "s" für Sekunden.

Wert	Attack-Zeit	Decay-Zeit	Release-Zeit
0	2 ms	6 ms	6 ms
1	8 ms	24 ms	24 ms
2	16 ms	48 ms	48 ms
3	24 ms	72 ms	72 ms
4	38 ms	114 ms	114 ms
5	56 ms	168 ms	168 ms
6	68 ms	204 ms	204 ms
7	80 ms	240 ms	240 ms

Wert	Attack-Zeit	Decay-Zeit	Release-Zeit
8	100 ms	300 ms	300 ms
9	250 ms	750 ms	750 ms
10	500 ms	1,5 s	1,5 s
11	800 ms	2,4 s	2,4 s
12	1 s	3,0 s	3,0 s
13	3 s	9,0 s	9,0 s
14	5 s	15,0 s	15,0 s
15	8 s	24,0 s	24,0 s

4.10 Zusammenfassung zu den Oszillatoren des Sound-Chips

Bisher haben wir immer nur einen Ton gleichzeitig gespielt, der Sound-Chip kann aber bis zu drei Töne gleichzeitig produzieren. Nun sind aber die Einstellungen für alle drei Töne gleich. Für jeden Ton gibt es die gleiche Zahl von Angestellten, nämlich 52. Diese 52 Angestellten sind über 7 Zimmer verteilt. Jeweils sieben Zimmer bilden also eine Gruppe von Zimmern, die zusammengehören. Davon gibt es drei Stück.

Jede solche Gruppe faßt man unter dem Begriff "Oszillator" zusammen. Wenn wir von nun an von "Oszillatoren" sprechen, ist stets eine dieser Sieben-Zimmer-Gruppen gemeint. Diese Zimmergruppen liegen im Sound-Chip-"Büro" alle hintereinander, so daß wir nun 3 * 7 = 21 der 29 Zimmer bereits kennen. In der nun folgenden Tabelle sind sie alle einmal zusammengestellt.

Nr.	Oszillator 1
0	Frequenz Low
1	Frequenz High
2	Pulsbreite Low
3	Pulsbreite High
4	Wellenform Kontrollbits
5	Attack-Wert Decay-Wert
6	Sustain-Wert Release-Wert

Nr.	Oszillator 2	
7	Frequenz Low	
8	Frequenz High	
9	Pulsbreite Low	
10	Pulsbreite High	
11	Wellenform	Kontrollbits
12	Attack-Wert	Decay-Wert
13	Sustain-Wert	Release-Wert

Nr.	Oszillator 3	
14	Frequenz Low	
15	Frequenz High	
16	Pulsbreite Low	
17	Pulsbreite High	
18	Wellenform	Kontrollbits
19	Attack-Wert	Decay-Wert
20	Sustain-Wert	Release-Wert

Wir sehen also, daß die Oszillatoren in ihrer Funktion gleichwertig sind. Oszillator 1 belegt die Zimmernummern 0 bis 6, Oszillator 2 7 bis 13 und Oszillator 3 14 bis 20.

In der Tabelle taucht auch ein neuer Begriff auf, es werden "Kontrollbits" erwähnt. Hier haben wir einen Begriff aus der Fachterminologie vor uns, aber im Grund kennen wir ihn schon. Statt von "Kontrollbits" sprachen wir immer von Angestellten. Eins dieser Kontrollbits haben wir auch schon kennengelernt, nämlich das Gate-Signal. Die übrigen Kontrollbits werden wir uns später anschauen.

Da gerade die Rede von der Fachterminologie ist: Was wir die ganze Zeit "Zimmer" des Sound-Chip-"Büros" nennen, heißt dort "Kontrollregister" oder einfach nur "Register". Genauso wie wir ein Zimmer mit den darin befindlichen Angestellten identifiziert haben, so kann man sich ein "Register" als Gruppe verschiedener Kontrollbits denken. Vorerst wollen wir die beiden Bezeichnungsweisen noch nebeneinander verwenden.

4.11 Der Filter im Sound-Chip: Prinzip und Einstellungen

Der Sound-Chip hat neben den drei Oszillatoren noch einen Filter. Es ist nicht schwierig, das Prinzip eines Filters zu verstehen. Schauen Sie sich einmal die Tonregler an Ihrer Stereoanlage an. Es gibt da einen Regler, der die Bässe anheben oder absenken kann, und einen Regler, der Höhen anheben oder absenken kann. Diese Tonregler sind Filter, die jedoch immer nur eine bestimmte Frequenz anheben oder absenken können. Beim Filter des Sound-Chips kann man jedoch genau angeben, welche Frequenzen man verändern will.

Die Wellenformen, die vom Sound-Chip erzeugt werden, haben sehr viele Obertöne. Deswegen klingen diese Wellenformen oft sehr grell und höhenreich. Um dem Klang natürlicher Instrumente näherzukommen, gibt es die Möglichkeit, Obertöne dieser Wellenformen "abzukappen" und somit den Klang weicher zu gestalten. Das geschieht, indem man die Wellenform, die aus dem Oszillator herauskommt, durch den Filter leitet. Dieser Filter läßt die tiefen Frequenzen passieren und schneidet die hohen Frequenzen ab. Deswegen nennt man diesen Filtertyp auch "Tiefpass"-Filter.

Man kann den Filter aber auch so einstellen, daß er genau umgekehrt arbeitet, nämlich die hohen Frequenzen passieren läßt und die tiefen abschneidet. Diesen Filtertyp nennt man "Hochpass"-Filter. Die Frequenz, ab der Obertöne "abgekappt" werden sollen, kann man bei beiden Filtertypen genau einstellen.

Der dritte Filtertyp, den man beim Sound-Chip einstellen kann, entfernt sowohl tiefe als auch hohe Frequenzen aus dem Frequenzspektrum und läßt ein Frequenz-"Band" in der Mitte passieren. Deswegen heißt dieser Filtertyp auch "Bandpass"-Filter. Die Frequenz des Filters bedeutet bei einem Bandpass-Filter, an welcher Stelle des Frequenzspektrums die meisten Frequenzen durchgelassen werden.

Neben Filterfrequenz und Filtertyp läßt sich beim Filter des Sound-Chips noch die Filterresonanz einstellen. Um die Bedeutung dieses Parameters zu verstehen, sollte man sich den Filter als vierten Oszillator im Sound-Chip vorstellen. Beim Filter kann man ja auch eine Frequenz einstellen wie bei den übrigen drei Oszillatoren.

Durch den Resonanz-Wert kann man eingeben, in welchem Grad der Filter selbst als Oszillator wirkt. Wenn die Resonanz auf 0 gestellt wird, tut der Filter nichts, als Frequenzen abzuschneiden. Wenn man den Resonanz-Wert nun Schritt für Schritt erhöht, fängt der Filter bei der Filterfrequenz an, immer mehr selbst zu schwingen.

Der Maximalwert der Filterresonanz ist 15. Der Klang der Oszillatoren, die durch den Filter geleitet werden, wird dann stark verändert und durch die Filterfrequenz beeinflußt.

Man sieht hier leicht, daß man durch den Filter ein unheimliches Spektrum an Klängen hinzugewinnt, da die normalen Wellenformen der Oszillatoren auf mannigfaltige Art und Weise manipuliert werden können.

Aus der nun folgenden Tabelle können Sie ersehen, welche Register des Sound-Chips den Filter beeinflussen. Diese Register haben für jede oben beschriebene Funktion Kontrollbits.

Nr.	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
21	.....					freq 2	freq 1	freq 0
22	freq10	freq 9	freq 8	freq 7	freq 6	freq 5	freq 4	freq 3
23	res 3	res 2	res 1	res 0	FILTEX	FILT 3	FILT 2	FILT 1
24	3 AUS	Hochp.	Bandp.	Tiefp.	VOL 3	VOL 2	VOL 1	VOL 0

Hier haben wir alle Register ("Zimmer" des Sound-Chip-"Büros") vor uns, die Kontrollbits ("Angestellte") enthalten, die etwas mit dem Filter zu tun haben. In Register 21 und 22 haben wir die Bits, die die Filter-Grenzfrequenz, also die Frequenz, die bestimmt, ab welcher der Filter anfängt zu arbeiten (also Obertöne abzuschneiden). Da die Filterfrequenz nur 11 Bit breit ist (man kann also nur Werte zwischen 0 und 2047 einstellen), sind einige Bits in Register 21 nicht definiert.

Den gewünschten Filtertyp kann man durch die Bits 6, 5, und 4 in Register 24 anwählen. Diese Bits lassen sich getrennt setzen und löschen, so daß es möglich ist, die zur Verfügung stehenden Filtertypen beliebig zu mischen. Es sind folgende acht Kombinationen möglich:

Bit 6	Bit 5	Bit 4	resultierender Filtertyp
0	0	0	Filter-Sperre
0	0	1	Tiefpass
0	1	0	Bandpass
1	0	0	Hochpass
0	1	1	Tiefpass & Bandpass
1	0	1	Tiefpass & Hochpass
1	1	0	Hochpass & Bandpass
1	1	1	All-Pass

Die Filtertypen, die sich durch Mischen der drei Grund-Filtertypen ergeben, lassen wesentlich mehr Frequenzen passieren als die Grund-Filtertypen. Die Filter-Grenzfrequenz hat deswegen keine so entscheidende Bedeutung mehr für den Klang wie bei den Grund-Filtertypen.

"Filter-Sperre" bedeutet, daß wenn alle drei Bits auf "0" sitzen, der Ton derjenigen Oszillatoren, die durch den Filter geleitet werden, nicht mehr zu hören ist. Er wird vom Filter "gesperrt", der Filter läßt dann keinen Ton mehr durch.

"All-Pass" ist die Mischung aller drei Filtertypen. Hier läßt der Filter sämtliche Frequenzen durch, ohne welche abzuschneiden. Wie bei der Filter-Sperre hat hier die Grenzfrequenz keine Bedeutung mehr.

Mit den vier "res"-Bits aus Register 23 läßt sich die Filterresonanz von 0 bis 15 variieren. Die übrigen Bits aus Register 13 bestimmen, welche Oszillatoren durch den Filter geleitet werden. "FILTEX" bedeutet hier die (theoretische) Möglichkeit, das "Audio In"-Signal durch den Sound-Chip zu leiten und zu filtern. Hierzu ist es allerdings notwendig, die Hardware des Sound-Chips genauer zu kennen.

In Register 24 finden wir noch einige Bits, die nichts mit dem Filter zu tun haben: die uns wohlbekannten Lautstärke-Kontrollbits und das Bit "3 AUS", auf das wir im fünften Kapitel eingehen werden.

4.12 Synchronisation und Ring-Modulation

Der Filter des Sound-Chips ermöglichte es uns, das Signal einzelner Oszillatoren gezielt zu verändern, indem wir eine bestimmte Anzahl von Obertönen aus dem Frequenzspektrum entfernt haben. Im Sound-Chip gibt es noch zwei weitere Möglichkeiten, das Signal, das aus einem Oszillator kommt, drastisch zu verfremden: Synchronisation und Ring-Modulation.

Während beim Filter immer nur das Signal eines einzelnen Oszillators verändert wurde, bieten die Synchronisation und die Ring-Modulation uns die Möglichkeit, in Abhängigkeit vom Signal zweier Oszillatoren das Signal eines der beiden Oszillatoren oder auch beider zu verändern. Man muß sich das so vorstellen, daß ein Oszillator als Tonquelle wie bisher wirkt, aber sein Signal durch das Signal eines anderen Oszillators beeinflußt wird. Im folgenden wird der Oszillator, der beeinflußt wird, der Basisoszillator, und der Oszillator, der selbst beeinflussend wirkt, der Kontrolloszillator genannt.

Die Synchronisation basiert darauf, daß das Signal des Basisoszillators wie des Kontrolloszillators aus einzelnen Wellen besteht. Diese beiden Wellen können in Abhängigkeit von der Frequenz und der Wellenform ganz unterschiedliche Form und Gestalt haben.

Wenn der Basisoszillator vom Kontrolloszillator synchronisiert wird, so wirkt der Kontrolloszillator wie ein Start-/Stop-Schalter für den Basisoszillator. Jedesmal, wenn der Kontrolloszillator eine Welle beendet hat und eine neue Welle beginnt, gibt er sein Start-Signal für eine neue Schwingung an den Basisoszillator weiter. Also immer, wenn der Kontrolloszillator "startet", wird der Basisoszillator "mitgestartet".

Wenn beide Oszillatoren die gleiche Wellenform und Frequenz haben, so laufen beide exakt synchron. Liegt aber bei einem Oszillator eine andere Frequenz oder gar Wellenform vor als bei anderen, so bewirkt das "Start"-Signal beim Basis-Oszillator, daß dieser mitten in seinen Schwingungen plötzlich durch das "Start"-Signal des Kontrolloszillators gezwungen wird, die Welle von neuem zu beginnen.

Dadurch beendet der Basisoszillator nie seine Schwingungen vollständig, sondern nur zu einem Teil und fängt dann immer wieder von vorne an, da die Schwingungen beider Oszillatoren immer unterschiedlich lange dauern.

Aus diesem recht komplexen Prinzip ergeben sich dann aber auch die kuriosesten Wellenformen. Da jede Frequenz und Wellenform des Kontrolloszillators eine andere Wirkung hat, hat man eine fast unbegrenzte Anzahl von Möglichkeiten.

Bei der Ring-Modulation werden die digitalen Zahlenwerte der Schwingungen von Kontrolloszillator und Basisoszillator innerhalb des Sound-Chips miteinander multipliziert und durch den Basisoszillator ausgegeben.

Wenn die Frequenzen beider Oszillatoren nahe beieinander liegen, entstehen sehr komplexe Wellenformen, die viele nicht-harmonische Obertöne enthalten und deshalb oft metallischen oder glockenähnlichen Charakter haben.

Nach dieser theoretischen Beschreibung beider Effekte wollen wir uns anschauen, welche Kontrollbits dafür zuständig sind.

Bit 3	Bit 2	Bit 1	Bit 0
Test-Bit	Ring-Mod.	Sync.	Gate

Diese vier Bits findet man bei allen drei Oszillatoren. Da man die Kontrollbits für Synchronisation und Ring-Modulation für jeden Oszillator getrennt setzen und löschen kann, gibt es insgesamt jeweils acht Kombinationen für die Synchronisation und die Ring-Modulation, die Sie folgender Tabelle entnehmen können:

Osz.1	Osz.2	Osz.3	Ergebnis:
0	0	0	keine Synchronisation
1	0	0	Osz. 1 wird von Osz. 3 synchronisiert
0	1	0	Osz. 2 wird von Osz. 1 synchronisiert
0	0	1	Osz. 3 wird von Osz. 2 synchronisiert
1	1	0	Osz. 1 & Osz. 2 synchr. durch Osz. 3
0	1	1	Osz. 2 & Osz. 3 synchr. durch Osz. 1
1	0	1	Osz. 1 & Osz. 3 synchr. durch Osz. 2
1	1	1	alle Oszillatoren laufen synchron

Osz.1	Osz.2	Osz.3	Ergebnis:
0	0	0	keine Ring-Modulation
1	0	0	Osz. 1 wird von Osz. 3 ringmoduliert
0	1	0	Osz. 2 wird von Osz. 1 ringmoduliert
0	0	1	Osz. 3 wird von Osz. 2 ringmoduliert
1	1	0	Osz. 1 & Osz. 2 ringmod. durch Osz. 3
0	1	1	Osz. 2 & Osz. 3 ringmod. durch Osz. 1
1	0	1	Osz. 1 & Osz. 3 ringmod. durch Osz. 2
1	1	1	alle Oszillatoren modulieren einander

Das oben genannte "Test-Bit" ist für jeden Oszillator getrennt vorhanden. Ist test=0, so funktioniert alles normal. Ist test=1, so ist der betreffende Oszillator ausgeschaltet. Dieses Bit kann vom Benutzer gesetzt werden, um z.B. einen unbenötigten Oszillator auszuschalten.

Das Test-Bit wird vom Sound-Chip auf 1 gesetzt, falls versucht wurde, bei der Einstellung der Wellenform die Wellenform "Rauschen" mit einer anderen Wellenform zu kombinieren. Das kann der Sound-Chip nicht, der betreffende Oszillator "hängt" sich dann "auf" und das Test-Bit wird auf 1 gesetzt. Setzt der Benutzer das Bit wieder auf 0, funktioniert alles wie gewohnt.

Sie haben nun alle Register des Sound-Chips einmal kennengelernt. Auf den nachfolgenden Seiten finden Sie noch zwei Programme, eins zur Demonstration des Filters und eines, was einen Ring-Modulator-Effekt produziert, sowie eine Registertabelle, in der alle Register des Sound-Chips noch einmal zusammengefaßt dargestellt sind. Danach kommen zum Abschluß des Kapitels Informationen für Fortgeschrittene zu den Registern.

```
0 REM DIES IST DIE VOLLSTAENDIGE FASSUNG
1 REM DES BEISPIELPROGRAMMS
2 REM MIT ADSR-WERT-EINSTELLUNG
3 REM UND ALLEN FILTEREINSTELLUNGEN
5 WELLE = 64
6 REM WELLENFORM
7 ERSTER = 54272
8 POKE ERSTER + 3,8
9 REM PULSBREITE
10 LAUTST = ERSTER + 24
20 DREUCK = ERSTER + 4
40 INPUT "FREQUENZ";F
50 F2=F/256
60 F1=F-INT(F2)*256
70 POKE ERSTER+1,F2
80 INPUT "A,D,S,R";A,D,S,R
90 POKE ERSTER + 5, A*16 + D
100 POKE ERSTER + 6, S*16 + R
110 INPUT "FILTER AUS (0) ODER AN (1)";P
120 FILT = 15
130 IF P=0 THEN 270
140 INPUT "HOCHPASS (0/1)";P
150 IF P THEN FILT=FILT OR 64
160 INPUT "BANDPASS (0/1)";P
170 IF P THEN FILT=FILT OR 32
180 INPUT "TIEFPASS (0/1)";P
190 IF P THEN FILT=FILT OR 16
200 INPUT "FREQUENZ (0-2047)";FRQ
210 FH=INT (FRQ/8)
220 FL=FRQ-FH*8
230 POKE ERSTER + 21,FL
240 POKE ERSTER + 22,FH
250 INPUT "RESONANZ (0-15)";RES
260 RES=RES*16 + 1
270 POKE ERSTER + 23,RES
```

```
280 POKE ERSTER + 24,FILT
300 IF PEEK (197)=64 THEN 330
310 POKE DRUECK, WELLE + 1
320 GOTO 300
330 POKE DRUECK, WELLE
340 GOTO 300

READY.
```

```
 0 ERSTER = 54272
 10 FOR I=0 TO 7
 20 POKE ERSTER + 24, 15
 30 POKE ERSTER + 5, 0
 40 POKE ERSTER + 6, 240
 50 POKE ERSTER + 4, 16 + 4 + 1
 60 READ A:IF A=0 THEN 500
 70 A=A/2↑I
 80 L=INT(A/256):H=INT(A-(256*L))
 90 IF TI<TM+(8-I)*2 THEN 90
 100 TM = TI
 110 POKE ERSTER + 1,L
 120 POKE ERSTER + 15,H
 130 GOTO 50
 500 RESTORE
 510 NEXT
 520 FOR C=1 TO 1000:NEXT
 530 POKE ERSTER + 24,0
 900 DATA 64814,61176,57743,54502,51443,48556,45830,43258,40830,3853
 910 DATA 36376,34334
 920 DATA 0

 READY.
```

4.13 Registertabelle

Nr.	Dez.	Hex.	Funktion	Bit 7	Bit 6	Bit 5	Bit 4
0	54272	$D400	Frequenz Low	f1 7	f1 6	f1 5	f1 4
1	54273	$D401	Frequenz High	f1 15	f1 14	f1 13	f1 12
2	54274	$D402	Pulsbreite Low	p1 7	p1 6	p1 5	p1 4
3	54275	$D403	Pulsbreite High	.....	.....	.....	.....
4	54276	$D404	Wellenform	Rausch	Puls	Sägez.	Dreieck
5	54277	$D405	Attack/Decay	a1 3	a1 2	a1 1	a1 0
6	54278	$D406	Sustain/Release	s1 3	s1 2	s1 1	s1 0
7	54279	$D407	Frequenz Low	f2 7	f2 6	f2 5	f2 4
8	54280	$D408	Frequenz High	f2 15	f2 14	f2 13	f2 12
9	54281	$D409	Pulsbreite Low	p2 7	p2 6	p2 5	p2 4
10	54282	$D40A	Pulsbreite High	.....	.....	.....	.....
11	54283	$D40B	Wellenform	Rausch	Puls	Sägez.	Dreieck
12	54284	$D40C	Attack/Decay	a2 3	a2 2	a2 1	a2 0
13	54285	$D40D	Sustain/Release	s2 3	s2 2	s2 1	s2 0
14	54286	$D40E	Frequenz Low	f3 7	f3 6	f3 5	f3 4
15	54287	$D40F	Frequenz High	f3 15	f3 14	f3 13	f3 12
16	54288	$D410	Pulsbreite Low	p3 7	p3 6	p3 5	p3 4
17	54289	$D411	Pulsbreite High	.....	.....	.....	.....
18	54290	$D412	Wellenform	Rausch	Puls	Sägez.	Dreieck
19	54291	$D413	Attack/Decay	a3 3	a3 2	a3 1	a3 0
20	54292	$D414	Sustain/Release	s3 3	s3 2	s3 1	s3 0
21	54293	$D415	Filterfreq. Low	.....	.....	.....	.....
22	54294	$D416	Filterfreq. High	fc 10	fc 9	fc 8	fc 7
23	54295	$D417	Filterresonanz	RES 3	RES 2	RES 1	RES 0
24	54296	$D418	Lautstärke u.a.	3 AUS	Hochp.	Bandp.	Tiefp
25	54297	$D419	Potentiometer 1	7	6	5	4
26	54298	$D41A	Potentiometer 2	7	6	5	4
27	54299	$D41B	Osz. 3 High Byte	f3 15	f3 14	f3 13	f3 12
28	54300	$D41C	Env. 3 High Byte	e3 7	e3 6	e3 5	e3 4

Nr.	Dez.	Hex.	Funktion	Bit 3	Bit 2	Bit 1	Bit 0
0	54272	$D400	Frequenz Low	f1 3	f1 2	f1 1	f1 0
1	54273	$D401	Frequenz High	f1 11	f1 10	f1 9	f1 8
2	54274	$D402	Pulsbreite Low	p1 3	p1 2	p1 1	p1 0
3	54275	$D403	Pulsbreite High	p1 11	p1 10	p1 9	p1 8
4	54276	$D404	Wellenform	Test	R.Mod.	Sync.	Gate
5	54277	$D405	Attack/Decay	d1 3	d1 2	d1 1	d1 0
6	54278	$D406	Sustain/Release	r1 3	r1 2	r1 1	r1 0
7	54279	$D407	Frequenz Low	f2 3	f2 2	f2 1	f2 0
8	54280	$D408	Frequenz High	f2 11	f2 10	f2 9	f2 8
9	54281	$D409	Pulsbreite Low	p2 3	p2 2	p2 1	p2 0
10	54282	$D40A	Pulsbreite High	p2 11	p2 10	p2 9	p2 8
11	54283	$D40B	Wellenform	Test	R.Mod.	Sync.	Gate
12	54284	$D40C	Attack/Decay	d2 3	d2 2	d2 1	d2 0
13	54285	$D40D	Sustain/Release	r2 3	r2 2	r2 1	r2 0
14	54286	$D40E	Frequenz Low	f3 3	f3 2	f3 1	f3 0
15	54287	$D40F	Frequenz High	f3 11	f3 10	f3 9	f3 8
16	54288	$D410	Pulsbreite Low	p3 3	p3 2	p3 1	p3 0
17	54289	$D411	Pulsbreite High	p3 11	p3 10	p3 9	p3 8
18	54290	$D412	Wellenform	Test	R.Mod.	Sync.	Gate
19	54291	$D413	Attack/Decay	d3 3	d3 2	d3 1	d3 0
20	54292	$D414	Sustain/Release	r3 3	r3 2	r3 1	r3 0
21	54293	$D415	Filterfreq. Low	.....	fc 2	fc 1	fc 0
22	54294	$D416	Filterfreq. High	fc 6	fc 5	fc 4	fc 3
23	54295	$D417	Filterresonanz	FILTEX	FILT3	FILT2	FILT1
24	54296	$D418	Lautstärke u.a.	VOL 3	VOL 2	VOL 1	VOL 0
25	54297	$D419	Potentiometer 1	3	2	1	0
26	54298	$D41A	Potentiometer 2	3	2	1	0
27	54299	$D41B	Osz. 3 High Byte	f3 11	f3 10	f3 9	f3 8
28	54300	$D41C	Env. 3 High Byte	e3 3	e3 2	e3 1	e3 0

4.14 Erläuterungen zur Registertabelle

Aus Platzgründen wurde die Tabelle in zwei Hälften aufgeteilt. Im ersten Teil werden die Bits 7 bis 4, im zweiten Teil die Bits 3 bis 0 der entsprechenden Register genannt.

Die Register 0-6, 7-13 und 14-20 sind in ihrer Funktion gleichwertig. Sie sprechen jeweils den ersten, zweiten bzw. dritten Oszillator an.

Bei den Registern 3, 10 und 17, die die oberen vier Bit der Pulsbreite steuern, sind die Bits 7-4 nicht definiert. Ebenso sind bei Register 21, das die niederen drei Bit der Filterfrequenz steuert, die oberen fünf Bit nicht definiert. Die nicht definierten Bits sind durch Punkte gekennzeichnet.

Die Register 0-24 sind Schreib-Register. Man kann also nur hineinPOKEn, PEEKen erzeugt undefinierte Resultate. Man kann also nicht mehr einen Wert, der einmal hineingePOKEt wurde, durch PEEK lesen. Es kommt immer 0 heraus. Die Register 25-28 sind Lese-Register. Diese können nur gePEEKt werden. Mit Register 24 und 25 werden die beiden AD-Wandler abgefragt (z.B. für Paddles). Register 27 liest die oberen acht Bit der Frequenz des dritten Oszillators, Register 28 liest die oberen acht Bit der Hüllkurve von Oszillator 3. Diese Register können für Modulationszwecke benutzt werden. Ein Beispiel hierfür finden Sie im fünften Kapitel.

4.15 Anmerkungen und Programme für Fortgeschrittene

Da es ziemlich schwierig ist, das Bus-Konzept und den Anschluß von I/O-Bausteinen an den Prozessor zu verstehen, sollen erst jetzt einige Programme und Anmerkungen kommen, die sich mit den Registern des Sound-Chips und ihrer hardwaremäßigen Struktur intensiver auseinandersetzen.

Beim Konzept des "memory mapped I/O", also der Möglichkeit, I/O-Zellen als Speicherzellen zu adressieren, ist beim Commodore 64 etwas übertrieben worden. Vier I/O-Bausteine liegen im Adressraum von $D000 bis $DFFF verteilt.

Die Zugriffszeiten dieser vier Chips unter einen Hut zu bringen, ist schon ein Kunststück. Diese vier Chips - zwei 6526, der Video-Chip 6569 und der Sound-Chip 6581 - müssen alle irgendwie auf den "ihnen zugewiesenen" Speicherplatz zugreifen und dürfen dabei dem Prozessor nicht ins Gehege kommen. Und irgendwie klappt das auch - nur mit einigen "Macken". Viele werden sich schon über Fehler des Video-Chips bei den Sprites oder beim Zeichensatz geärgert haben.

Beim Sound-Chip liegt das Manko darin, daß die Register nur als Schreib-Register ausgelegt sind - wenn man PEEKt, steht nicht das drin, was reingePOKEt wurde. Um dieses Manko zu beseitigen, kann man das "Read-Write Register Interrupt"-Programm benutzen (Listings siehe folgende Seiten).

Wenn Sie den Basic-Loader benutzen, müssen Sie nach Eingeben dieses Loaders und RUN den entsprechenden SYS tippen, um den Interrupt zu aktivieren. Wenn Sie in Assembler programmieren, können Sie mit dem Quelltext arbeiten. Achten Sie bitte auf die Hinweise für Assembler-Programme im fünften Kapitel.

Dieses Programm besteht aus einem Interrupt, der in Page 2 von $02C0 bis $02D8 eine exakte Kopie der 25 Write-Only-Register anlegt und sie per Interrupt nach $D400 bis $D418 herüberkopiert. Wird dieser Interrupt benutzt, kann man diese Adressen als quasi Read-Write Register verwenden. Im Basic-Beispielprogramm schreibt man dann einfach statt "ERSTER = 54272" (dez. 54272 = $D400) "ERSTER = 704" (dez. 704 = $02C0).

```
0 OPEN 4,4
1 SYS 49152
2 .OPT OO,P4
3 *=$033C
4 IRQ = $0314
5 BUF = $02C0
6 SID = $D400
10 ENABLE SEI
11 :       LDA #<IRQENTRY
12 :       STA IRQ
13 :       LDA #>IRQENTRY
14 :       STA IRQ+1
15 :       CLI
16 EXIT    RTS
20 IRQENTRY SEI
21 :         LDX #0
22 :         LDA BUF,X
23 :         STA SID,X
24 :         INX
25 :         CPX #$19
26 :         BCC IRQENTRY+3
30 :         JMP $EA31
40 .END

READY.
```

```
0 A=828
1 READ P:IF P<0 THEN SYS 828:END
2 POKE A,P:A=A+1:GOTO 1
10 DATA 120,169, 73,141, 20,  3,169
11 DATA   3,141, 21,  3, 88, 96,120
12 DATA 162,  0,189,192,  2,157,  0
13 DATA 212,232,224, 25,144,245, 76
14 DATA  49,234, -1

READY.
```

Eine interessante Anwendung des Sound-Chips in Verbindung mit einem 6526 ist das Übertragen des Signals vom Cassetten-Recorder über den Sound-Chip zum Lautsprecher des Fernsehers.

Ausgegangen wird hierbei davon, daß das Rechtecksignal des Cassettenrecorders in Bit 4 vom Control Register 13 des ersten I/O-Bausteins ("CR 13 von CIA #1") als Impuls übergeben wird. Diesen Impuls muß man nur (natürlich in Assembler) in einen Impuls oder "Knacks" des Sound-Chips umsetzen.

Um diesen Impuls zu erzeugen, bieten sich mehrere Möglichkeiten an. Die erste wurde im Programm "Cassetten-Abhören Nr.1" verwendet. Hier wird, sobald ein Impuls auftritt, die Gesamtlautstärke auf 15 gesetzt, was, wir gesehen haben, einen Knacks im Lautsprecher erzeugt. Dann wartet das Programm eine gewisse Zeit und setzt dann die Lautstärke auf 0 zurück.

Eine andere Möglichkeit, einen Knacks zu erzeugen, liegt in der Benutzung des Gate-Signals, wie das im Programm "Cassetten-Abhören Nr.2" getan wurde. Der Impuls, der beim Triggern des Gate-Bits entsteht, ist jedoch leiser als der Lautstärkeknacks. Eine andere Möglichkeit wäre noch, den Knacks auszunutzen, der entsteht, wenn man Oszillatoren zum Filter leitet oder wieder davon trennt.

Wenn Sie eines der beiden Programme eingegeben und mit dem entsprechenden SYS gestartet haben, kann man in eine angeschlossene Datasette eine Cassette einlegen, auf PLAY drücken und die Aufnahme abhören. Zum Abstoppen muß man mit RUN/STOP RESTORE einen NMI auslösen, um in Basic zurückzugelangen.

Qualitätsmäßig darf man jedoch von beiden Programmen nicht viel erwarten. Aus einem Rechtecksignal kann man allerhöchstens noch Gesprochenenes erkennen, bei Musik versteht man nicht mehr viel.

```
0 OPEN 4,4
1 SYS 49152
2 .OPT OO,P4
3 *=$033C
4 IRQ = $0314
5 BUF = $02C0
6 SID = $D400
10 ENABLE SEI
11 :       LDA #<IRQENTRY
12 :       STA IRQ
13 :       LDA #>IRQENTRY
14 :       STA IRQ+1
15 :       CLI
16 EXIT    RTS
20 IRQENTRY SEI
21 :         LDX #0
22 :         LDA BUF,X
23 :         STA SID,X
24 :         INX
25 :         CPX #$19
26 :         BCC IRQENTRY+3
30 :         JMP $EA31
40 .END

READY.
```

```
0 A=828
1 READ P:IF P<0 THEN SYS 828:END
2 POKE A,P:A=A+1:GOTO 1
10 DATA 120,169, 73,141, 20,  3,169
11 DATA   3,141, 21,  3, 88, 96,120
12 DATA 162,  0,189,192,  2,157,  0
13 DATA 212,232,224, 25,144,245, 76
14 DATA  49,234, -1

READY.
```

```
0 OPEN 1,4
1 SYS 49152
2 .OPT OO,P
3 *=$33C
9 ;**** ADRESSEN
10 CASS   = $DC0D
11 SIDREG = $D400
12 REG1   = SIDREG+4
13 REG2   = SIDREG+11
14 REG3   = SIDREG+18
15 LAUTST = SIDREG+24
16 ;**** KONSTANTEN
17 WARTE  = $F1
18 WELLE  = $21
19 ;**** PROGRAMMANFANG
20 LDA #15:STA LAUTST
28 ;
29 ;**** WARTEN AUF CASSETTEN-IMPULS
30 SCHLEIFE LDA CASS       ;CASSETTENINPUT LESEN
31 ^        CMP #%00010000 ;LESE-BITMUSTER
32 ^        BNE SCHLEIFE   ;ABFRAGE, OB IMPULS
38 ;
39 ;**** IMPULS EIN
40 LDA #WELLE:STA REG1
41 LDA #WELLE:STA REG2
42 LDA #WELLE:STA REG3
48 ;
49 ;**** WARTESCHLEIFE
50 ^        LDX #0
51 ZAEHLEN  DEX
52 ^        CPX #WARTE
53 ^        BNE ZAEHLEN
58 ;
59 ;**** IMPULS AUS
60 LDA #0:STA REG1
61 LDA #0:STA REG2
62 LDA #0:STA REG3
```

```
68 ;
69 ;**** ENDLOSSCHLEIFE
70 JMP SCHLEIFE

READY.
```

```
0 A=828
1 READ P:IF P<0 THEN SYS 828
2 POKE A,P:A=A+1:GOTO 1
10 DATA 169, 15,141, 24,212,173, 13
11 DATA 220,201, 16,208,249,169, 33
12 DATA 141,  4,212,169, 33,141, 11
13 DATA 212,169, 33,141, 18,212,162
14 DATA   0,202,224,241,208,251,169
15 DATA   0,141,  4,212,169,  0,141
16 DATA  11,212,169,  0,141, 18,212
17 DATA  76, 65,  3, -1

READY.
```

5. Fortgeschrittene Musikprogrammierung des Commodore 64

5.1 Das Counter-Prinzip

Ein Musikprogramm - **gleich welcher Art** - besteht aus zwei Teilen: dem eigentlichen Programm und den zugehörigen Daten.

Das eigentliche Programm ist meist im Vergleich zu den Daten ziemlich kurz: Es hat ja keine andere Aufgabe, als einen Ton aus den Daten zu lesen, die Tonhöhe in die entsprechenden Register zu schreiben und eine bestimmte Zeit zu warten. Wie man ein solches Programm entwerfen kann, wollen wir im folgenden analysieren.

Grundsätzlich lassen sich diese Programm in zwei Klassen unterteilen: in einstimmige und mehrstimmige. Die einstimmigen Musikprogramme haben es sehr einfach. Sie brauchen immer nur einen einzelnen Ton zu lesen und dann zu warten. Diese Warteschleife liegt hierbei außerhalb der eigentlichen Spielroutine.

Ganz anders ist das bei mehrstimmigen Musikprogrammen. Hier tritt das Problem auf, daß die Töne, die die einzelnen Stimmen zu spielen haben, unterschiedlich lang sein können.

Deswegen müssen hier Spielroutine und Warteschleife miteinander verquickt werden, und hier begegnet uns das Prinzip des Counters. Jede Stimme muß ihren eigenen Counter haben, da jede Stimme unabhängig von der anderen laufen muß. Dieser Counter kann entweder ein "Dekrement"- oder ein "Inkrement"-Counter sein.

Beim "Dekrement"-Counter wird jedesmal, wenn eine neue Note gespielt wird, die Notendauer in den Counter übertragen. Nun wird bei jedem Durchlaufen der Spielroutine der Counter dekrementiert (um 1 erniedrigt). Irgendwann erreicht der Counter durch das laufende Herunterzählen den Wert 0. Sobald der Counter auf 0 steht, wird eine neue Note gespielt und auch der Counter neu gesetzt.

Im Beispielprogramm "Michelle" können wir diesen Vorgang deutlich verfolgen. In diesem Programm wird während des Programmablaufs der aktuelle Wert der beiden Counter auf den Bildschirm gePRINTet.

Man kann deutlich erkennen, daß zu Beginn jeder Note dem Counter ein bestimmter Wert zugewiesen wird, der dann wie bei einem "Countdown" immer um 1 dekrementiert wird, bis er 0 erreicht. Sobald die 0 erscheint, wird eine neue Note gespielt, und das ganze geht von vorne los. Wichtig ist hierbei zu beobachten, daß die Counter bei beiden Stimmen unabhängig laufen.

Anmerkungen zum Programm: Aus Platzgründen wurde hier auf die komfortablere Art, die Noten durch Halbtöne und Oktave einzugeben, verzichtet, da die Umrechnungsroutine zu viel Platz und Zeit in Anspruch nimmt.

Beim Eingeben der DATA's sollten Sie darauf achten, daß bei der ersten Stimme die Zeilen 1000-1007, 1009-1016 und 1038-1045 sowie bei der zweiten Stimme die Zeilen 2000-2006, 2007-2013 und 2030-2035 (bis auf 2036) identisch sind.

Deswegen sollte man beim Eingeben die Zeilen einmal eintippen, im Bildschirmspeicher stehenlassen, mit dem Cursor nach oben fahren, jeweils die Zeilennummern verändern und RETURN drücken. Dadurch kann man sich einiges an Tipparbeit sparen.

Das Prinzip des "Inkrement-Counters" ist gerade anders herum: Hier wird der Counter bei jeder neuen Note auf 0 gesetzt und dann bei jedem Durchlauf um 1 erhöht. Danach wird der Counter mit der Notendauer verglichen. Ist der Wert des Counters kleiner als die Notendauer, so passiert nichts. Ist der Wert des Counters größer als die Notendauer, so wird eine neue Note gespielt und der Counter wieder auf 0 gesetzt usw.

Alle mehrstimmigen Programme des dritten Kapitels basierten auf dem Inkrement-Prinzip. Welches der beiden man anwendet, ist von der Programmstruktur und Programmlänge her einerlei, da laufen beide aufs gleiche hinaus. Lediglich bei der Ablaufzeit können sich beim Vergleich mit 0 beim Dekrement-Prinzip und beim Vergleich mit der Notendauer beim Inkrement-Prinzip geringfügige Differenzen ergeben.

5.2 Lineare Musikprogramme in Basic

Wir wollen uns das "Michelle"-Programm als Beispiel für ein mehrstimmiges, lineares Basic-Programm anschauen. "Linear" deshalb, weil Wiederholungen im Lied nicht im Programmteil berücksichtigt werden. Die Arbeitsweise nicht-linearer Basic-Musikprogramme betrachten wir dann im Anschluß.

Am Anfang eines Musikprogramms in einer höheren Programmiersprache stehen für gewöhnlich bestimmte Deklarationen. In Assembler, wie wir später sehen werden, finden wir diese Deklarationen in exakt der gleichen Form im Quelltext wieder, wohingegen im Objekt-Code, also im ablauffähigen Maschinencode, diese Deklarationen nicht mehr auftreten.

Wozu braucht man solche Deklarationen? Wir haben bereits gesehen, daß die Register des Sound-Chips in Basic über POKE angesprochen werden. Nun ist es recht mühselig, nach jedem POKE-Befehl fünfstellige Zahlen einzutippen. Deswegen werden diejenigen Sound-Chip-Register, die im Programm verwendet werden, wie Variablen gehandhabt.

```
10 ERSTER = 54272
20 L1     = ERSTER
30 H1     = ERSTER + 1
40 W1     = ERSTER + 4
50 A1     = ERSTER + 5
60 S1     = ERSTER + 6
70 L2     = L1     + 7
80 H2     = H1     + 7
90 W2     = W1     + 7
100 A2    = A1     + 7
110 S2    = S1     + 7
120 POKE A1,3 * 16+10
130 POKE S1,1 * 16+7
140 POKE A2,8 * 16+8
150 POKE S2,8 * 16+10
200 DIM F1 (200), D1 (200)
210 DIM F2 (200), D2 (200)
300 I=0
310 READ F1(I),D1(I):IF F1(I)>=0 THEN I=I+1:GOTO 310
320 I=0
330 READ F2(I),D2(I):IF F2(I)>=0 THEN I=I+1:GOTO 330
400 I1 = -1:I2 = -1
410 C1 = 1:C2 = 1
420 POKE ERSTER + 24,15
430 PRINT CHR$(147)
500 C1 = C1 - 1:PRINT CHR$(19)"1.STIMME:"C1;CHR$(157)"  ",
510 IF C1 > 0 THEN 580
520 I1 = I1 + 1
530 C1 = D1 (I1)
540 IF C1 < 0 THEN POKE L1,0:POKE H1,0:END
550 P2 = INT (F1(I1)/256)
560 P1 = F1(I1) - P2*256
570 POKE L1,P1:POKE H1,P2:POKE W1,33
580 IF C1 < 3  THEN POKE W1,32
600 C2 = C2 - 1:PRINT "2.STIMME:"C2;CHR$(157)"  "
610 IF C2 > 0 THEN 670
```

```
620 I2 = I2 + 1
630 C2 = D2 (I2)
640 P2 = INT (F2(I2)/256)
650 P1 = F2(I2) - P2*256
660 POKE L2,P1:POKE H2,P2:POKE W2,33
670 IF C2 < 2  THEN POKE W2,32
680 GOTO 500
1000 DATA 14435,12,14435,12
1001 DATA     0, 6,15294, 6
1002 DATA 11457,12,10814, 6
1003 DATA 14435, 6,10814, 6
1004 DATA 10207, 6, 9634, 6
1005 DATA 11457, 6,13625, 6
1006 DATA 11457, 6,10814,12
1007 DATA  9634, 6,11457, 4
1008 DATA 10814,26
1009 DATA 14435,12,14435,12
1010 DATA     0, 6,15294, 6
1011 DATA 11457,12,10814, 6
1012 DATA 14435, 6,10814, 6
1013 DATA 10207, 6, 9634, 6
1014 DATA 11457, 6,13625, 6
1015 DATA 11457, 6,10814,12
1016 DATA  9634, 6,11457, 4
1017 DATA 10814,20
1018 DATA 14435, 6,19269, 4
1019 DATA 17167, 4,14435, 4
1020 DATA 19269, 4,17167, 4
1021 DATA 14435, 4,21629, 8
1022 DATA 19269,16,     0, 3
1023 DATA 14435, 3,15294, 3
1024 DATA 14435, 3,15294, 6
1025 DATA 11457, 4,11457,26
1026 DATA     0, 3,14435, 3
1027 DATA 14435, 3,14435, 3
1028 DATA 19269, 6,14435, 6
1029 DATA 12860, 3,11457, 3
1030 DATA 10814, 9,10814, 3
1031 DATA 12860, 6,14435, 6
```

```
1032 DATA 14435, 6,14435, 6
1033 DATA 14435, 6,14435, 6
1034 DATA 14435, 3,14435,11
1035 DATA 14435, 4,14435,12
1036 DATA 12860, 6,11457, 6
1037 DATA 10814,24
1038 DATA 14435,12,14435,12
1039 DATA     0, 6,15294, 6
1040 DATA 11457,12,10814, 6
1041 DATA 14435, 6,10814, 6
1042 DATA 10207, 6, 9634, 6
1043 DATA 11457, 6,13625, 6
1044 DATA 11457, 6,10814,12
1045 DATA  9634, 6,11457, 4
1046 DATA 10814,20, 9634, 3
1047 DATA 10814, 3,11457, 6
1048 DATA  9634, 6,12860, 6
1049 DATA 10814, 6,11457, 6
1050 DATA  9634, 6,12860, 9
1051 DATA 10814, 3,11457,12
1052 DATA 10814, 6, 9634, 6
1053 DATA  9094,12, 9634, 6
1054 DATA 10814, 6, 9634,96
1055 DATA  9094,24
1999 DATA -1,-1
2000 DATA 2408, 12, 3608,12
2001 DATA 3215, 12, 3823,12
2002 DATA 4291, 18, 4050, 6
2003 DATA 3823, 24, 3608,12
2004 DATA 3823, 12, 3608, 6
2005 DATA 5407,  6, 3608, 6
2006 DATA 5407,  6
2007 DATA 2408, 12, 3608,12
2008 DATA 3215, 12, 3823,12
2009 DATA 4291, 18, 4050, 6
2010 DATA 3823, 24, 3608,12
2011 DATA 3823, 12, 3608, 6
2012 DATA 5407,  6, 3608, 6
2013 DATA 5407,  6
```

```
2014 DATA 2408, 12, 2408,12
2015 DATA 2408,  6, 3608, 6
2016 DATA 4817,  6, 5728, 6
2017 DATA 4291, 12, 2864,12
2018 DATA 3823, 12, 4291, 6
2019 DATA 4817,  6, 7217,12
2020 DATA 4817, 12, 3823,12
2021 DATA 3215, 12
2022 DATA 2408,  6, 3608, 6
2023 DATA 2273,  6, 3608, 6
2024 DATA 2145,  6, 3608, 6
2025 DATA 2025,  6, 3608, 6
2026 DATA 1911,  6, 3215, 6
2027 DATA 1911,  6, 2864, 6
2028 DATA 1804,  6, 2703, 6
2029 DATA 1804,  6, 2703, 6
2030 DATA 2408, 12, 3608,12
2031 DATA 3215, 12, 3823,12
2032 DATA 4291, 18, 4050, 6
2033 DATA 3823, 24, 3608,12
2034 DATA 3823, 12, 3608, 6
2035 DATA 5407,  6, 3608, 6
2036 DATA    0,  6
2037 DATA 3823, 24, 3608,24
2038 DATA 3215, 24, 3608,24
2039 DATA 4817,  8, 4817, 2
2040 DATA 5407,  2, 6069, 8
2041 DATA 6069,  2, 7217, 2
2042 DATA 7647, 12, 5728,12
2043 DATA 5407,  8, 5407, 2
2044 DATA 5728,  2, 6430, 6
2045 DATA 5407,  6, 4817, 8
2046 DATA 4817,  2, 5407, 2
2047 DATA 5728,  6, 4817, 6
2048 DATA 3608, 24
2999 DATA -1,-1

READY.
```

Aus folgender Tabelle können Sie sämtliche im Programm vertretenen Variablen entnehmen.

Variable	Funktion
L1	Frequenz Low von Oszillator 1
H1	Frequenz High von Oszillator 1
W1	Wellenform und Gate-Signal von Oszillator 1
A1	AD-Werte von Oszillator 1
S1	SR-Werte von Oszillator 1

(entsprechende Variablen L2, H2,... für Oszillator 2)

C1	Counter für erste Stimme
I1	Index für erste Stimme
F1	Array für Frequenzen der ersten Stimme
D1	Array für Notendauern der ersten Stimme

(entsprechende Variablen C2, I2,... für Oszillator 2)

(P1, P2 für die Low-High-Umrechnung, I als Arrayindex)

Man kann leicht erkennen, daß sich die im Programm verwendeten Variablen in drei Gruppen einteilen lassen:

1. Die Variablen, die für Sound-Chip-Register stehen.

2. Die Variablen, die die Daten der Tonhöhen und Tondauern betreffen.

3. Die Variablen, die für Umrechnungen u.ä. gebraucht werden.

Man kann die Funktion eines Musikprogramms durch diese Einteilung leicht in einem Satz zusammenfassen: Die Variablen von 2. werden unter Zuhilfenahme von 3. innerhalb eines bestimmten Zeitabschnitts Schritt für Schritt nach 1. übertragen.

Diesen Vorgang kann man auch als "zeitlich geordnetes Beschreiben der Sound-Chip-Register" bezeichnen. Nicht mehr und nicht weniger muß ein jegliches Musikprogramm leisten. Es muß bestimmte Daten (Tonhöhen) nach und nach, unterbrochen von bestimmten Zeiten (Tondauern), in die Sound-Chip-Register schreiben.

Schauen wir uns das Programm weiter an! Nach den Deklarationen kommt ein Programmabschnitt, der den Variablen der zweiten Gruppe bestimmte Anfangswerte zuweist.

Einen solchen Vorgang, Anfangswerte zuzuweisen, nennt man auch "Initialisierung" von Variablen. Hier werden vier Arrays dimensioniert und mit den Daten der DATA-Statements gefüllt. Danach werden Counter und Array-Indizes mit Anfangswerten belegt.

Nach der "Initialisierung" haben wir von Zeile 500 bis 680 den Kern des Musikprogramms vor uns, die Spielroutine. Sie tut eigentlich nichts anderes, als den Inhalt der vier Arrays auszuwerten. Notenhöhen und Notendauern werden nach und nach "abgespielt", indem der Array-Index nach und nach inkrementiert wird.

Würde der Index innerhalb der Initialisierung auf das Arrayende gesetzt werden und innerhalb der Spielroutine dekrementiert werden, so könnte man das Musikstück rückwärts spielen lassen. Wir sehen also, daß man neben den zwei unabhängigen Countern noch zwei unabhängige Array-Indizes braucht.

Counter und Indizes, das sind die beiden Dinge, die den zeitlichen Ablauf einer Stimme bestimmen. Der Wert eines Counters hängt davon ab, wie lange eine Note gespielt werden soll. Was wir in den DATA-Zeilen unter Tondauer finden, ist stets der Maximalwert, den der Counter im Ablauf einer Note annimmt.

Der Wert eines Notenindex hängt von der Anzahl der Noten ab, aus der eine Stimme besteht. Man kann sich die Arbeitsweise eines Index verständlich machen, indem man sich alle Noten einer Stimme durchnumeriert denkt. Der Index zählt nun einfach von ersten bis zur letzten Note einer Stimme alle Noten durch.

Das Programm "Michelle" ist ein lineares Musikprogramm. Beim Eingeben haben wir gesehen, daß ein Teil des Stücks dreifach vorkam und deswegen auch dreimal eingetippt werden mußte.

Wenn man das Prinzip des Index verstanden hat und programmiertechnisch anwenden kann, so kann man sich jegliches mehrfaches Eintippen von Teilen eines Musikstückes ersparen, indem man die Indexsteuerung erweitert.

Bisher wurde der Index immer nur um 1 erhöht (linear!). Wie man durch das (auch mehrfache) Zurücksetzen eines Index Wiederholungen steuert, wollen wir im folgenden Programm sehen.

I FEEL LOVE

```
10 ERSTER = 54272
20 L1     = ERSTER
30 H1     = ERSTER + 1
40 W1     = ERSTER + 4
50 A1     = ERSTER + 5
60 S1     = ERSTER + 6
70 L2     = L1     + 7
80 H2     = H1     + 7
90 W2     = W1     + 7
100 A2    = A1     + 7
110 S2    = S1     + 7
120 POKE A1,6 * 16+7
130 POKE S1,6 * 16+12
140 POKE A2,1 * 16+6
150 POKE S2,8 * 16+7
200 DIM F1 (200), D1 (200)
210 DIM F2 (200), D2 (200)
300 I=0
310 READ F1(I),D1(I):IF F1(I)>=0 THEN I=I+1:GOTO 310
320 I=0
330 READ F2(I),D2(I):IF F2(I)>=0 THEN I=I+1:GOTO 330
400 I1 = -1:I2 = -1
410 C1 = 1:C2 = 1
420 TI$ = "000000"
430 T1 = TI:T2 = T1
440 POKE ERSTER + 24,15
500 IF T1 < TI + 100 THEN C1 = C1 - 1
510 IF C1 > 0 THEN 580
520 I1 = I1 + 1
530 C1 = D1 (I1):T1 = TI
540 IF C1 < 0 THEN POKE L1,0:POKE H1,0:END
550 P2 = INT (F1(I1)/256)
560 P1 = F1(I1) - P2*256
570 POKE L1,P1:POKE H1,P2:POKE W1,33
580 IF C1 < 5  THEN POKE W1,32
```

```
600 IF T2 < TI + 100 THEN C2 = C2 - 1
610 IF C2 > 0 THEN 670
620 I2 = I2 + 1
630 C2 = D2 (I2):T2 = TI
640 P2 = INT (F2(I2)/256)
650 P1 = F2(I2) - P2*256
660 POKE L2,P1:POKE H2,P2:POKE W2,33
670 IF C2 < 2  THEN POKE W2,32
680 IF I2=31 THEN IF C2=1 THEN I2=-1:C2=1
690 GOTO 500
1000 DATA  8583,24, 7647,24
1001 DATA  7217,24, 8101,24
1002 DATA  8583,24,10207,24
1003 DATA 11457,24,12860,24
1004 DATA     0,24,    0,24
1005 DATA     0,24,    0,24
1006 DATA  8583,18, 6430, 6
1007 DATA 10207,18, 7647, 6
1008 DATA 11457,18, 8583, 6
1009 DATA 12860,18, 9634, 6
1010 DATA  8583,24,    0,24
1011 DATA     0,24,    0,24
1012 DATA 17167, 6,12860, 6
1013 DATA 11457, 6,12860, 6
1014 DATA  8583,12,    0,12
1015 DATA  8583, 6,11457, 6
1016 DATA 12860, 6,17167, 6
1017 DATA 16203,12,    0,12
1018 DATA  8583, 6
1999 DATA -1,-1
2000 DATA 1072,  3, 1072, 3
2001 DATA  803,  3,  955, 3
2002 DATA 1072,  3, 1072, 3
2003 DATA  803,  3,  955, 3
2004 DATA 1275,  3, 1275, 3
2005 DATA  955,  3, 1136, 3
2006 DATA 1275,  3, 1275, 3
2007 DATA  955,  3, 1136, 3
2008 DATA 1432,  3, 1432, 3
2009 DATA 1072,  3, 1275, 3
2010 DATA 1432,  3, 1432, 3
2011 DATA 1072,  3, 1275, 3
2012 DATA 1607,  3, 1607, 3
2013 DATA 1204,  3, 1432, 3
2014 DATA 1607,  3, 1607, 3
2015 DATA 1204,  3, 1432, 3
2999 DATA -1,-1

READY.
```

5.3 Ein Beispiel zu einem nicht-linearen Basic-Musikprogramm

Um dieses Programm, das auf den vorangegangenen Seiten abgedruckt ist, einzugeben, ist es am besten, beim Programm "Michelle" mit einem Toolkit den Befehl "DELETE 1000 -" einzugeben. Das Hauptprogramm muß man nicht völlig neu eingeben, es reicht, wenn man die beiden Listings vergleicht und die Unterschiede durch Editieren des im Speicher befindlichen Programms eingibt.

Dieses Programm spielt neben einer freien Melodie eine Basslinie des Lieds "I Feel Love" von Donna Summer nach. Diese Basslinie läuft das ganze Stück durch. Eine solche Basslinie, die über einen längeren Zeitabschnitt immer wiederholt wird, nennt man auch einen "ostinaten" Bass oder einfach nur "Ostinato".

Wir haben hier den einfachsten Fall einer Wiederholung vor uns. Es wird einfach ein Abschnitt einer Stimme immer wieder wiederholt. Jedoch läuft diese Wiederholung nur in der zweiten Stimme ab, die erste Stimme läuft unabhängig von der zweiten.

Würden wir diese Wiederholung nicht im Programmteil steuern, so wäre es nötig, die Programmzeilen von 2000 bis 2015 für jede neue Wiederholung noch einmal einzugeben. Das würde das Programm unnötig aufblähen. Die Anweisung, daß die Zeilen 2000 bis 2015 immer wieder wiederholt werden sollen, finden wir in Zeile 680.

Diese Anweisung kann man auch folgendermaßen ausdrücken: Wenn der Index (I2) den Wert der letzten Note, die vor der Wiederholung gespielt werden soll, erreicht hat, dann prüfe, ob der Counter (C2) den Wert hat, der anzeigt, daß die Note zu Ende gespielt wurde.

Wenn das der Fall ist, dann setze den Index auf den Wert vor der ersten Note, ab der wiederholt werden soll und den Counter auf den Wert, durch den in der Spielroutine unmittelbar beim ersten Ausführen die nächste Note gespielt wird.

Das ist zugegebenermaßen eine sehr komplizierte Anweisungsfolge, bei deren Umsetzung in eine Programmzeile man leicht Fehler produzieren kann. Doch es ist besser, eine komplizierte Programmzeile zu schreiben, als sinnlos DATA's mehrfach einzutippen.

Diese Art der IF...THEN IF...THEN IF...THEN...-Anweisung finden Sie auch in allen Programmen des dritten Kapitels, in denen Wiederholungen programmiert wurden.

Eine Anmerkung für Basic-Kenner: Es ist zeitsparender, in Musikprogrammen bei Abfrage auf Mehrfachbedingungen die Anweisungsfolge IF... THEN IF...THEN IF zu verwenden als IF (...AND...AND...AND...) THEN, da bei der "AND"-Variante bei jedem Schleifendurchlauf alle Bedingungen geprüft werden, was sehr viel Zeit in Anspruch nimmt. Wenn man hingegen IF...THEN-Anweisungen geschickt schachtelt, kann man dadurch, daß die Bedingungen verschiedene Prioritäten haben, viel Zeit sparen.

5.4 Die Grenzen der Musikprogrammierung in Basic

Einstimmige Melodien kann man in Basic ja noch gut programmieren, aber bei mehrstimmigen Basic-Programmen hat man das Problem, daß sie über eine bestimmte Geschwindigkeit nicht hinauskommen. Die Geschwindigkeit des Stücks selbst ist hierbei nicht so wichtig, vielmehr ist es wichtig, daß die Noten, die "gleichzeitig" gespielt werden sollen, auch schnell genug hintereinander gespielt werden.

Es ist nicht möglich, zwei Noten exakt zur gleichen Zeit zu spielen, weder für einen Klavierspieler noch für einen Computer. Es kommt nur darauf an, daß die Noten so schnell hintereinander gespielt werden, daß der Eindruck entsteht, sie würden tatsächlich "gleichzeitig" gespielt.

Das menschliche Ohr kann Töne, die ungefähr um eine Zehntelsekunde auseinander liegen, noch voneinander unterscheiden. Töne, die schneller hintereinander gespielt werden, scheinen dann "gleichzeitig" zu kommen. Die Aufgabe eines Musikprogramms ist es, diese Zehntelsekunde Zeitdifferenz zu unterbieten.

Dies ist auch genau der Punkt, wo mehrstimmige Musikprogramme in Basic aussteigen. Bei den zweistimmigen Programmen ist es noch nicht so schlimm, aber bei den dreistimmigen merkt man schon, daß es an der Genauigkeit hapert.

Um ein Musikprogramm möglichst genau klingen zu lassen, das wollen wir hier noch einmal sehen, ist es notwendig, daß die Noten der einzelnen Stimmen möglichst schnell hintereinander gespielt werden. Also müssen die entsprechenden Routinen möglichst kurz sein und unmittelbar aufeinander folgen.

Es gibt einige Programmierkniffe, wie man diese Routinen programmiert, so daß sie zeitlich möglichst schnell ablaufen. Wir haben bereits den Kniff gesehen, wie man Mehrfachabfragen unter Verwendung der IF THEN IF THEN IF-Befehlsfolge zeitsparend programmiert. Ein weiterer Punkt ist die Umrechnung der Frequenzen in die Low-/High-Werte.

Im vorigen Beispielprogramm wurde absichtlich einmal ein Weg gezeigt, wie Zeit verloren geht. Betrachten Sie einmal die Zeilen 550-570 und die entsprechenden Zeilen 640-660! Sie sehen: die Umrechnung erfolgt innerhalb der Spielroutine. Das ist auch der Grund dafür, daß in diesem Programm die Noten zuweilen nicht ganz exakt zu kommen scheinen.

In den Programmen des zweiten Kapitels, in denen Ihnen jeweils der beste Weg gezeigt wurde, wie man in Basic Musik programmiert, wurde diese Umrechnung außerhalb der eigentlichen Spielroutine gestellt.

5.5 Eine Anmerkung zum Flag-Konzept

Wir haben gesehen, daß man durch Zurücksetzen des Index einer Stimme Wiederholungen programmieren kann. Wichtig war hierbei, daß der Index auf die Note vor derjenigen Note, mit der die Wiederholung beginnen sollte, gesetzt wird, und der Counter auf einen solchen Wert, daß die Spielroutine beim ersten Ansprung die erste Note der Wiederholung spielt.

Wenn man nun einen bestimmten Teil eines Musikstücks eine bestimmte Anzahl mal wiederholen möchte, braucht man eine zusätzliche Variable, die die Anzahl der Wiederholungen zählt. Diese Variable wird in diesem Buch "Flag" genannt.

Am Anfang eines Programms werden alle Flags - jede Stimme hat ihr eigenes - auf 0 gesetzt. Mit jeder Wiederholung wird das Flag der betreffenden Stimme um 1 erhöht. Die Wiederholungsroutine frägt zunächst nach, ob das Flag schon einen bestimmten Maximalwert hat. Ist das der Fall, so wird die Wiederholungsroutine übersprungen.

Beispiele für die Anwendung von Flags finden Sie im zweiten Kapitel. Im Programm "Yesterday" wurden Flags für Wiederholungen und auch Temposteuerung eingesetzt. Im Programm "Strawberry Fields Forever" sehen Sie, wie Flags für Sound-Umschaltungen genutzt werden.

Bei der Musikprogrammierung in Assembler werden wir uns noch näher mit dem Flag-Konzept beschäftigen.

5.6 Das Konzept der Musiktools

Nachdem wir nun auf die hauptsächlichen Schwächen des Musikprogrammierens in Basic eingegangen sind, wollen wir sehen, wie man es besser machen kann. Da der wunde Punkt vor allem die Geschwindigkeit ist, liegt es nahe, auf Maschinensprache oder Assembler auszuweichen. In Assembler ist die Geschwindigkeit kein Problem mehr, dafür tritt ein anderes, noch viel schwerwiegenderes Problem auf: die Bedienungsfreundlichkeit. Das Arbeiten mit Monitor und Assembler ist für den Nicht-Profi ein Greuel. Aus diesem Grund haben wir uns bisher auf Basic beschränkt, da dort insbesondere die Editiermöglichkeiten optimal sind.

Wenn sowohl Basic als auch Assembler Vor- und Nachteile haben, was liegt da näher, als die Vorteile beider Programmiersprachen zu verbinden? Hier stoßen wir auf das Konzept der Musiktools.

Der Punkt, bei dem es auf die Geschwindigkeit ankommt, sind die Spielroutinen, das haben wir bereits gesehen. Was liegt also näher, als die eigentlichen Spielroutinen nach Assembler zu übertragen, womit man eine optimale Geschwindigkeit erreicht, und die komfortable Noteneingabe in Basic zu belassen? Genau das ist das Konzept der meisten Musiktools. Hier werden schnelle Zusatzbefehle wie Basic-Befehle zur Verfügung gestellt, die mehrere Noten gleichzeitig spielen. Die Noten selbst werden wie Basic-Statements eingegeben.

Es gibt zur Zeit (Anfang 1984) schon so viele unterschiedliche Musiktools (und es ist abzusehen, daß ihre Zahl noch steigen wird), daß eine Beschreibung eines speziellen Musiktools hier fehl am Platz wäre. Es sollen hier nur die Grundlagen des Arbeitens mit einem Musiktool erläutert werden.

Die meisten Musiktools erlauben es, die Noten eines Musikstücks wie in den längeren Programmen des zweiten Kapitels über ihre Notennamen und Oktaven einzugeben. Hier müssen Sie dann nicht mehr laufend die "Tabelle der Tonfrequenzen" aus dem vierten Kapitel zur Hilfe nehmen.

Die Notenumrechnung erfolgt in Assembler und ist demzufolge entscheidend kürzer als in einem reinen Basic-Programm. Hier müssen Sie darauf achten, wie die Halbtöne bezeichnet werden. Bei Programmen aus dem englischen Sprachraum heißt der Halbton "H" ja "B".

Außerdem müssen Sie darauf achten, wie die Erhöhungen und Erniedrigungen bezeichnet werden müssen, also in welcher Form "Cis" und "Des" eingegeben werden müssen. Ferner ist die Bezeichnungsweise der Oktaven nicht standardisiert. Meistens wird jedoch wie in diesem Buch die erste, tiefste Oktave mit "0" (also z.B. C0) und die höchste Oktave mit "7" (also z.B. A7) bezeichnet.

Der nächste Punkt sind die Notendauern. Sind sich bei den Notennamen viele Musiktools noch einig, so denkt sich bei den Notendauern jeder Autor eines Musiktools etwas anderes aus.

Viele Musiktools bedienen sich der musikalischen Bezeichnung von Vierteln und halben Noten, aber die Eingabeformalismen unterscheiden sich stark. Bei "Synthy 64" verwendet man einen Schrägstrich, bei "Simon's Basic" die Funktionstasten. Hier hilft nichts als in der Anleitung nachsehen.

Nachdem wir die Noteneingabe behandelt haben, kommen wir zur Klangeinstellung. Hier unterscheiden sich die Musiktools am stärksten, und hieran kann man auch die Qualität eines Musiktools beurteilen.

Viele Musiktools zeichnen sich hier durch Einfallslosigkeit aus. Sie haben einen Befehl (etwa "SOUND" bei Simon's Basic), der es ermöglicht, wie über POKE das Register für die Wellenformen zu setzen. Hier wird dem Benutzer eigentlich nichts erspart, man könnte genauso gut auch POKE eingeben. Dieselbe Situation findet man auch bei den Filtereinstellungen. Hier muß der Benutzer oft die Bits genauso ausrechnen wie auch bei POKE.

Worum der Benutzer sich aber nicht mehr kümmern muß, ist die eigentliche Spielroutine. Das automatische Setzen und Löschen des Gate-Signals beim Spielen eines Musikstücks haben alle Spielroutinen gemeinsam.

Eine Besonderheit bietet hier "Synthy 64". Dieses Musiktool hat eine sogenannte "TRACE"-Funktion. Erinnern Sie sich an das "Read-Write Register"-Programm aus dem vierten Kapitel. Dieses Prinzip, sämtliche Register in einem gesonderten Speicherbereich zwischenzuspeichern und dann beliebig abrufen zu können, liegt auch dieser Funktion zugrunde.

Hier kann man sehr schön verfolgen, wie im Laufe eines Stückes die verschiedenen Noten gespielt werden und sogar Sound-Umschaltungen erfolgen. Bei den Sound-Umschaltungen sollte "Synthy 64" ein Beispiel für die Autoren anderer Musiktools sein: Hier können Soundeinstellungen wie Basic-Unterprogramme eingegeben und über "GOSUB" aufgerufen werden.

Bei der Ablaufsteuerung eines Musikstücks gibt es oft einen weiteren Befehl, der es erlaubt, die Ablaufgeschwindigkeit beliebig zu variieren ("TEMPO", "SPEED" o.ä.). Hier hat man eine wesentlich größere Temporeserve als in reinen Basic-Musikprogrammen.

Alles dies, bis vielleicht auf die größere Geschwindigkeit, kann man auch durch Basic-Routinen in etwa nachvollziehen. Doch es gibt einen Punkt, wo manche Musiktools einen entscheidenden Vorteil haben:

Sie können die Musik simultan zur Ausführung eines normalen Programms spielen. Das heißt, während ein Musikstück gespielt wird, können gleichzeitig völlig andere Vorgänge ablaufen, wie z.B. Sprite-Steuerung, Grafik-Routinen oder Joystick-Abfrage.

Doch auch dies geht nur in einem begrenzten Rahmen. Das liegt daran, daß sobald auf irgendeine Art und Weise Basic-Routinen zur Variablenverwaltung oder zur Befehlsauffindung in Programmzeilen genutzt werden, eine Menge Zeit verloren geht. Diese Routinen sind nämlich sehr, sehr zeitaufwendig.

Wenn ein Spiel programmiert werden soll, das Grafik und Musik auf irgendeine Weise kombiniert, wird keiner daran denken, dies etwa mit Simon's Basic zu schreiben.

Die Geschwindigkeit, die in einem Spiel benötigt wird, erhält man nur, wenn man zu 100% in Assembler programmiert. Und selbst, wenn man nur eine Titelmelodie zu einem Programm schreiben will, kommt man um Assembler nicht herum. Deswegen wollen wir nun auf die Musikprogrammierung in Assembler eingehen.

5.7 Anmerkung zu den Assembler-Programmen dieses Buchs

Sämtliche Assembler-Programme in diesem Buch wurden mit "Profi-Ass" erstellt, einem Assembler der Firma Data Becker. Dieser Assembler ist auch im Programmpaket "PROFIMAT" enthalten.

Die Version von "Profi-Ass", mit der die Programme erstellt wurden, liegt im Speicher von $C000-$CFFF und belegt deshalb keinen Basic-RAM. Falls Sie mit einer Profi-Ass-Version arbeiten, die von $9000-$9FFF liegt und 4 K Basic-RAM beansprucht, sollten Sie darauf achten, daß der SYS-Befehl zu Beginn jedes Profi-Ass-Quelltextes von SYS 49152 auf SYS 36864 geändert werden muß.

5.8 Ein mehrstimmiges Musik-Programm in Assembler

Es wäre vom Platz her viel zu aufwendig, um an dieser Stelle ähnlich wie im dritten Kapitel die Entwicklung eines Assemblerprogramms von der einstimmigen bis zur dreistimmigen Form zu verfolgen. Zudem sind die programmiertechnischen Zusammenhänge ungleich komplizierter. Ihnen wird deshalb ein fertiges dreistimmiges Musikprogramm vorgestellt.

Es handelt sich hierbei um das Musikstück "Cinema Show" der Gruppe Genesis. Wie bei allen Assembler-Programmen in diesem Buch ist sowohl der Quelltext als auch ein Basic-Loader abgedruckt. Wenn Sie nicht mit Profi-Ass arbeiten, können Sie den Basic-Loader eingeben, um das Stück zu hören. Wenn Sie jedoch Profi-Ass besitzen, so ist es eine gute Übung, den Quelltext einzugeben, da man hier einiges über die Struktur eines Assemblerprogramms lernt.

Wir wollen im folgenden den Quelltext Schritt für Schritt durchgehen. Die Programmabschnitte ohne besondere Bezeichnung sind als fix anzusehen. Die Programmabschnitte, die variabel sind, sind als solche ausgezeichnet. Es ist ohne weiteres möglich, das Programm so zu verändern, daß es ein beliebiges anderes Musikstück spielt.

0000-0002: Da der Objekt-Code von $8000 an aufwärts liegt, wird RAM-Top für Basic auf $8000 gesetzt. Nach dem Start-SYS werden die Ausgabeoptionen auf Object at Origin ("OO") gesetzt.

0011-0035: Hier haben wir den ersten Teil der Deklarationen vor uns. Sämtliche Sound-Chip-Register werden deklariert.

0050-0073: Im zweiten Teil der Deklarationen werden sämtliche Noten als Profi-Ass-Symbole deklariert. Das hat den Vorteil, daß die Notenumrechnung direkt beim Assemblieren erfolgt.

0080-0089: Im dritten Teil der Deklarationen werden die "Variablen" des Assembler-Programms definiert. Es handelt sich hierbei um Counter, Pointer und Flags.

0110-0118: Textausgabe-Routine für Titeltext

0120-0149: freier Platz für die Ascii-Codes des Titeltexts. Dieser Teil ist variabel, es kann ein beliebiger Text eingegeben werden.

0150-0152: Die eigentliche Spielroutine wird als Unterprogramm aufgerufen. Nach dem Stück wird die Lautstärke auf 0 gesetzt und mit RTS nach Basic zurückgesprungen.

0200-0250: Hier haben wir die Initialisierung vor uns, eine Mischung aus fixen und variablen Assembler-Anweisungen. Wichtig ist hier, daß die "Variablen" des Assembler-Programms, also Counter und Pointer, definierte Anfangswerte erhalten. Weiterhin wird in der Initialisierung der Anfangssound eingestellt. Achten Sie darauf, daß die Wellenform der drei Oszillatoren in den drei Variablen WAVE1, WAVE2 und WAVE3 gespeichert werden muß. Das LSB muß hier gelöscht sein, sonst funktioniert die Steuerung des Gate-Signals nicht.

0500-0523: Hier haben wir die Spielroutine für die erste Stimme vor uns. Sie läuft unabhängig von den beiden anderen Stimmen. Wir haben es mit einem Inkrement-Counter zu tun, der jeweils um 1 erhöht wird. Wenn er den Wert der Notendauer erreicht hat, so wird eine neue Note gespielt und das Gate-Signal gesetzt. Wenn er die Hälfte der Notendauer erreicht hat, so wird das Gate-Signal zurückgesetzt.

0600-0623: Dies ist die Spielroutine für die zweite Stimme. Hier gilt das gleiche wie bei der ersten Stimme.

0700-0723: Dies ist die Spielroutine für die dritte Stimme. Hier gilt das gleiche wie bei der ersten Stimme.

0800-0918: Hier haben wir freien Raum für beliebige Flag-Abfragen. Durch eine Flag-Abfrage können beliebige Teile einer Stimme beliebig oft wiederholt werden und an einer beliebigen Stelle eines Stücks Sound-Umschaltungen vorgenommen werden. Die i-te Flag-Abfrage hat folgendes Format:

```
XX0 N<i-1> *=*
XX1 LDA POINTER<v>  :CMP #<POS<nr>-1:BNE N<i>
XX2 LDA POINTER<v>+1:CMP #>POS<nr>-1:BNE N<i>
XX3 LDY #0:LDA (POINTER<v>),Y
XX4 TAX:DEX:CPX COUNTER<v>:BNE N<i>
```

Diese Anweisungsfolge läßt das Programm nur zu EINEM ganz bestimmten Zeitpunkt durch, nämlich genau dann, wenn die Note vor dem Label POS<nr> gerade fertig gespielt wurde. In allen anderen Fällen, also wenn der Pointer nicht auf die letzte Note vor der Wiederholung zeigt oder die Note noch nicht fertiggespielt wurde (das ist dann, wenn der Counter noch nicht seinen Maximalwert minus 1 erreicht hat), wird mit BNE auf die nächste Flag-Abfrage gesprungen.

(An dieser Stelle sieht man auch, daß bei diesem Assembler-Programm das Prinzip des Inkrement-Counters angewandt wurde, da der Counter mit einem Maximalwert verglichen wird.)

Bei Wiederholungen geht die Anweisungsfolge folgendermaßen weiter:

```
XX5 INC FLAG<v>
XX6 LDA FLAG<v>:CMP #<Wert nach letzter Wieder-
                     holung + Anzahl der Wie-
                     derholungen>:BEQ N<i>
XX7 LDA #< <Position des Wiederanfangs>:
                              STA POINTER<v>
XX8 LDA #> <Position des Wiederanfangs>:
                            STA POINTER<v>+1
```

Hier muß im Datenteil <Position des Wiederanfangs> das Format XXXX POS<nr> .BYTE 1 haben. Das ".BYTE 1" kann man sich auch schenken, wenn man folgendes Format verwendet:

```
(Zeilen XX5 und XX6 wie vorhin)
XX7 LDA #< <Position des Wiederanfangs>-1:
                              STA POINTER<v>
XX8 LDA #> <Position des Wiederanfangs>-1:
                            STA POINTER<v>+1
XX9 LDA #200:STA COUNTER<v>
```

Dann hat die Positionszeile das übliche Format. Dieses Format lautet wie folgt: XXXX POS<nr> *=*. Die Anweisungsfolge "*=*" ist übrigens die Leeranweisung von Profi-Ass. Man kann sie gut verwenden, wenn man in einer Programmzeile nur ein Label definieren will.

In den vorangehenden Zeilen bedeutete XXX immer eine Zeilennummer oder den Teil einer Zeilennummer. <i> stand für die Nummer der Flag-Abfrage. <v> stand für die Stimme, auf die die Flag-Abfrage sich bezog. <nr> ist die Nummer der Positionszeile, die man der Deutlichkeit halber so setzen sollte:

POS11, POS12, POS13... Positionszeilen der ersten Stimme
POS21, POS22, POS23... Positionszeilen der zweiten Stimme
POS31, POS32, POS33... Positionszeilen der dritten Stimme

Wenn man Sound-Umschaltungen programmieren will, so kann man die Zeilen XX5 bis XX9 für beliebige Assembler-Befehle zum Setzen der Sound-Chip-Register benutzen. Wenn man jedoch die Wellenform ändern will, so sollte man die Variablen WAVE1, WAVE2 und WAVE3 verändern. Zusätzlich kann man noch der Variablen SPEED einen neuen Wert zuweisen. Somit ist es möglich, mitten im Stück das Tempo zu ändern.

0990-0992: Dies ist die Warteschleife, die nach jedem Durchlauf der Spielroutine und der Flag-Abfragen durchlaufen wird. Es ist möglich, zwischen die "LOOP"-Schleife und dem Sprungbefehl "JMP LOOP1" andere Programmstücke einzulagern. Ein erfahrener Assembler-Programmierer könnte hier Grafikroutinen usw. "dazwischenschieben". Diese Assemblerroutinen müssen jedoch folgenden Voraussetzungen genügen: Sie müssen Schleifenrumpfcharakter haben, d.h. sie müssen laufend terminieren. Es ist also notwendig, daß diese Routinen nur kurze Zeit in Anspruch nehmen und dann sofort den Befehl "JMP LOOP1" ausführen. Es wäre z.B. nicht zulässig, daß wenn eine solche Routine auf eine gedrückte Taste warten sollte, dann intern eine Warteschleife enthält. Sie muß dann vielmehr mit einem Flag-System arbeiten.

1000-1999: Dies ist der variable Datenteil, der Tonhöhen und Tondauern der ersten Stimme enthält. Die Zeile 1000 ist obligatorisch. Alle anderen Zeilen sind entweder Positionszeilen wie vorhin beschrieben, oder sie sind nach folgendem Muster aufgebaut: XXXX .WORD <Tonhöhe>:.BYTE <Tondauer>. Als Tondauer sind Werte von 0 bis 255 zulässig. Als Tonhöhe ist jeder Wert von 0 bis 65535 zulässig. "0" kann man als Pause verwenden. Ansonsten kann man die Notensymbole verwenden, die von Zeile 50 bis 73 definiert wurden.

Die Zeilen 2000-2999 und 3000-3999 werden für die Tonhöhen und Tondauern der zweite und dritte Stimme genutzt. Auch hier müssen die Zeilennummern 2000 und 3000 das Format haben: <Zeilennummer> STIMME<x> .BYTE 1, wobei <x> für die Stimme steht. Ansonsten ist alles analog wie bei der ersten Stimme.

```
0 POKE56,128:CLR:SYS49152
1 .OPT OO
2 *=$8000
3 ;
4 ; THE CINEMA SHOW
5 ; ASSEMBLER - SOURCE
6 ; ARRANGEMENT BY THOMAS DACHSEL
7 ;
8 ;>>KONSTANTEN
9 ;
11 OSC1LF = $D400
12 OSC1HF = $D401
13 OSC1PL = $D402
14 OSC1PH = $D403
15 OSC1WV = $D404
16 OSC1AD = $D405
17 OSC1SR = $D406
18 OSC2LF = $D407
19 OSC2HF = $D408
20 OSC2PL = $D409
21 OSC2PH = $D40A
22 OSC2WV = $D40B
23 OSC2AD = $D40C
24 OSC2SR = $D40D
25 OSC3LF = $D40E
26 OSC3HF = $D40F
27 OSC3PL = $D410
28 OSC3PH = $D411
29 OSC3WV = $D412
30 OSC3AD = $D413
31 OSC3SR = $D414
32 FILTLF = $D415
33 FILTHF = $D416
34 FILMOD = $D417
35 VOLUME = $D418
50 C1  =  536:C2  = 1072:C3  = 2145
51 C4  = 4291:C5  = 8583:C6  =17167
```

```
52 CS1 =  568:CS2 = 1136:CS3 = 2273
53 CS4 = 4547:CS5 = 9094:CS6 =18188
54 D1  =  602:D2  = 1204:D3  = 2408
55 D4  = 4817:D5  = 9634:D6  =19269
56 DS1 =  637:DS2 = 1275:DS3 = 2551
57 DS4 = 5103:DS5 =10207:DS6 =20415
58 E1  =  675:E2  = 1351:E3  = 2703
59 E4  = 5407:E5  =10814:E6  =21629
60 F1  =  716:F2  = 1432:F3  = 2864
61 F4  = 5728:F5  =11457:F6  =22915
62 FS1 =  758:FS2 = 1517:FS3 = 3034
63 FS4 = 6069:FS5 =12139:FS6 =24278
64 G1  =  803:G2  = 1607:G3  = 3215
65 G4  = 6430:G5  =12860:G6  =25721
66 GS1 =  851:GS2 = 1703:GS3 = 3406
67 GS4 = 6812:GS5 =13625:GS6 =27251
68 A1  =  902:A2  = 1804:A3  = 3608
69 A4  = 7217:A5  =14435:A6  =28871
70 B1  =  955:B2  = 1911:B3  = 3823
71 B4  = 7647:B5  =15294:B6  =30588
72 H1  = 1012:H2  = 2025:H3  = 4050
73 H4  = 8101:H5  =16203:H6  =32407
77 ;
78 ;>>VARIABLEN
79 ;
80 POINTER1 =2:POINTER2 =5:POINTER3 =8
81 COUNTER1 =4:COUNTER2 =7:COUNTER3 =10
82 WAVE1    =$2C0
83 WAVE2    =$2C1
84 WAVE3    =$2C2
85 FLAG1    =$2D0
86 FLAG2    =$2D1
87 FLAG3    =$2D2
88 SPEED    =$2E0
89 FLAGX    =$2E1
100 ;
101 ;>>MAIN PROGRAM
102 ;
110 LDY #0
```

```
111 LDA #<TEXT:STA POINTER1
112 LDA #>TEXT:STA POINTER1+1
113 READSIGN LDA (POINTER1),Y
114 BNE PRINT
115 JMP PLAY
116 PRINT JSR $FFD2
117 INC POINTER1:BNE *+4:INC POINTER1+1
118 JMP READSIGN
120 TEXT  .BYTE $93,$D,$D,$D,$D,$D,$D,$D,$D,$D,$D
121 .ASC "            THE CINEMA SHOW":.BYTE $D
122 .ASC "          MUSIC BY: GENESIS":.BYTE $D
123 .ASC "     ARRANGEMENT BY: THOMAS DACHSEL"
124 .BYTE 0
150 PLAY SEI:JSR INIT:CLI
151 LDA #0:STA VOLUME
152 RTS
200 INIT LDA # 31:STA VOLUME
201 ^    LDA #$25:STA OSC1AD
202 ^    LDA #$7B:STA OSC1SR
203 ^    LDA #$25:STA OSC2AD
204 ^    LDA #$7B:STA OSC2SR
205 ^    LDA #$49:STA OSC3AD
206 ^    LDA #$69:STA OSC3SR
207 ^    LDA #$00:STA FILMOD
210 LDA #<STIMME1:STA POINTER1
211 LDA #>STIMME1:STA POINTER1+1
212 LDA #<STIMME2:STA POINTER2
213 LDA #>STIMME2:STA POINTER2+1
214 LDA #<STIMME3:STA POINTER3
215 LDA #>STIMME3:STA POINTER3+1
220 LDA #0:STA COUNTER1:STA FLAG1
221 ^      STA COUNTER2:STA FLAG2
222 ^      STA COUNTER3:STA FLAG3
230 LDA # 16:STA WAVE1
231 LDA # 16:STA WAVE2
232 LDA # 32:STA WAVE3
240 LDA #120:STA SPEED
250 LDA #  0:STA FLAGX
500 LOOP1 INC COUNTER1
```

```
501 LDA COUNTER1:LDY #0:CMP (POINTER1),Y
502 BCC GATE1
504 INC POINTER1:BNE *+4:INC POINTER1+1
505 LDY #0:LDA (POINTER1),Y:STA OSC1LF
506 INC POINTER1:BNE *+4:INC POINTER1+1
507 LDY #0:LDA (POINTER1),Y:STA OSC1HF
508 INC POINTER1:BNE *+4:INC POINTER1+1
510 LDA #0:STA COUNTER1
511 LDA WAVE1:ORA #1:STA OSC1WV
512 JMP LOOP2
520 GATE1 LDA COUNTER1:ASL
521 ^     LDY #0:CMP (POINTER1),Y
522 ^     BCC LOOP2
523 ^     LDA WAVE1:STA OSC1WV
600 LOOP2 INC COUNTER2
601 LDA COUNTER2:LDY #0:CMP (POINTER2),Y
602 BCC GATE2
604 INC POINTER2:BNE *+4:INC POINTER2+1
605 LDY #0:LDA (POINTER2),Y:STA OSC2LF
606 INC POINTER2:BNE *+4:INC POINTER2+1
607 LDY #0:LDA (POINTER2),Y:STA OSC2HF
608 INC POINTER2:BNE *+4:INC POINTER2+1
610 LDA #0:STA COUNTER2
611 LDA WAVE2:ORA #1:STA OSC2WV
612 JMP LOOP3
620 GATE2 LDA COUNTER2:ASL
621 ^     LDY #0:CMP (POINTER2),Y
622 ^     BCC LOOP3
623 ^     LDA WAVE2:STA OSC2WV
700 LOOP3 INC COUNTER3
701 LDA COUNTER3:LDY #0:CMP (POINTER3),Y
702 BCC GATE3
704 INC POINTER3:BNE *+4:INC POINTER3+1
705 LDY #0:LDA (POINTER3),Y:STA OSC3LF
706 INC POINTER3:BNE *+4:INC POINTER3+1
707 LDY #0:LDA (POINTER3),Y:STA OSC3HF
708 INC POINTER3:BNE *+4:INC POINTER3+1
709 LDY #0:LDA (POINTER3),Y:BNE *+3:RTS
710 LDA #0:STA COUNTER3
```

```
711 LDA WAVE3:ORA #1:STA OSC3WV
712 JMP FLAG
720 GATE3 LDA COUNTER3:ASL
721 ^     LDY #0:CMP (POINTER3),Y
722 ^     BCC FLAG
723 ^     LDA WAVE3:STA OSC3WV
800 FLAG *=*
802 LDA POINTER1  :CMP #<POS11-1:BNE N1
803 LDA POINTER1+1:CMP #>POS11-1:BNE N1
804 LDY #0 :LDA (POINTER1),Y
805 TAX:DEX:CPX COUNTER1:BNE N1
806 INC FLAG1
807 LDA FLAG1:CMP #4:BEQ N1
810 LDA #<STIMME1:STA POINTER1
811 LDA #>STIMME1:STA POINTER1+1
820 N1  *=*
822 LDA POINTER2  :CMP #<POS21-1:BNE N2
823 LDA POINTER2+1:CMP #>POS21-1:BNE N2
824 LDY #0 :LDA (POINTER2),Y
825 TAX:DEX:CPX COUNTER2:BNE N2
826 INC FLAG2
827 LDA FLAG2:CMP #4:BEQ N2
830 LDA #<STIMME2:STA POINTER2
831 LDA #>STIMME2:STA POINTER2+1
840 N2  *=*
841 LDA POINTER2  :CMP #<POS22-1:BNE N3
842 LDA POINTER2+1:CMP #>POS22-1:BNE N3
843 LDY #0 :LDA (POINTER2),Y
844 TAX:DEX:CPX COUNTER2:BNE N3
845 LDA #180:STA SPEED
850 N3  *=*
851 LDA POINTER1  :CMP #<POS12-1:BNE N4
852 LDA POINTER1+1:CMP #>POS12-1:BNE N4
853 LDY #0 :LDA (POINTER1),Y
854 TAX:DEX:CPX COUNTER1:BNE N4
855 LDA #0:STA FLAG1
856 LDA #1:STA FLAGX
857 LDA #<STIMME1:STA POINTER1
858 LDA #>STIMME1:STA POINTER1+1
```

```
859 LDA #120:STA SPEED
860 N4  *=*
861 LDA POINTER2  :CMP #<POS23-1:BNE N5
862 LDA POINTER2+1:CMP #>POS23-1:BNE N5
863 LDY #0 :LDA (POINTER2),Y
864 TAX:DEX:CPX COUNTER2:BNE N5
865 LDA #0:STA FLAG2
866 LDA #1:STA FLAGX
867 LDA #<STIMME2:STA POINTER2
868 LDA #>STIMME2:STA POINTER2+1
870 N5  *=*
871 LDA POINTER1  :CMP #<POS13-1:BNE N6
872 LDA POINTER1+1:CMP #>POS13-1:BNE N6
873 LDY #0 :LDA (POINTER1),Y
874 TAX:DEX:CPX COUNTER1:BNE N6
875 LDA FLAGX:BEQ N6
876 LDA #<NEW1:STA POINTER1
877 LDA #>NEW1:STA POINTER1+1
878 LDA #60:STA SPEED
880 N6  *=*
881 LDA POINTER2  :CMP #<POS24-1:BNE N7
882 LDA POINTER2+1:CMP #>POS24-1:BNE N7
883 LDY #0 :LDA (POINTER2),Y
884 TAX:DEX:CPX COUNTER2:BNE N7
885 LDA FLAGX:BEQ N7
886 LDA #<NEW2:STA POINTER2
887 LDA #>NEW2:STA POINTER2+1
890 ^    LDA #$25:STA OSC1AD
891 ^    LDA #$7A:STA OSC1SR
892 ^    LDA #$38:STA OSC2AD
893 ^    LDA #$98:STA OSC2SR
894 ^    LDA #$D0:STA FILTHF
895 ^    LDA #$03:STA FILMOD
896 ^    LDA #$06:STA OSC1PH
897 ^    LDA #$04:STA OSC2PH
898 ^    LDA #$40:STA WAVE1
899 ^             STA WAVE2
900 N7  *=*
902 LDA POINTER1  :CMP #<POS15-1:BNE N8
```

```
903 LDA POINTER1+1:CMP #>POS15-1:BNE N8
904 LDY #0 :LDA (POINTER1),Y
905 TAX:DEX:CPX COUNTER1:BNE N8
906 LDA #<POS14-1:STA POINTER1
907 LDA #>POS14-1:STA POINTER1+1
908 LDA #100:STA COUNTER1
910 N8  *=*
912 LDA POINTER2  :CMP #<POS26-1:BNE N9
913 LDA POINTER2+1:CMP #>POS26-1:BNE N9
914 LDY #0 :LDA (POINTER2),Y
915 TAX:DEX:CPX COUNTER2:BNE N9
916 LDA #<POS25-1:STA POINTER2
917 LDA #>POS25-1:STA POINTER2+1
918 LDA #100:STA COUNTER2
920 N9  *=*
990 WAIT  LDX SPEED:LDY #0
991 LOOP  DEY:BNE LOOP:DEX:BNE LOOP
992 JMP LOOP1
1000 STIMME1 .BYTE 1
1001 .WORD 0:.BYTE 2
1002 .WORD FS4:.BYTE 2
1003 .WORD CS5:.BYTE 2
1004 .WORD D5:.BYTE 2
1005 .WORD 0:.BYTE 2
1006 .WORD FS4:.BYTE 2
1007 .WORD CS5:.BYTE 2
1008 .WORD D5:.BYTE 2
1009 .WORD 0:.BYTE 2
1010 .WORD FS4:.BYTE 2
1011 .WORD CS5:.BYTE 2
1012 .WORD D5:.BYTE 2
1013 .WORD CS5:.BYTE 2
1014 .WORD D5:.BYTE 2
1015 .WORD CS5:.BYTE 2
1016 .WORD D5:.BYTE 2
1017 .WORD 0:.BYTE 2
1018 .WORD G4:.BYTE 2
1019 .WORD CS5:.BYTE 2
1020 .WORD D5:.BYTE 2
```

```
1021 .WORD 0:.BYTE 2
1022 .WORD G4:.BYTE 2
1023 .WORD CS5:.BYTE 2
1024 .WORD D5:.BYTE 2
1025 .WORD 0:.BYTE 2
1026 .WORD G4:.BYTE 2
1027 .WORD CS5:.BYTE 2
1028 .WORD D5:.BYTE 2
1029 .WORD CS5:.BYTE 2
1030 .WORD D5:.BYTE 2
1031 .WORD CS5:.BYTE 2
1032 .WORD D5:.BYTE 2
1033 POS11 *=*
1034 .WORD A5:.BYTE 2
1035 .WORD F5:.BYTE 2
1036 .WORD E5:.BYTE 2
1037 .WORD F5:.BYTE 2
1038 .WORD D5:.BYTE 2
1039 .WORD F5:.BYTE 2
1040 .WORD E5:.BYTE 2
1041 .WORD F5:.BYTE 2
1042 .WORD A5:.BYTE 2
1043 .WORD F5:.BYTE 2
1044 .WORD E5:.BYTE 2
1045 .WORD F5:.BYTE 2
1046 .WORD E5:.BYTE 2
1047 .WORD F5:.BYTE 2
1048 .WORD E5:.BYTE 2
1049 .WORD F5:.BYTE 2
1050 .WORD B5:.BYTE 2
1051 .WORD F5:.BYTE 2
1052 .WORD E5:.BYTE 2
1053 .WORD F5:.BYTE 2
1054 .WORD D5:.BYTE 2
1055 .WORD F5:.BYTE 2
1056 .WORD E5:.BYTE 2
1057 .WORD F5:.BYTE 2
1058 .WORD B5:.BYTE 2
1059 .WORD F5:.BYTE 2
1060 .WORD E5:.BYTE 2
1061 .WORD F5:.BYTE 2
1062 .WORD D5:.BYTE 2
1063 .WORD F5:.BYTE 2
1064 .WORD E5:.BYTE 2
1065 .WORD D5:.BYTE 2
1066 .WORD 0:.BYTE 2
1067 .WORD F4:.BYTE 2
1068 .WORD D5:.BYTE 2
1069 .WORD B4:.BYTE 2
1070 .WORD 0:.BYTE 2
1071 .WORD G4:.BYTE 2
1072 .WORD E5:.BYTE 2
1073 .WORD C5:.BYTE 2
1074 POS13 *=*
1075 .WORD G5:.BYTE 2
1076 .WORD E5:.BYTE 2
1077 .WORD C5:.BYTE 2
1078 .WORD A4:.BYTE 2
1079 .WORD F4:.BYTE 2
1080 .WORD C5:.BYTE 2
1081 .WORD A4:.BYTE 4
1082 POS12 *=*
1083 NEW1 .BYTE 1
1084 .WORD G2:.BYTE 16
1085 .WORD G2:.BYTE 16
1086 .WORD G2:.BYTE 16
1087 .WORD G2:.BYTE 16
1088 .WORD G2:.BYTE 16
1089 .WORD G2:.BYTE 16
1090 .WORD G2:.BYTE 16
1091 .WORD G2:.BYTE 16
1092 .WORD G2:.BYTE 16
1093 .WORD G2:.BYTE 16
1094 .WORD G2:.BYTE 16
1095 .WORD G2:.BYTE 16
1096 .WORD G2:.BYTE 16
1097 .WORD G2:.BYTE 16
1098 .WORD G2:.BYTE 16
```

```
1099 .WORD G2:.BYTE 16
1100 .WORD F2:.BYTE 16
1101 .WORD F2:.BYTE 16
1102 .WORD E2:.BYTE 16
1103 .WORD E2:.BYTE 8
1104 .WORD D2:.BYTE 8
1105 .WORD C2:.BYTE 16
1106 .WORD C2:.BYTE 8
1107 .WORD H1:.BYTE 8
1108 .WORD CS2:.BYTE 32
1109 .WORD CS3:.BYTE 8
1110 .WORD H2:.BYTE 8
1111 .WORD A2:.BYTE 16
1112 .WORD A2:.BYTE 16
1113 .WORD A2:.BYTE 16
1114 .WORD A2:.BYTE 6
1115 .WORD G2:.BYTE 18
1116 .WORD FS2:.BYTE 8
1117 POS14 *=*
1118 .WORD 0:.BYTE 2
1119 .WORD D4:.BYTE 2
1120 .WORD F4:.BYTE 6
1121 .WORD D4:.BYTE 2
1122 .WORD F4:.BYTE 2
1123 .WORD D4:.BYTE 2
1124 .WORD F4:.BYTE 2
1125 .WORD D4:.BYTE 2
1126 .WORD F4:.BYTE 2
1127 .WORD D4:.BYTE 2
1128 .WORD F4:.BYTE 2
1129 .WORD D4:.BYTE 2
1130 .WORD F4:.BYTE 2
1131 .WORD D4:.BYTE 2
1132 .WORD 0:.BYTE 2
1133 .WORD D4:.BYTE 2
1134 .WORD F4:.BYTE 6
1135 .WORD D4:.BYTE 2
1136 .WORD F4:.BYTE 2
1137 .WORD D4:.BYTE 2
1138 .WORD F4:.BYTE 2
1139 .WORD D4:.BYTE 2
1140 .WORD F4:.BYTE 2
1141 .WORD D4:.BYTE 2
1142 .WORD F4:.BYTE 2
1143 .WORD D4:.BYTE 2
1144 .WORD F4:.BYTE 2
1145 .WORD D4:.BYTE 2
1146 .WORD 0:.BYTE 2
1147 .WORD D4:.BYTE 2
1148 .WORD F4:.BYTE 6
1149 .WORD D4:.BYTE 2
1150 .WORD F4:.BYTE 2
1151 .WORD D4:.BYTE 2
1152 .WORD F4:.BYTE 2
1153 .WORD D4:.BYTE 2
1154 .WORD F4:.BYTE 2
1155 .WORD D4:.BYTE 2
1156 .WORD F4:.BYTE 2
1157 .WORD D4:.BYTE 2
1158 .WORD F4:.BYTE 2
1159 .WORD D4:.BYTE 2
1160 .WORD 0:.BYTE 2
1161 .WORD D4:.BYTE 2
1162 .WORD F4:.BYTE 6
1163 .WORD D4:.BYTE 2
1164 .WORD F4:.BYTE 2
1165 .WORD D4:.BYTE 2
1166 .WORD F4:.BYTE 2
1167 .WORD D4:.BYTE 2
1168 .WORD F4:.BYTE 2
1169 .WORD D4:.BYTE 2
1170 .WORD F4:.BYTE 2
1171 .WORD D4:.BYTE 2
1172 .WORD F4:.BYTE 2
1173 .WORD D4:.BYTE 2
1174 .WORD 0:.BYTE 2
1175 .WORD A4:.BYTE 4
1176 .WORD C5:.BYTE 4
```

```
1177 .WORD A4:.BYTE 2
1178 .WORD C5:.BYTE 2
1179 .WORD A4:.BYTE 2
1180 .WORD C5:.BYTE 2
1181 .WORD A4:.BYTE 2
1182 .WORD C5:.BYTE 2
1183 .WORD A4:.BYTE 2
1184 .WORD C5:.BYTE 2
1185 .WORD A4:.BYTE 2
1186 .WORD C5:.BYTE 2
1187 .WORD A4:.BYTE 2
1188 .WORD 0:.BYTE 2
1189 .WORD A4:.BYTE 4
1190 .WORD C5:.BYTE 4
1191 .WORD A4:.BYTE 2
1192 .WORD C5:.BYTE 2
1193 .WORD A4:.BYTE 2
1194 .WORD C5:.BYTE 2
1195 .WORD A4:.BYTE 2
1196 .WORD C5:.BYTE 2
1197 .WORD A4:.BYTE 2
1198 .WORD C5:.BYTE 2
1199 .WORD A4:.BYTE 2
1200 .WORD C5:.BYTE 2
1201 .WORD A4:.BYTE 2
1202 .WORD 0:.BYTE 2
1203 .WORD A4:.BYTE 4
1204 .WORD C5:.BYTE 4
1205 .WORD A4:.BYTE 2
1206 .WORD C5:.BYTE 2
1207 .WORD A4:.BYTE 2
1208 .WORD C5:.BYTE 2
1209 .WORD A4:.BYTE 2
1210 .WORD C5:.BYTE 2
1211 .WORD A4:.BYTE 2
1212 .WORD C5:.BYTE 2
1213 .WORD A4:.BYTE 2
1214 .WORD C5:.BYTE 2
1215 .WORD A4:.BYTE 2
1216 .WORD 0:.BYTE 2
1217 .WORD A4:.BYTE 4
1218 .WORD C5:.BYTE 4
1219 .WORD A4:.BYTE 2
1220 .WORD C5:.BYTE 2
1221 .WORD A4:.BYTE 2
1222 .WORD C5:.BYTE 2
1223 .WORD A4:.BYTE 2
1224 .WORD C5:.BYTE 2
1225 .WORD A4:.BYTE 2
1226 .WORD C5:.BYTE 2
1227 .WORD A4:.BYTE 2
1228 .WORD C5:.BYTE 2
1229 .WORD A4:.BYTE 2
1230 POS15 *=*
2000 STIMME2 .BYTE 1
2001 .WORD A3:.BYTE 4
2002 .WORD D4:.BYTE 4
2003 .WORD A3:.BYTE 4
2004 .WORD D4:.BYTE 4
2005 .WORD A3:.BYTE 4
2006 .WORD D4:.BYTE 4
2007 .WORD D4:.BYTE 4
2008 .WORD D4:.BYTE 4
2009 .WORD A3:.BYTE 4
2010 .WORD E4:.BYTE 4
2011 .WORD A3:.BYTE 4
2012 .WORD E4:.BYTE 4
2013 .WORD A3:.BYTE 4
2014 .WORD E4:.BYTE 4
2015 .WORD E4:.BYTE 4
2016 .WORD E4:.BYTE 4
2017 POS21 *=*
2018 .WORD D3:.BYTE 2
2019 .WORD A3:.BYTE 2
2020 .WORD D4:.BYTE 2
2021 .WORD A3:.BYTE 2
2022 .WORD D3:.BYTE 2
2023 .WORD A3:.BYTE 2
```

```
2024 .WORD E4:.BYTE 2
2025 .WORD D4:.BYTE 2
2026 .WORD D3:.BYTE 2
2027 .WORD A3:.BYTE 2
2028 .WORD D4:.BYTE 2
2029 .WORD A3:.BYTE 2
2030 .WORD D3:.BYTE 2
2031 .WORD A3:.BYTE 2
2032 .WORD E4:.BYTE 2
2033 .WORD D4:.BYTE 2
2034 .WORD D3:.BYTE 2
2035 .WORD B3:.BYTE 2
2036 .WORD F4:.BYTE 2
2037 .WORD D4:.BYTE 2
2038 .WORD D3:.BYTE 2
2039 .WORD B3:.BYTE 2
2040 .WORD F4:.BYTE 2
2041 .WORD D4:.BYTE 2
2042 .WORD D3:.BYTE 2
2043 .WORD B4:.BYTE 2
2044 .WORD F4:.BYTE 2
2045 .WORD D4:.BYTE 2
2046 .WORD D3:.BYTE 2
2047 .WORD B3:.BYTE 2
2048 .WORD F4:.BYTE 2
2049 .WORD D4:.BYTE 2
2050 POS22 *=*
2051 .WORD D3:.BYTE 2
2052 .WORD B3:.BYTE 2
2053 .WORD B3:.BYTE 2
2054 .WORD B3:.BYTE 2
2055 .WORD D3:.BYTE 2
2056 .WORD G3:.BYTE 2
2057 .WORD G3:.BYTE 2
2058 .WORD G3:.BYTE 2
2059 POS24 *=*
2060 .WORD D3:.BYTE 16
2061 POS23 *=*
2062 NEW2 .BYTE 1
2063 .WORD D4:.BYTE 8
2064 .WORD D4:.BYTE 8
2065 .WORD D4:.BYTE 8
2066 .WORD D4:.BYTE 8
2067 .WORD C4:.BYTE 8
2068 .WORD C4:.BYTE 8
2069 .WORD C4:.BYTE 8
2070 .WORD C4:.BYTE 8
2071 .WORD D4:.BYTE 8
2072 .WORD D4:.BYTE 8
2073 .WORD D4:.BYTE 8
2074 .WORD D4:.BYTE 8
2075 .WORD C4:.BYTE 8
2076 .WORD C4:.BYTE 8
2077 .WORD C4:.BYTE 8
2078 .WORD C4:.BYTE 8
2079 .WORD H3:.BYTE 1
2080 .WORD C4:.BYTE 1
2081 .WORD D4:.BYTE 1
2082 .WORD G4:.BYTE 1
2083 .WORD H3:.BYTE 1
2084 .WORD C4:.BYTE 1
2085 .WORD D4:.BYTE 1
2086 .WORD G4:.BYTE 1
2087 .WORD H3:.BYTE 1
2088 .WORD C4:.BYTE 1
2089 .WORD D4:.BYTE 1
2090 .WORD G4:.BYTE 1
2091 .WORD H3:.BYTE 1
2092 .WORD C4:.BYTE 1
2093 .WORD D4:.BYTE 1
2094 .WORD G4:.BYTE 1
2095 .WORD H3:.BYTE 1
2096 .WORD C4:.BYTE 1
2097 .WORD D4:.BYTE 1
2098 .WORD G4:.BYTE 1
2099 .WORD H3:.BYTE 1
2100 .WORD C4:.BYTE 1
2101 .WORD D4:.BYTE 1
```

```
2102 .WORD G4:.BYTE 1
2103 .WORD H3:.BYTE 1
2104 .WORD C4:.BYTE 1
2105 .WORD D4:.BYTE 1
2106 .WORD G4:.BYTE 1
2107 .WORD H3:.BYTE 1
2108 .WORD C4:.BYTE 1
2109 .WORD D4:.BYTE 1
2110 .WORD G4:.BYTE 1
2111 .WORD A3:.BYTE 2
2112 .WORD H3:.BYTE 2
2113 .WORD C4:.BYTE 2
2114 .WORD H3:.BYTE 4
2115 .WORD A3:.BYTE 2
2116 .WORD C4:.BYTE 2
2117 .WORD E4:.BYTE 2
2118 .WORD A3:.BYTE 2
2119 .WORD H3:.BYTE 2
2120 .WORD C4:.BYTE 2
2121 .WORD H3:.BYTE 4
2122 .WORD A3:.BYTE 2
2123 .WORD C4:.BYTE 2
2124 .WORD E4:.BYTE 2
2125 .WORD D4:.BYTE 8
2126 .WORD D4:.BYTE 8
2127 .WORD D4:.BYTE 8
2128 .WORD D4:.BYTE 8
2129 .WORD C4:.BYTE 8
2130 .WORD C4:.BYTE 8
2131 .WORD C4:.BYTE 8
2132 .WORD C4:.BYTE 8
2133 .WORD A3:.BYTE 8
2134 .WORD A3:.BYTE 8
2135 .WORD A3:.BYTE 8
2136 .WORD A3:.BYTE 8
2137 .WORD G3:.BYTE 8
2138 .WORD G3:.BYTE 8
2139 .WORD G3:.BYTE 8
2140 .WORD FS3:.BYTE 8
2141 .WORD E3:.BYTE 8
2142 .WORD E3:.BYTE 8
2143 .WORD E3:.BYTE 8
2144 .WORD DS3:.BYTE 8
2145 .WORD E3:.BYTE 8
2146 .WORD E3:.BYTE 8
2147 .WORD E3:.BYTE 8
2148 .WORD E3:.BYTE 8
2149 .WORD E3:.BYTE 8
2150 .WORD DS3:.BYTE 8
2151 .WORD CS3:.BYTE 8
2152 .WORD CS3:.BYTE 8
2153 .WORD CS3:.BYTE 8
2154 .WORD CS3:.BYTE 8
2155 .WORD CS3:.BYTE 8
2156 .WORD CS3:.BYTE 8
2157 .WORD CS3:.BYTE 6
2158 .WORD CS3:.BYTE 10
2159 .WORD C3:.BYTE 8
2160 .WORD DS3:.BYTE 8
2161 POS25 *=*
2162 .WORD B4:.BYTE 6
2163 .WORD C5:.BYTE 6
2164 .WORD B4:.BYTE 6
2165 .WORD C5:.BYTE 6
2166 .WORD B4:.BYTE 4
2167 .WORD C5:.BYTE 4
2168 .WORD B4:.BYTE 6
2169 .WORD C5:.BYTE 6
2170 .WORD B4:.BYTE 6
2171 .WORD C5:.BYTE 6
2172 .WORD B4:.BYTE 4
2173 .WORD C5:.BYTE 4
2174 .WORD B4:.BYTE 6
2175 .WORD C5:.BYTE 6
2176 .WORD B4:.BYTE 6
2177 .WORD C5:.BYTE 6
2178 .WORD B4:.BYTE 4
2179 .WORD C5:.BYTE 4
```

```
2180 .WORD B4:.BYTE 6
2181 .WORD C5:.BYTE 6
2182 .WORD B4:.BYTE 6
2183 .WORD C5:.BYTE 6
2184 .WORD B4:.BYTE 4
2185 .WORD C5:.BYTE 4
2186 .WORD F5:.BYTE 6
2187 .WORD G5:.BYTE 6
2188 .WORD F5:.BYTE 6
2189 .WORD E5:.BYTE 6
2190 .WORD F5:.BYTE 4
2191 .WORD E5:.BYTE 4
2192 .WORD F5:.BYTE 6
2193 .WORD G5:.BYTE 6
2194 .WORD F5:.BYTE 6
2195 .WORD E5:.BYTE 6
2196 .WORD F5:.BYTE 4
2197 .WORD E5:.BYTE 4
2198 .WORD F5:.BYTE 6
2199 .WORD G5:.BYTE 6
2200 .WORD F5:.BYTE 6
2201 .WORD E5:.BYTE 6
2202 .WORD F5:.BYTE 4
2203 .WORD E5:.BYTE 4
2204 .WORD F5:.BYTE 6
2205 .WORD G5:.BYTE 6
2206 .WORD F5:.BYTE 6
2207 .WORD E5:.BYTE 6
2208 .WORD F5:.BYTE 4
2209 .WORD E5:.BYTE 4
2210 POS26 *=*
3000 STIMME3 .BYTE 1
3001 .WORD 0:.BYTE 128
3002 .WORD D4:.BYTE 6
3003 .WORD FS4:.BYTE 2
3004 .WORD FS4:.BYTE 6
3005 .WORD D5:.BYTE 2
3006 .WORD D5:.BYTE 6
3007 .WORD A4:.BYTE 2
3008 .WORD A4:.BYTE 8
3009 .WORD A4:.BYTE 6
3010 .WORD G4:.BYTE 2
3011 .WORD FS4:.BYTE 4
3012 .WORD E4:.BYTE 4
3013 .WORD E4:.BYTE 8
3014 .WORD 0:.BYTE 6
3015 .WORD E4:.BYTE 2
3016 .WORD D4:.BYTE 4
3017 .WORD FS4:.BYTE 4
3018 .WORD FS4:.BYTE 6
3019 .WORD D5:.BYTE 2
3020 .WORD D5:.BYTE 4
3021 .WORD A4:.BYTE 4
3022 .WORD A4:.BYTE 6
3023 .WORD A4:.BYTE 2
3024 .WORD A4:.BYTE 2
3025 .WORD G4:.BYTE 6
3026 .WORD 0:.BYTE 4
3027 .WORD FS4:.BYTE 2
3028 .WORD E4:.BYTE 2
3029 .WORD E4:.BYTE 8
3030 .WORD 0:.BYTE 8
3031 .WORD D4:.BYTE 12
3032 .WORD E4:.BYTE 2
3033 .WORD F4:.BYTE 4
3034 .WORD A4:.BYTE 4
3035 .WORD D4:.BYTE 8
3036 .WORD A4:.BYTE 2
3037 .WORD G4:.BYTE 10
3038 .WORD A4:.BYTE 2
3039 .WORD G4:.BYTE 3
3040 .WORD A4:.BYTE 1
3041 .WORD B4:.BYTE 16
3042 .WORD 0:.BYTE 2
3043 .WORD B4:.BYTE 2
3044 .WORD C5:.BYTE 2
3045 .WORD D5:.BYTE 2
3046 .WORD 0:.BYTE 2
```

```
3047 .WORD E5:.BYTE 2
3048 .WORD F5:.BYTE 2
3049 .WORD G5:.BYTE 2
3050 .WORD G5:.BYTE 2
3051 .WORD A5:.BYTE 2
3052 .WORD A5:.BYTE 4
3054 .WORD A5:.BYTE 8
3055 .WORD FS5:.BYTE 4
3056 .WORD A5:.BYTE 4
3057 .WORD A5:.BYTE 8
3058 .WORD A5:.BYTE 16
3059 .WORD 0:.BYTE 96
3060 .WORD D4:.BYTE 2
3061 .WORD D4:.BYTE 2
3062 .WORD D4:.BYTE 4
3063 .WORD 0:.BYTE 4
3064 .WORD FS4:.BYTE 2
3065 .WORD D5:.BYTE 2
3066 .WORD D5:.BYTE 4
3067 .WORD A4:.BYTE 2
3068 .WORD A4:.BYTE 8
3069 .WORD A4:.BYTE 2
3070 .WORD A4:.BYTE 4
3071 .WORD G4:.BYTE 2
3072 .WORD FS4:.BYTE 4
3073 .WORD E4:.BYTE 2
3074 .WORD E4:.BYTE 8
3075 .WORD 0:.BYTE 10
3076 .WORD D4:.BYTE 2
3077 .WORD D4:.BYTE 4
3078 .WORD FS4:.BYTE 2
3079 .WORD FS4:.BYTE 6
3080 .WORD 0:.BYTE 2
3081 .WORD D5:.BYTE 2
3082 .WORD D5:.BYTE 4
3083 .WORD A4:.BYTE 2
3084 .WORD A4:.BYTE 6
3085 .WORD 0:.BYTE 2
3086 .WORD A4:.BYTE 2
3087 .WORD A4:.BYTE 2
3088 .WORD G4:.BYTE 6
3089 .WORD 0:.BYTE 4
3090 .WORD FS4:.BYTE 2
3091 .WORD G4:.BYTE 2
3092 .WORD E4:.BYTE 8
3093 .WORD 0:.BYTE 8
3094 .WORD D4:.BYTE 8
3095 .WORD 0:.BYTE 4
3096 .WORD E4:.BYTE 2
3097 .WORD F4:.BYTE 4
3098 .WORD A4:.BYTE 4
3099 .WORD D4:.BYTE 6
3100 .WORD 0:.BYTE 2
3101 .WORD A4:.BYTE 2
3102 .WORD G4:.BYTE 3
3103 .WORD G4:.BYTE 1
3104 .WORD G4:.BYTE 8
3105 .WORD 0:.BYTE 2
3106 .WORD A4:.BYTE 2
3107 .WORD B4:.BYTE 8
3108 .WORD 0:.BYTE 8
3109 .WORD 0:.BYTE 2
3110 .WORD B4:.BYTE 2
3111 .WORD C5:.BYTE 2
3112 .WORD D5:.BYTE 2
3113 .WORD 0:.BYTE 2
3114 .WORD E5:.BYTE 2
3115 .WORD F5:.BYTE 2
3116 .WORD G5:.BYTE 2
3117 .WORD G5:.BYTE 4
3118 .WORD A5:.BYTE 8
3119 .WORD G5:.BYTE 2
3120 .WORD H5:.BYTE 10
3121 .WORD 0:.BYTE 8
3122 .WORD A5:.BYTE 4
3123 .WORD A5:.BYTE 4
3124 .WORD 0:.BYTE 20
3125 .WORD E5:.BYTE 4
```

```
3126 .WORD D5:.BYTE 2
3127 .WORD D5:.BYTE 2
3128 .WORD D5:.BYTE 2
3129 .WORD D5:.BYTE 2
3130 .WORD D5:.BYTE 4
3131 .WORD D5:.BYTE 4
3132 .WORD 0:.BYTE 12
3133 .WORD H4:.BYTE 4
3134 .WORD A4:.BYTE 4
3135 .WORD H4:.BYTE 4
3136 .WORD C5:.BYTE 8
3137 .WORD H4:.BYTE 4
3138 .WORD A4:.BYTE 2
3139 .WORD A4:.BYTE 6
3140 .WORD 0:.BYTE 68
3141 .WORD D5:.BYTE 2
3142 .WORD D5:.BYTE 2
3143 .WORD D5:.BYTE 2
3144 .WORD D5:.BYTE 2
3145 .WORD D5:.BYTE 6
3146 .WORD D5:.BYTE 2
3147 .WORD D5:.BYTE 8
3148 .WORD 0:.BYTE 4
3149 .WORD G4:.BYTE 4
3150 .WORD A4:.BYTE 6
3151 .WORD H4:.BYTE 4
3152 .WORD C5:.BYTE 6
3153 .WORD H4:.BYTE 6
3154 .WORD A4:.BYTE 10
3155 .WORD 0:.BYTE 6
3156 .WORD G4:.BYTE 2
3157 .WORD H4:.BYTE 4
3158 .WORD A4:.BYTE 4
3159 .WORD D5:.BYTE 4
3160 .WORD C5:.BYTE 2
3161 .WORD H4:.BYTE 6
3162 .WORD A4:.BYTE 2
3163 .WORD H4:.BYTE 10
3164 .WORD 0:.BYTE 12
3165 .WORD G4:.BYTE 2
3166 .WORD G4:.BYTE 2
3167 .WORD A4:.BYTE 2
3168 .WORD H4:.BYTE 4
3169 .WORD H4:.BYTE 4
3170 .WORD C5:.BYTE 4
3171 .WORD G4:.BYTE 4
3172 .WORD G4:.BYTE 6
3173 .WORD 0:.BYTE 8
3174 .WORD H4:.BYTE 6
3175 .WORD H4:.BYTE 2
3176 .WORD E5:.BYTE 12
3177 .WORD DS5:.BYTE 2
3178 .WORD DS5:.BYTE 2
3179 .WORD DS5:.BYTE 4
3180 .WORD CS5:.BYTE 8
3181 .WORD CS5:.BYTE 12
3182 .WORD H4:.BYTE 6
3183 .WORD A4:.BYTE 2
3184 .WORD A4:.BYTE 4
3185 .WORD A4:.BYTE 4
3186 .WORD 0:.BYTE 4
3187 .WORD DS5:.BYTE 2
3188 .WORD DS5:.BYTE 2
3189 .WORD DS5:.BYTE 4
3190 .WORD FS5:.BYTE 4
3191 .WORD H4:.BYTE 8
3192 .WORD 0:.BYTE 10
3193 .WORD A4:.BYTE 4
3194 .WORD A4:.BYTE 2
3195 .WORD A4:.BYTE 4
3196 .WORD A4:.BYTE 2
3197 .WORD G4:.BYTE 6
3198 .WORD G4:.BYTE 4
3199 .WORD G4:.BYTE 8
3200 .WORD A4:.BYTE 8
3201 .WORD G4:.BYTE 16
3202 .WORD 0:.BYTE 4
3203 .WORD B4:.BYTE 4
```

```
3204 .WORD C5:.BYTE 4
3205 .WORD B4:.BYTE 4
3206 .WORD G4:.BYTE 32
3207 .WORD 0:.BYTE 64
3208 .WORD D3:.BYTE 6
3209 .WORD A3:.BYTE 6
3210 .WORD C4:.BYTE 6
3211 .WORD A4:.BYTE 6
3212 .WORD C4:.BYTE 4
3213 .WORD A4:.BYTE 4
3214 .WORD D3:.BYTE 6
3215 .WORD A3:.BYTE 6
3216 .WORD C4:.BYTE 6
3217 .WORD A4:.BYTE 6
3218 .WORD C4:.BYTE 4
3219 .WORD A4:.BYTE 4
3220 .WORD D3:.BYTE 6
3221 .WORD A3:.BYTE 6
3222 .WORD C4:.BYTE 6
3223 .WORD A4:.BYTE 6
3224 .WORD C4:.BYTE 4
3225 .WORD A4:.BYTE 4
3226 .WORD D3:.BYTE 6
3227 .WORD A3:.BYTE 6
3228 .WORD C4:.BYTE 6
3229 .WORD A4:.BYTE 6
3230 .WORD C4:.BYTE 4
3231 .WORD A4:.BYTE 4
3232 .WORD G3:.BYTE 32
3233 .WORD G3:.BYTE 32
3234 .WORD G3:.BYTE 32
3235 .WORD G3:.BYTE 32
3236 .WORD D3:.BYTE 6
3237 .WORD A3:.BYTE 6
3238 .WORD C4:.BYTE 6
3239 .WORD A4:.BYTE 6
3240 .WORD C4:.BYTE 4
3241 .WORD A4:.BYTE 4
3242 .WORD D3:.BYTE 6
3243 .WORD A3:.BYTE 6
3244 .WORD C4:.BYTE 6
3245 .WORD A4:.BYTE 6
3246 .WORD C4:.BYTE 4
3247 .WORD A4:.BYTE 4
3248 .WORD D3:.BYTE 6
3249 .WORD A3:.BYTE 6
3250 .WORD C4:.BYTE 6
3251 .WORD A4:.BYTE 6
3252 .WORD C4:.BYTE 4
3253 .WORD A4:.BYTE 4
3254 .WORD D3:.BYTE 2
3999 .WORD 0,0,0,0

READY.
```

```
1000 POKE 56,128:CLR:FOR I = 32768 TO 35690
1010 READ X:POKE I,X:S = S+X:NEXT
1020 DATA 160,  0,169, 29,133,  2,169,128,133,  3,177,  2
1030 DATA 208,  3, 76,133,128, 32,210,255,230,  2,208,  2
1040 DATA 230,  3, 76, 10,128,147, 13, 13, 13, 13, 13, 13
1050 DATA  13, 13, 13, 13, 32, 32, 32, 32, 32, 32, 32, 32
1060 DATA  32, 32, 32, 32, 84, 72, 69, 32, 67, 73, 78, 69
1070 DATA  77, 65, 32, 83, 72, 79, 87, 13, 32, 32, 32, 32
1080 DATA  32, 32, 32, 32, 32, 32, 32, 77, 85, 83, 73, 67
1090 DATA  32, 66, 89, 58, 32, 71, 69, 78, 69, 83, 73, 83
1100 DATA  13, 32, 32, 32, 32, 32, 65, 82, 82, 65, 78, 71
1110 DATA  69, 77, 69, 78, 84, 32, 66, 89, 58, 32, 84, 72
1120 DATA  79, 77, 65, 83, 32, 68, 65, 67, 72, 83, 69, 76
1130 DATA   0,120, 32,144,128, 88,169,  0,141, 24,212, 96
1140 DATA 169, 31,141, 24,212,169, 37,141,  5,212,169,123
1150 DATA 141,  6,212,169, 37,141, 12,212,169,123,141, 13
1160 DATA 212,169, 73,141, 19,212,169,105,141, 20,212,169
1170 DATA   0,141, 23,212,169,102,133,  2,169,131,133,  3
1180 DATA 169,  8,133,  5,169,134,133,  6,169,107,133,  8
1190 DATA 169,136,133,  9,169,  0,133,  4,141,208,  2,133
1200 DATA   7,141,209,  2,133, 10,141,210,  2,169, 16,141
1210 DATA 192,  2,169, 16,141,193,  2,169, 32,141,194,  2
1220 DATA 169,120,141,224,  2,169,  0,141,225,  2,230,  4
1230 DATA 165,  4,160,  0,209,  2,144, 47,230,  2,208,  2
1240 DATA 230,  3,160,  0,177,  2,141,  0,212,230,  2,208
1250 DATA   2,230,  3,160,  0,177,  2,141,  1,212,230,  2
1260 DATA 208,  2,230,  3,169,  0,133,  4,173,192,  2,  9
1270 DATA   1,141,  4,212, 76, 66,129,165,  4, 10,160,  0
1280 DATA 209,  2,144,  6,173,192,  2,141,  4,212,230,  7
1290 DATA 165,  7,160,  0,209,  5,144, 47,230,  5,208,  2
1300 DATA 230,  6,160,  0,177,  5,141,  7,212,230,  5,208
1310 DATA   2,230,  6,160,  0,177,  5,141,  8,212,230,  5
1320 DATA 208,  2,230,  6,169,  0,133,  7,173,193,  2,  9
1330 DATA   1,141, 11,212, 76,138,129,165,  7, 10,160,  0
1340 DATA 209,  5,144,  6,173,193,  2,141, 11,212,230, 10
1350 DATA 165, 10,160,  0,209,  8,144, 54,230,  8,208,  2
```

```
1360 DATA 230,  9,160,  0,177,  8,141, 14,212,230,  8,208
1370 DATA   2,230,  9,160,  0,177,  8,141, 15,212,230,  8
1380 DATA 208,  2,230,  9,160,  0,177,  8,208,  1, 96,169
1390 DATA   0,133, 10,173,194,  2,  9,  1,141, 18,212, 76
1400 DATA 217,129,165, 10, 10,160,  0,209,  8,144,  6,173
1410 DATA 194,  2,141, 18,212,165,  2,201,198,208, 34,165
1420 DATA   3,201,131,208, 28,160,  0,177,  2,170,202,228
1430 DATA   4,208, 18,238,208,  2,173,208,  2,201,  4,240
1440 DATA   8,169,102,133,  2,169,131,133,  3,165,  5,201
1450 DATA  56,208, 34,165,  6,201,134,208, 28,160,  0,177
1460 DATA   5,170,202,228,  7,208, 18,238,209,  2,173,209
1470 DATA   2,201,  4,240,  8,169,  8,133,  5,169,134,133
1480 DATA   6,165,  5,201,152,208, 21,165,  6,201,134,208
1490 DATA  15,160,  0,177,  5,170,202,228,  7,208,  5,169
1500 DATA 180,141,224,  2,165,  2,201, 83,208, 39,165,  3
1510 DATA 201,132,208, 33,160,  0,177,  2,170,202,228,  4
1520 DATA 208, 23,169,  0,141,208,  2,169,  1,141,225,  2
1530 DATA 169,102,133,  2,169,131,133,  3,169,120,141,224
1540 DATA   2,165,  5,201,179,208, 34,165,  6,201,134,208
1550 DATA  28,160,  0,177,  5,170,202,228,  7,208, 18,169
1560 DATA   0,141,209,  2,169,  1,141,225,  2,169,  8,133
1570 DATA   5,169,134,133,  6,165,  2,201, 62,208, 34,165
1580 DATA   3,201,132,208, 28,160,  0,177,  2,170,202,228
1590 DATA   4,208, 18,173,225,  2,240, 13,169, 84,133,  2
1600 DATA 169,132,133,  3,169, 60,141,224,  2,165,  5,201
1610 DATA 176,208, 77,165,  6,201,134,208, 71,160,  0,177
1620 DATA   5,170,202,228,  7,208, 61,173,225,  2,240, 56
1630 DATA 169,180,133,  5,169,134,133,  6,169, 37,141,  5
1640 DATA 212,169,122,141,  6,212,169, 56,141, 12,212,169
1650 DATA 152,141, 13,212,169,208,141, 22,212,169,  3,141
1660 DATA  23,212,169,  6,141,  3,212,169,  4,141, 10,212
1670 DATA 169, 64,141,192,  2,141,193,  2,165,  2,201,  7
1680 DATA 208, 28,165,  3,201,134,208, 22,160,  0,177,  2
1690 DATA 170,202,228,  4,208, 12,169,183,133,  2,169,132
1700 DATA 133,  3,169,100,133,  4,165,  5,201,106,208, 28
1710 DATA 165,  6,201,136,208, 22,160,  0,177,  5,170,202
1720 DATA 228,  7,208, 12,169,218,133,  5,169,135,133,  6
1730 DATA 169,100,133,  7,174,224,  2,160,  0,136,208,253
1740 DATA 202,208,250, 76,250,128,  1,  0,  0,  2,181, 23
```

```
1750 DATA   2,134, 35,  2,162, 37,  2,  0,  0,  2,181, 23
1760 DATA   2,134, 35,  2,162, 37,  2,  0,  0,  2,181, 23
1770 DATA   2,134, 35,  2,162, 37,  2,134, 35,  2,162, 37
1780 DATA   2,134, 35,  2,162, 37,  2,  0,  0,  2, 30, 25
1790 DATA   2,134, 35,  2,162, 37,  2,  0,  0,  2, 30, 25
1800 DATA   2,134, 35,  2,162, 37,  2,  0,  0,  2, 30, 25
1810 DATA   2,134, 35,  2,162, 37,  2,134, 35,  2,162, 37
1820 DATA   2,134, 35,  2,162, 37,  2, 99, 56,  2,193, 44
1830 DATA   2, 62, 42,  2,193, 44,  2,162, 37,  2,193, 44
1840 DATA   2, 62, 42,  2,193, 44,  2, 99, 56,  2,193, 44
1850 DATA   2, 62, 42,  2,193, 44,  2, 62, 42,  2,193, 44
1860 DATA   2, 62, 42,  2,193, 44,  2,190, 59,  2,193, 44
1870 DATA   2, 62, 42,  2,193, 44,  2,162, 37,  2,193, 44
1880 DATA   2, 62, 42,  2,193, 44,  2,190, 59,  2,193, 44
1890 DATA   2, 62, 42,  2,193, 44,  2,162, 37,  2,193, 44
1900 DATA   2, 62, 42,  2,162, 37,  2,  0,  0,  2, 96, 22
1910 DATA   2,162, 37,  2,223, 29,  2,  0,  0,  2, 30, 25
1920 DATA   2, 62, 42,  2,135, 33,  2, 60, 50,  2, 62, 42
1930 DATA   2,135, 33,  2, 49, 28,  2, 96, 22,  2,135, 33
1940 DATA   2, 49, 28,  4,  1, 71,  6, 16, 71,  6, 16, 71
1950 DATA   6, 16, 71,  6, 16, 71,  6, 16, 71,  6, 16, 71
1960 DATA   6, 16, 71,  6, 16, 71,  6, 16, 71,  6, 16, 71
1970 DATA   6, 16, 71,  6, 16, 71,  6, 16, 71,  6, 16, 71
1980 DATA   6, 16, 71,  6, 16,152,  5, 16,152,  5, 16, 71
1990 DATA   5, 16, 71,  5,  8,180,  4,  8, 48,  4, 16, 48
2000 DATA   4,  8,244,  3,  8,112,  4, 32,225,  8,  8,233
2010 DATA   7,  8, 12,  7, 16, 12,  7, 16, 12,  7, 16, 12
2020 DATA   7,  6, 71,  6, 18,237,  5,  8,  0,  0,  2,209
2030 DATA  18,  2, 96, 22,  6,209, 18,  2, 96, 22,  2,209
2040 DATA  18,  2, 96, 22,  2,209, 18,  2, 96, 22,  2,209
2050 DATA  18,  2, 96, 22,  2,209, 18,  2, 96, 22,  2,209
2060 DATA  18,  2,  0,  0,  2,209, 18,  2, 96, 22,  6,209
2070 DATA  18,  2, 96, 22,  2,209, 18,  2, 96, 22,  2,209
2080 DATA  18,  2, 96, 22,  2,209, 18,  2, 96, 22,  2,209
2090 DATA  18,  2, 96, 22,  2,209, 18,  2,  0,  0,  2,209
2100 DATA  18,  2, 96, 22,  6,209, 18,  2, 96, 22,  2,209
2110 DATA  18,  2, 96, 22,  2,209, 18,  2, 96, 22,  2,209
2120 DATA  18,  2, 96, 22,  2,209, 18,  2, 96, 22,  2,209
2130 DATA  18,  2,  0,  0,  2,209, 18,  2, 96, 22,  6,209
```

```
2140 DATA  18,  2, 96, 22,  2,209, 18,  2, 96, 22,  2,209
2150 DATA  18,  2, 96, 22,  2,209, 18,  2, 96, 22,  2,209
2160 DATA  18,  2, 96, 22,  2,209, 18,  2,  0,  0,  2, 49
2170 DATA  28,  4,135, 33,  4, 49, 28,  2,135, 33,  2, 49
2180 DATA  28,  2,135, 33,  2, 49, 28,  2,135, 33,  2, 49
2190 DATA  28,  2,135, 33,  2, 49, 28,  2,135, 33,  2, 49
2200 DATA  28,  2,  0,  0,  2, 49, 28,  4,135, 33,  4, 49
2210 DATA  28,  2,135, 33,  2, 49, 28,  2,135, 33,  2, 49
2220 DATA  28,  2,135, 33,  2, 49, 28,  2,135, 33,  2, 49
2230 DATA  28,  2,135, 33,  2, 49, 28,  2,  0,  0,  2, 49
2240 DATA  28,  4,135, 33,  4, 49, 28,  2,135, 33,  2, 49
2250 DATA  28,  2,135, 33,  2, 49, 28,  2,135, 33,  2, 49
2260 DATA  28,  2,135, 33,  2, 49, 28,  2,135, 33,  2, 49
2270 DATA  28,  2,  0,  0,  2, 49, 28,  4,135, 33,  4, 49
2280 DATA  28,  2,135, 33,  2, 49, 28,  2,135, 33,  2, 49
2290 DATA  28,  2,135, 33,  2, 49, 28,  2,135, 33,  2, 49
2300 DATA  28,  2,135, 33,  2, 49, 28,  2,  1, 24, 14,  4
2310 DATA 209, 18,  4, 24, 14,  4,209, 18,  4, 24, 14,  4
2320 DATA 209, 18,  4,209, 18,  4,209, 18,  4, 24, 14,  4
2330 DATA  31, 21,  4, 24, 14,  4, 31, 21,  4, 24, 14,  4
2340 DATA  31, 21,  4, 31, 21,  4, 31, 21,  4,104,  9,  2
2350 DATA  24, 14,  2,209, 18,  2, 24, 14,  2,104,  9,  2
2360 DATA  24, 14,  2, 31, 21,  2,209, 18,  2,104,  9,  2
2370 DATA  24, 14,  2,209, 18,  2, 24, 14,  2,104,  9,  2
2380 DATA  24, 14,  2, 31, 21,  2,209, 18,  2,104,  9,  2
2390 DATA 239, 14,  2, 96, 22,  2,209, 18,  2,104,  9,  2
2400 DATA 239, 14,  2, 96, 22,  2,209, 18,  2,104,  9,  2
2410 DATA 223, 29,  2, 96, 22,  2,209, 18,  2,104,  9,  2
2420 DATA 239, 14,  2, 96, 22,  2,209, 18,  2,104,  9,  2
2430 DATA 239, 14,  2,239, 14,  2,239, 14,  2,104,  9,  2
2440 DATA 143, 12,  2,143, 12,  2,143, 12,  2,104,  9, 16
2450 DATA   1,209, 18,  8,209, 18,  8,209, 18,  8,209, 18
2460 DATA   8,195, 16,  8,195, 16,  8,195, 16,  8,195, 16
2470 DATA   8,209, 18,  8,209, 18,  8,209, 18,  8,209, 18
2480 DATA   8,195, 16,  8,195, 16,  8,195, 16,  8,195, 16
2490 DATA   8,210, 15,  1,195, 16,  1,209, 18,  1, 30, 25
2500 DATA   1,210, 15,  1,195, 16,  1,209, 18,  1, 30, 25
2510 DATA   1,210, 15,  1,195, 16,  1,209, 18,  1, 30, 25
2520 DATA   1,210, 15,  1,195, 16,  1,209, 18,  1, 30, 25
```

```
2530 DATA   1,210, 15,   1,195, 16,   1,209, 18,   1, 30, 25
2540 DATA   1,210, 15,   1,195, 16,   1,209, 18,   1, 30, 25
2550 DATA   1,210, 15,   1,195, 16,   1,209, 18,   1, 30, 25
2560 DATA   1,210, 15,   1,195, 16,   1,209, 18,   1, 30, 25
2570 DATA   1, 24, 14,   2,210, 15,   2,195, 16,   2,210, 15
2580 DATA   4, 24, 14,   2,195, 16,   2, 31, 21,   2, 24, 14
2590 DATA   2,210, 15,   2,195, 16,   2,210, 15,   4, 24, 14
2600 DATA   2,195, 16,   2, 31, 21,   2,209, 18,   8,209, 18
2610 DATA   8,209, 18,   8,209, 18,   8,195, 16,   8,195, 16
2620 DATA   8,195, 16,   8,195, 16,   8, 24, 14,   8, 24, 14
2630 DATA   8, 24, 14,   8, 24, 14,   8,143, 12,   8,143, 12
2640 DATA   8,143, 12,   8,218, 11,   8,143, 10,   8,143, 10
2650 DATA   8,143, 10,   8,247,  9,   8,143, 10,   8,143, 10
2660 DATA   8,143, 10,   8,143, 10,   8,143, 10,   8,247,  9
2670 DATA   8,225,  8,   8,225,  8,   8,225,  8,   8,225,  8
2680 DATA   8,225,  8,   8,225,  8,   8,225,  8,   6,225,  8
2690 DATA  10, 97,  8,   8,247,  9,   8,223, 29,   6,135, 33
2700 DATA   6,223, 29,   6,135, 33,   6,223, 29,   4,135, 33
2710 DATA   4,223, 29,   6,135, 33,   6,223, 29,   6,135, 33
2720 DATA   6,223, 29,   4,135, 33,   4,223, 29,   6,135, 33
2730 DATA   6,223, 29,   6,135, 33,   6,223, 29,   4,135, 33
2740 DATA   4,223, 29,   6,135, 33,   6,223, 29,   6,135, 33
2750 DATA   6,223, 29,   4,135, 33,   4,193, 44,   6, 60, 50
2760 DATA   6,193, 44,   6, 62, 42,   6,193, 44,   4, 62, 42
2770 DATA   4,193, 44,   6, 60, 50,   6,193, 44,   6, 62, 42
2780 DATA   6,193, 44,   4, 62, 42,   4,193, 44,   6, 60, 50
2790 DATA   6,193, 44,   6, 62, 42,   6,193, 44,   4, 62, 42
2800 DATA   4,193, 44,   6, 60, 50,   6,193, 44,   6, 62, 42
2810 DATA   6,193, 44,   4, 62, 42,   4,  1,  0,   0,128,209
2820 DATA  18,  6,181, 23,   2,181, 23,   6,162, 37,   2,162
2830 DATA  37,  6, 49, 28,   2, 49, 28,   8, 49, 28,   6, 30
2840 DATA  25,  2,181, 23,   4, 31, 21,   4, 31, 21,   8,  0
2850 DATA   0,  6, 31, 21,   2,209, 18,   4,181, 23,   4,181
2860 DATA  23,  6,162, 37,   2,162, 37,   4, 49, 28,   4, 49
2870 DATA  28,  6, 49, 28,   2, 49, 28,   2, 30, 25,   6,  0
2880 DATA   0,  4,181, 23,   2, 31, 21,   2, 31, 21,   8,  0
2890 DATA   0,  8,209, 18,  12, 31, 21,   2, 96, 22,   4, 49
2900 DATA  28,  4,209, 18,   8, 49, 28,   2, 30, 25,  10, 49
2910 DATA  28,  2, 30, 25,   3, 49, 28,   1,223, 29,  16,  0
```

```
2920 DATA   0,  2,223, 29,  2,135, 33,  2,162, 37,  2,  0
2930 DATA   0,  2, 62, 42,  2,193, 44,  2, 60, 50,  2, 60
2940 DATA  50,  2, 99, 56,  2, 99, 56,  4, 99, 56,  8,107
2950 DATA  47,  4, 99, 56,  4, 99, 56,  8, 99, 56, 16,  0
2960 DATA   0, 96,209, 18,  2,209, 18,  2,209, 18,  4,  0
2970 DATA   0,  4,181, 23,  2,162, 37,  2,162, 37,  4, 49
2980 DATA  28,  2, 49, 28,  8, 49, 28,  2, 49, 28,  4, 30
2990 DATA  25,  2,181, 23,  4, 31, 21,  2, 31, 21,  8,  0
3000 DATA   0, 10,209, 18,  2,209, 18,  4,181, 23,  2,181
3010 DATA  23,  6,  0,  0,  2,162, 37,  2,162, 37,  4, 49
3020 DATA  28,  2, 49, 28,  6,  0,  0,  2, 49, 28,  2, 49
3030 DATA  28,  2, 30, 25,  6,  0,  0,  4,181, 23,  2, 30
3040 DATA  25,  2, 31, 21,  8,  0,  0,  8,209, 18,  8,  0
3050 DATA   0,  4, 31, 21,  2, 96, 22,  4, 49, 28,  4,209
3060 DATA  18,  6,  0,  0,  2, 49, 28,  2, 30, 25,  3, 30
3070 DATA  25,  1, 30, 25,  8,  0,  0,  2, 49, 28,  2,223
3080 DATA  29,  8,  0,  0,  8,  0,  0,  2,223, 29,  2,135
3090 DATA  33,  2,162, 37,  2,  0,  0,  2, 62, 42,  2,193
3100 DATA  44,  2, 60, 50,  2, 60, 50,  4, 99, 56,  8, 60
3110 DATA  50,  2, 75, 63, 10,  0,  0,  8, 99, 56,  4, 99
3120 DATA  56,  4,  0,  0, 20, 62, 42,  4,162, 37,  2,162
3130 DATA  37,  2,162, 37,  2,162, 37,  2,162, 37,  4,162
3140 DATA  37,  4,  0,  0, 12,165, 31,  4, 49, 28,  4,165
3150 DATA  31,  4,135, 33,  8,165, 31,  4, 49, 28,  2, 49
3160 DATA  28,  6,  0,  0, 68,162, 37,  2,162, 37,  2,162
3170 DATA  37,  2,162, 37,  2,162, 37,  6,162, 37,  2,162
3180 DATA  37,  8,  0,  0,  4, 30, 25,  4, 49, 28,  6,165
3190 DATA  31,  4,135, 33,  6,165, 31,  6, 49, 28, 10,  0
3200 DATA   0,  6, 30, 25,  2,165, 31,  4, 49, 28,  4,162
3210 DATA  37,  4,135, 33,  2,165, 31,  6, 49, 28,  2,165
3220 DATA  31, 10,  0,  0, 12, 30, 25,  2, 30, 25,  2, 49
3230 DATA  28,  2,165, 31,  4,165, 31,  4,135, 33,  4, 30
3240 DATA  25,  4, 30, 25,  6,  0,  0,  8,165, 31,  6,165
3250 DATA  31,  2, 62, 42, 12,223, 39,  2,223, 39,  2,223
3260 DATA  39,  4,134, 35,  8,134, 35, 12,165, 31,  6, 49
3270 DATA  28,  2, 49, 28,  4, 49, 28,  4,  0,  0,  4,223
3280 DATA  39,  2,223, 39,  2,223, 39,  4,107, 47,  4,165
3290 DATA  31,  8,  0,  0, 10, 49, 28,  4, 49, 28,  2, 49
3300 DATA  28,  4, 49, 28,  2, 30, 25,  6, 30, 25,  4, 30
```

```
3310 DATA  25,  8, 49, 28,  8, 30, 25, 16,  0,  0,  4,223
3320 DATA  29,  4,135, 33,  4,223, 29,  4, 30, 25, 32,  0
3330 DATA   0, 64,104,  9,  6, 24, 14,  6,195, 16,  6, 49
3340 DATA  28,  6,195, 16,  4, 49, 28,  4,104,  9,  6, 24
3350 DATA  14,  6,195, 16,  6, 49, 28,  6,195, 16,  4, 49
3360 DATA  28,  4,104,  9,  6, 24, 14,  6,195, 16,  6, 49
3370 DATA  28,  6,195, 16,  4, 49, 28,  4,104,  9,  6, 24
3380 DATA  14,  6,195, 16,  6, 49, 28,  6,195, 16,  4, 49
3390 DATA  28,  4,143, 12, 32,143, 12, 32,143, 12, 32,143
3400 DATA  12, 32,104,  9,  6, 24, 14,  6,195, 16,  6, 49
3410 DATA  28,  6,195, 16,  4, 49, 28,  4,104,  9,  6, 24
3420 DATA  14,  6,195, 16,  6, 49, 28,  6,195, 16,  4, 49
3430 DATA  28,  4,104,  9,  6, 24, 14,  6,195, 16,  6, 49
3440 DATA  28,  6,195, 16,  4, 49, 28,  4,104,  9,  2,  0
3450 DATA   0,  0,  0,  0,  0,  0,  0
3460 IF S <> 191135 THEN PRINT "FEHLER IN DATAS!!!":END
3470 PRINT "OK - START MIT SYS 32768"
```

5.9 Frequenzmodulation

Nach diesem ziemlich langen Assemblerprogramm wollen wir uns noch einigen kürzeren Anwendungen von Assembler bei der Programmierung des Sound-Chips zuwenden.

In einigen Fällen wurde im vierten Kapitel auf dieses fünfte Kapitel verwiesen. Das sind die Fälle, in denen Basic bei der Programmierung des Sound-Chips versagt. Einer dieser Fälle ist die Frequenzmodulation.

Bisher haben wir es stets mit nur einer über die Tondauer hinweg konstant bleibenden Frequenz zu tun gehabt. Was passiert nun, wenn man die Tonfrequenz sehr rasch ändert?

Mit einer Geschwindigkeit, wie sie nur Assembler bietet, lassen sich eine Unzahl von Effektsounds programmieren. Wenn man die Technik der Frequenzmodulation anwendet, muß man zuerst verstehen, was "Modulation" bedeutet.

Im vierten Kapitel wurde angesprochen, was "Ring Modulation" bewirkt. Bei der Frequenzmodulation können wir mit einer ähnlichen Vorstellung arbeiten. Wir haben einen Basisoszillator, der "moduliert" wird, und einen Kontrolloszillator, der "modulierend" wirkt.

"Moduliert" werden können nun bestimmte Register des Basisoszillators, "modulieren" bedeutet hier soviel wie "laufend verändern". Das heißt, daß wir nicht den normalen Ablauf Tonfrequenz A - Tonfrequenz B - Tonfrequenz C haben, wo der zeitliche Abstand zwischen A, B und C durch die Tondauer bestimmt wird, sondern die Frequenz wird kontinuierlich verändert.

Im Beispielprogramm wirkt der dritte Oszillator als Kontrolloszillator und der erste Oszillator als Basisoszillator. Hier brauchen wir nun zwei Register, die bisher nur am Rande erwähnt wurden: das Register $D41B und das Bit 3 AUS aus Register $D418.

Das Register $D41B erlaubt es dem Mikroprozessor, den Digitalwert des Signals von Oszillator 3 zu lesen. Dieser Wert wird nun in die Frequenz von Oszillator 1 geschrieben. Damit man nur das Signal von Oszillator 1 und nicht das von Oszillator 3 hört, der ja nur als Modulationsquelle fungieren soll, wird durch das Setzen des Bits "3 AUS" der Oszillator 3 "unhörbar" gemacht.

Das Programm produziert nun sich laufend ändernde Frequenzen, die man keiner bestimmten Tonhöhe mehr zuordnen kann. Von der Frequenz von Oszillator 3 hängt es ab, in welcher Art Oszillator 1 moduliert wird.

Im Programm wird die Frequenz von Oszillator 3 auf verschiedene Arten ermittelt. Ist Counter = $A2 (erster Basic-Loader), so wird die Frequenz von Oszillator 3 aus der Interrupt-gesteuerten Uhr (TI$) ermittelt.

Bei Counter = $C5 (zweiter Basic-Loader) hängt es davon ab, welche Taste Sie drücken, was man für einen Ton hört. Probieren Sie einmal die Funktionstasten oder die RETURN-Taste aus!

Bei Counter = $DC0E (dritter Basic-Loader) wird die Frequenz aus dem Inhalt eines Registers eines I/O-Chips ermittelt. Probieren Sie einmal, die Frequenz des dritten Oszillators durch Inhalte anderer Ihnen bekannter Speicherzellen zu ermitteln!

```
1 SYS49152
2 .OPT OO
3 *=828  ;START MIT SYS 828
10 COUNTER = $A2 ; ODER $C5 ODER $DC0E
11 OSC1HF = $D401:OSC1WV = $D404
12 OSC1AD = $D405:OSC1SR = $D406
13 OSC3HF = $D40F:OSC3WV = $D412
14 OSC3AD = $D413:OSC3SR = $D414
15 VOLUME = $D418:MODFRQ = $D41B
100 GETIN = $FFE4
110 LDA #$8F:STA VOLUME
120 LDA #$21:STA OSC3WV
130 LDA #$F0:STA OSC3SR
140 LDA #$01:STA OSC1WV
200 NEXT LDA !COUNTER:ASL:STA OSC3HF
220 LDA MODFRQ:STA OSC1HF
230 LDA #$21:STA OSC1WV
240 LDA #$08:STA OSC1AD
250 LDA #$F0:STA OSC1SR
260 JSR GETIN:CMP #3:BNE NEXT
270 LDA #$00:STA VOLUME:RTS

READY.
```

```
1000 FOR I = 828 TO 888
1010 READ X:POKE I,X:S = S+X:NEXT
1020 DATA 169,143,141, 24,212,169, 33,141, 18,212,169,240
1030 DATA 141, 20,212,169,  1,141,  4,212,173,162,  0, 10
1040 DATA 141, 15,212,173, 27,212,141,  1,212,169, 33,141
1050 DATA   4,212,169,  8,141,  5,212,169,240,141,  6,212
1060 DATA  32,228,255,201,  3,208,221,169,  0,141, 24,212
1070 DATA  96
1080 IF S <> 7702 THEN PRINT "FEHLER IN DATAS!!!":END
1090 PRINT "OK - START MIT SYS 828"
```

```
1000 FOR I = 828 TO 888
1010 READ X:POKE I,X:S = S+X:NEXT
1020 DATA 169,143,141, 24,212,169, 33,141, 18,212,169,240
1030 DATA 141, 20,212,169,  1,141,  4,212,173,197,  0, 10
1040 DATA 141, 15,212,173, 27,212,141,  1,212,169, 33,141
1050 DATA   4,212,169,  8,141,  5,212,169,240,141,  6,212
1060 DATA  32,228,255,201,  3,208,221,169,  0,141, 24,212
1070 DATA  96
1080 IF S <> 7737 THEN PRINT "FEHLER IN DATAS!!!":END
1090 PRINT "OK - START MIT SYS 828"
```

```
1000 FOR I = 828 TO 888
1010 READ X:POKE I,X:S = S+X:NEXT
1020 DATA 169,143,141, 24,212,169, 33,141, 18,212,169,240
1030 DATA 141, 20,212,169,  1,141,  4,212,173, 14,220, 10
1040 DATA 141, 15,212,173, 27,212,141,  1,212,169, 33,141
1050 DATA   4,212,169,  8,141,  5,212,169,240,141,  6,212
1060 DATA  32,228,255,201,  3,208,221,169,  0,141, 24,212
1070 DATA  96
1080 IF S <> 7774 THEN PRINT "FEHLER IN DATAS!!!":END
1090 PRINT "OK - START MIT SYS 828"
```

5.10 Interrupts in der Musikprogrammierung

Zum Abschluß des fünften Kapitels wollen wir auf eine besondere Art der Musikprogrammierung eingehen, die eigentlich gar keine ist.

Die beiden Programme, die auf den folgenden Seiten abgedruckt sind, sind sogenannte "Interrupts". Wenn Sie das Maschinenprogramm durch den entsprechenden SYS starten, meldet sich Basic sofort zurück mit "READY.". Wenn Sie nun beliebige Tasten drücken, so hören Sie einen Ton, der von Taste zu Taste unterschiedlich ist.

Dieser "Interrupt", der das bewirkt, ist solange aktiv, bis der "Interrupt-Vektor" wieder verändert wird. Dies geschieht zum Beispiel nach Drücken von RUN/STOP und RESTORE.

Auf welche Art und Weise aus der gedrückten Taste die Tonfrequenz errechnet wird, ist bei beiden Programmen unterschiedlich. Beim ersten Programm wird der Inhalt der Speicherzelle 197 oder wahlweise auch 162 direkt als High-Wert der Tonfrequenz verwendet.

Bei 197 wird ein Index der Tastaturabfrage verwendet, bei 162 das Low Byte der Interrupt-gesteuerten Uhr (TI$). Das Ergebnis sind ziemlich schräge Klänge, die aber für den Unwissenden ziemlich effektvoll sein können. Denn während der Interrupt aktiv ist, kann man in Basic editieren, Diskketteninhaltsverzeichnisse lesen, Programme ausdrucken usw.

Das zweite Programm verwendet die Tastatur des Commodore 64 als Klaviertastatur. Sie werden leicht herausfinden, welchen Tasten welche Töne zugeordnet sind. Die beiden oberen und die beiden unteren Tastenreihen bewältigen zusammen ca. 3 1/2 Oktaven.

Wer den Profi-Ass-Quelltext eingibt, kann die Tastaturbelegung nach Belieben verändern. Es wäre auch eine gute Idee, um diesen Interrupt herum ein kleines Basic-Programm zu schreiben, das es dem Benutzer erlaubt, z.B. Wellenform oder Hüllkurve zu ändern.

Wenn man besonders viel Ausdauer hat, kann man diesen kurzen Interrupt zu einem komplexen Musikprogramm hochzüchten, das Sounds speichert, Melodien speichert usw. Man könnte auch über eine GET-Schleife die gedrückten Tasten erkennen und auf dem Bildschirm eine grafisch simulierte Klaviertastatur anzeigen. Dem Einfallsreichtum sind hier keine Grenzen gesetzt.

Wieso ist das denn nun eigentlich keine Musikprogrammierung mehr? Ganz einfach, der Benutzer muß nicht mehr Note für Note mühselig eingeben, sondern kann direkt munter drauflosspielen. Wie das ganze auch mehrstimmig geht, können Sie im Anhang 1 nachlesen.

```
1 SYS49152
2 .OPT OO
3 *=828
10 TONE   = 197  ;ODER 162
11 OSC1HF = $D40i
12 OSC1PL = $D402
13 OSC1PH = $D403
14 OSC1WV = $D404
15 OSC1AD = $D405
16 OSC1SR = $D406
17 VOLUME = $D418
100 SEI:LDA #<IRQ:STA $0314:LDA #>IRQ:STA $315:CLI:RTS
110 IRQ LDA 197:CMP #64:BEQ GATEOFF
130 LDA #$0F:STA VOLUME
140 LDA 197:AND #$F:STA OSC1PH
150 LDA #$80:STA OSC1PL
160 LDA TONE:STA OSC1HF
170 LDA #$41:STA OSC1WV
180 LDA #$18:STA OSC1AD
190 LDA #$58:STA OSC1SR
200 EXIT JMP $EA31
210 GATEOFF LDA #$40:STA OSC1WV:BNE EXIT

READY.
```

```
1000 FOR I = 828 TO 893
1010 READ X:POKE I,X:S = S+X:NEXT
1020 DATA 120,169, 73,141, 20,  3,169,  3,141, 21,  3, 88
1030 DATA  96,165,197,201, 64,240, 40,169, 15,141, 24,212
1040 DATA 165,197, 41, 15,141,  3,212,169,128,141,  2,212
1050 DATA 165,197,141,  1,212,169, 65,141,  4,212,169, 24
1060 DATA 141,  5,212,169, 88,141,  6,212, 76, 49,234,169
1070 DATA  64,141,  4,212,208,246
1080 IF S <> 7818 THEN PRINT "FEHLER IN DATAS!!!":END
1090 PRINT "OK - START MIT SYS 828"
```

```
1000 FOR I = 828 TO 893
1010 READ X:POKE I,X:S = S+X:NEXT
1020 DATA 120,169, 73,141, 20,  3,169,  3,141, 21,  3, 88
1030 DATA  96,165,197,201, 64,240, 40,169, 15,141, 24,212
1040 DATA 165,197, 41, 15,141,  3,212,169,128,141,  2,212
1050 DATA 165,162,141,  1,212,169, 65,141,  4,212,169, 24
1060 DATA 141,  5,212,169, 88,141,  6,212, 76, 49,234,169
1070 DATA  64,141,  4,212,208,246
1080 IF S <> 7783 THEN PRINT "FEHLER IN DATAS!!!":END
1090 PRINT "OK - START MIT SYS 828"
```

```
1 POKE 56,128:CLR:SYS49152
2 .OPT OO
3 *=$8000
10 LSTKEY = 197
11 OSC1LF = $D400
12 OSC1HF = $D401
13 OSC1WV = $D404
14 OSC1AD = $D405
15 OSC1SR = $D406
20 VOLUME = $D418
100 SEI:LDA #<IRQ:STA $0314:LDA #>IRQ:STA $315:CLI:RTS
110 IRQ LDA LSTKEY:CMP #64:BEQ GATEOFF
120 ASL:TAX
130 LDA TAB,X:STA OSC1LF
140 INX
150 LDA TAB,X:STA OSC1HF
160 LDA #$21:STA OSC1WV
170 LDA #$18:STA OSC1AD
180 LDA #$58:STA OSC1SR
190 LDA #$0F:STA VOLUME
200 EXIT JMP $EA31
210 GATEOFF LDA #$20:STA OSC1WV:BNE EXIT
999 TAB *=*
1000 DE .WORD $0FD2 ;DELETE
1001 RE .WORD $0218
1002 CR .WORD $3863
1003 F7 .WORD $2187
1004 F1 .WORD $0430
1005 F3 .WORD $0861
1006 F5 .WORD $10C3
1007 CD .WORD $323C
1008 K3 .WORD $04FB
1009 KW .WORD $04B4
1010 KA .WORD $0FD2
1011 K4 .WORD $0000 ;NICHT VERW.
1012 KZ .WORD $10C3
1013 KS .WORD $11C3
```

```
1014 KE .WORD $0547
1015 .WORD $0000
1016 K5 .WORD $05ED
1017 KR .WORD $0598
1018 KD .WORD $13EF
1019 K6 .WORD $06A7
1020 KC .WORD $151F
1021 KF .WORD $0000 ;NICHT VERW.
1022 KT .WORD $0647
1023 KX .WORD $12D1
1024 K7 .WORD $0777
1025 KY .WORD $070C
1026 KG .WORD $17B5
1027 K8 .WORD $0000 ;NICHT VERW.
1028 KB .WORD $191E
1029 KH .WORD $1A9C
1030 KU .WORD $07E9
1031 KV .WORD $1660
1032 K9 .WORD $08E1
1033 KI .WORD $0861
1034 KJ .WORD $1DDF
1035 KO .WORD $09F7
1036 KM .WORD $1FA5
1037 KK .WORD $0000 ;NICHT VERW.
1038 KO .WORD $0968
1039 KN .WORD $1C31
1041 K+ .WORD $0000 ;NICHT VERW.
1042 KP .WORD $0A8F
1043 KL .WORD $2386
1044 DA .WORD $0BDA ;MINUS
1045 PO .WORD $25A2 ;PUNKT
1046 CO .WORD $27DF ;DOPPELPUNKT
1047 AT .WORD $0B30 ;SCHNECKE
1048 CM .WORD $2187 ;KOMMA
1049 PF .WORD $0D4E ;PFUND
1050 MU .WORD $0C8F ;MAL
1051 SC .WORD $2CC1 ;SEMIKOLON
1052 CL .WORD $0EEF ;CLEAR--HOME
1053 .WORD $0000
```

```
1054 EQ .WORD $2F6B ;GLEICH
1055 UA .WORD $0E18 ;PFEIL OBEN
1056 DB .WORD $2A3E ;SCHRAEGSTRICH
1057 EX .WORD $03F4 ;AUSRUFEZEICHEN
1058 LA .WORD $03BB ;PFEIL LINKS
1059 .WORD $0000
1060 QU .WORD $0470 ;QUOTE
1061 .WORD $0000
1062 .WORD $0000
1063 KQ .WORD $0430
1064 RS .WORD $0861 ;RUN-STOP

READY.
```

```
 1000 POKE 56,128:CLR:FOR I = 32768 TO 32959
 1010 READ X:POKE I,X:S = S+X:NEXT
 1020 DATA 120,169, 13,141, 20,  3,169,128,141, 21,  3, 88
 1030 DATA  96,165,197,201, 64,240, 38, 10,170,189, 64,128
 1040 DATA 141,  0,212,232,189, 64,128,141,  1,212,169, 33
 1050 DATA 141,  4,212,169, 24,141,  5,212,169, 88,141,  6
 1060 DATA 212,169, 15,141, 24,212, 76, 49,234,169, 32,141
 1070 DATA   4,212,208,246,210, 15, 24,  2, 99, 56,135, 33
 1080 DATA  48,  4, 97,  8,195, 16, 60, 50,251,  4,180,  4
 1090 DATA 210, 15,  0,  0,195, 16,195, 17, 71,  5,  0,  0
 1100 DATA 237,  5,152,  5,239, 19,167,  6, 31, 21,  0,  0
 1110 DATA  71,  6,209, 18,119,  7, 12,  7,181, 23,  0,  0
 1120 DATA  30, 25,156, 26,233,  7, 96, 22,225,  8, 97,  8
 1130 DATA 223, 29,247,  9,165, 31,  0,  0,104,  9, 49, 28
 1140 DATA   0,  0,143, 10,134, 35,218, 11,162, 37,223, 39
 1150 DATA  48, 11,135, 33, 78, 13,143, 12,193, 44,239, 14
 1160 DATA   0,  0,107, 47, 24, 14, 62, 42,244,  3,187,  3
 1170 DATA   0,  0,112,  4,  0,  0,  0,  0, 48,  4, 97,  8
 1180 IF S <> 16114 THEN PRINT "FEHLER IN DATAS!!!":END
 1190 PRINT "OK - START MIT SYS 32768"
```

Anhang 1: Kurzbeschreibung Synthimat 64

Wenn Sie das letzte Programm des fünften Kapitels eingegeben haben, so werden Sie bemerkt haben, daß es viel einfacher ist, direkt über die Tastatur zu spielen als mühselig Note für Note einzugeben.

Es gibt inzwischen schon unzählige Musikprogramme, die diesen Vorteil erkannt haben. Doch keines konnte bisher die Möglichkeit bieten, mehrstimmig zu spielen. Bei keinem konnte man direkt über die Tastatur Akkorde spielen oder Baß und Melodie gleichzeitig.

Bei den meisten Musikprogrammen für den Commodore 64 bleibt der Computer nichts als ein Sequenzer. Bei einem Sequenzer hat man wie bei den Musikprogrammen und Musiktools nur die Möglichkeit, Note für Note, Tondauer für Tondauer einzugeben. Das ist ein langwieriger Prozeß, bei dem man schnell die Lust verliert. Und wenn das Musikstück dann einmal fertig ist, dann hört man es sich zwei- bis dreimal an, und dann verliert es seinen Reiz.

Mit allen diesen Nachteilen räumt Synthimat 64 mit einem Schlag auf. Mit Synthimat 64 können Sie den Commodore 64 endlich wie einen richtigen polyphonen Synthesizer einsetzen. Sie brauchen keine einzige Tondauer oder Tonhöhe mehr zu programmieren, Sie können direkt über die Tastatur dreistimmig spielen. Sie können durch ausschließliche Benutzung der Funktionstasten alle Parameter einstellen, die der Sound-Chip hat, und noch einige mehr. Zudem können Sie alles, was Sie auf der Tastatur spielen und alle Sounds, die Sie einmal eingegeben haben, direkt auf Diskette abspeichern und wieder einladen. Es ist also möglich, sich seine eigene Sound-Bibliothek auf Diskette zu schaffen, ohne umständlich Notizen machen zu müssen.

Synthimat 64 wurde mit Profi-Ass geschrieben und ist nur 8 K Byte lang. Sie brauchen deshalb nicht lange warten, bis das Programm eingeladen ist. Nach weniger als einer halben Minute erscheint auf dem Bildschirm Ihres Commodore 64 das Bedienungspult von Synthimat 64. Alle Elemente des Sound-Chips sind logisch auf dem Bidschirm angeordnet und farblich gekennzeichnet. Jeder Oszillator hat eine eigene Kennfarbe. Alle Funktionen, die sich auf einen bestimmten Oszillator beziehen, werden in der entsprechende Kennfarbe angezeigt. Mit einem Cursor können Sie nach Belieben auf dem Bildschirm hin- und herwandern und alle Parameter einstellen.

Synthimat 64 nutzt die Tastatur des Commodore 64 voll aus. Den Hauptanteil belegen die zwei Manuale von Synthimat 64. Sie haben also zwei Tastaturen, die Sie unabhängig voneinander bespielen können. Mit den Funktionstasten können Sie einstellen, welche der drei Oszillatoren welchen der beiden Tastaturen zugeordnet werden sollen. Hier sind alle Kombinationen frei anwählbar.

Die beiden Tastaturen von Synthimat 64 sind zu jeder Zeit auf dem Bildschirm sichtbar. Sie können also zu jeder Zeit auch optisch verfolgen, welche Tasten Sie gerade spielen. Zudem können Sie durch Drücken einer der SHIFT-Tasten jederzeit die Tastaturbelegung einblenden lassen, falls Sie einmal nicht wissen, welche Taste welchem Ton zugeordnet ist.

Die Tonhöhe von Synthimat 64 ist in 1/8 Halbtönen einstellbar. Sie können also Synthimat 64 so stimmen, daß die Stimmung genau zu einem Lied von Ihrer Stereoanlage oder zu einem anderen Musikinstrument paßt. Dies geschieht nicht umständlich über Zahlenwerte, sondern mit den Cursorsteuerungstasten. Anhand eines Balkendiagramms können Sie die Stimmung von Synthesound 64 auch optisch registrieren.

Nicht nur die Gesamtstimmung, sondern auch die Stimmung der Oszillatoren untereinander ist in 1/8 Halbtönen regelbar. Somit können auch Schwebungen erzeugt werden. Es ist möglich, alle drei Oszillatoren auf einen Ton zu legen. Mit einer Schwebung erhält man dann einen sehr schönen breiten Klang.

Da der Sound-Chip nur über drei Stimmen verfügt, ist es von vornherein nicht möglich, vier- oder fünfstimmige Akkorde zu spielen. In Synthimat 64 kann man deshalb eine der beiden Tastaturen auf eine Akkordautomatik umschalten, womit man durch einen Arpeggio-Effekt auch mehrstimmige Akkorde spielen kann.

Mit Synthimat 64 können Sie die Lieder, die Sie spielen, direkt auf Diskette aufnehmen. Das funktioniert so: Sie legen eine Diskette ein und drücken die Funktionstasten für "Aufnahme". Nun können Sie mehrere Minuten hintereinander Lieder aufnehmen. Wenn Sie die Aufnahme beenden wollen, drücken Sie einfach die "Space"-Taste. Nun ist das Lied auf Diskette gespeichert. Wollen Sie sich ein Lied wieder anhören, so betätigen Sie die Funktionstasten für "Wiedergabe". Nun wird die Tastatur des Commodore 64 abgeschaltet, und das Diskettenlaufwerk übernimmt die Funktion der Tastatur. Das Lied wird jetzt exakt so gespielt, wie Sie es aufgenommen haben.

Synthimat 64 bietet Ihnen die Möglichkeit, 9 verschiedene Lieder auf eine Diskette aufzunehmen. Die Diskette kann auch andere Programme enthalten und muß nicht für Synthimat 64 speziell formatiert werden.

Synthimat 64 verwendet die Echtzeituhr eines I/O-Chips, die netzsynchron läuft. Diese Uhr wird zu jeder Zeit mitten auf dem Bildschirm angezeigt, egal, ob Sie spielen, auf Diskette aufnehmen oder Sounds programmieren. Diese Uhr können Sie mit den Funktionstasten stellen.

Synthimat 64 verwendet auch eine interne Uhr, die "SYS-Uhr", die Sie ebenfalls ständig auf dem Bildschirm verfolgen können. Sie zeigt Ihnen die interne Geschwindigkeit von Synthimat 64 an.

Nun zum wichtigsten Punkt, den Klangmöglichkeiten. Mit Synthimat 64 haben Sie Zugriff auf folgende Parameter:

Parameter	*Werte*
VCO 1 Wellenform	*8 Wellenformen*
Fußlage	*8 Oktaven*
Pulsbreite	*0-4095*
LFO 1 Wellenform	*8 Wellenformen*
Geschwindigkeit	*0-15*
LFO 2 Wellenform	*8 Wellenformen*
Geschwindigkeit	*0-15*
Attack-Dauer	*0-15*
Decay-Dauer	*0-15*
Sustain-Wert	*0-15*
Release-Dauer	*0-15*

VCO 2 entspricht VCO 1 mit LFO 3 und 4

VCO 3 entspricht VCO 1 mit LFO 5 und 6

VCF Einstellungen	*8 Filtertypen*
Frequenz	*0-2047*
Q-Wert	*0-15*
VCO-Wahl	*16 Werte*
LFO 7 Wellenform	*8 Wellenformen*
Geschwindigkeit	*0-15*
VCA Lautstärke	*0-15*
LFO 8 Wellenform	*8 Wellenformen*
Geschwindigkeit	*0-15*
SYNChronisation	*8 Arten*
RING Modulation	*8 Arten*

Mit Synthimat 64 können Sie alle Möglichkeiten ausschöpfen, die der Sound-Chip Ihnen bietet, und noch einige mehr. Sie finden in der obigen Tabelle acht (!) LFO's, wovon Nr. 1, 3, 5 die Frequenzen modulieren, Nr. 2, 4 und 6 die Pulsbreite (je einer pro Oszillator), Nr. 7 die Filterfrequenz und Nr. 8 die Lautstärke.

Alle Einstellungen, die Sie hier haben, werden durch Synthimat 64 auf dem Bildschirm optisch dargestellt. Sie können also auf einen Blick die Einstellung auch optisch wahrnehmen.

Durch Cursorsteuerung können Sie jeden Parameter in allen seinen Werten verändern und gleichzeitig dazu auf den Tastaturen spielen. Sie können also sofort das Klangergebnis Ihrer Einstellung hören.

Jederzeit können Sie eine beliebige Einstellung als Sound-Register im Commodore 64 speichern. Es ist möglich, 256 (!) Register gleichzeitig im Speicher des Commodore 64 zu haben. Innerhalb weniger Sekunden können Sie jedes gewünschte Sound-Register anwählen. Es ist auch möglich, die Register auf Diskette zu speichern und wieder einzuladen.

Das Programm "Synthimat 64" wurde vom Autor dieses Buchs geschrieben und wird von der Firma DATA BECKER vertrieben. Sie erhalten das Programm auf einer Diskette, die neben dem Programm noch viele Beispiele enthält, zusammen mit einem ausführlichen Handbuch, das Ihnen alle Funktionen des Programms und des Sound-Chips genau erklärt, für nur DM 99,-.

Erschließen Sie sich dieses Wunderland synthetischer Musik durch die Möglichkeiten von Synthimat 64!

Anhang 2: Ein Ausblick auf weitere Anwendungen und Hardware

In diesem Anhang sollen weitere musikalische Anwendungen des Commodore 64 zur Sprache kommen und diverse hardwaremäßige Manipulationen beschrieben werden.

Der Anschluß des Commodore 64 an Ihre Stereoanlage

Der Commodore 64 erzeugt ein Audio-Signal, genauso wie es der Cassettenrecorder Ihrer Stereoanlage tut. Dieses Audiosignal können Sie über ein Überspielkabel direkt an Ihre Stereoanlage weitergeben.

Wo kann man nun das Audio-Signal des Sound-Chips herbekommen? Man braucht nicht unbedingt den Audio Out-Pin des Sound-Chips "anzuzapfen", das Signal liegt direkt an Pin 3 der Video-/Audio-Buchse an der Rückseite des Commodore 64 an. Diese Buchse liegt genau zwischen der Buchse für die Floppy (serieller Bus) und der Buchse für das Antennenkabel.

Ihr Überspielkabel (Signal + Masse) muß an einem Ende einen DIN-Stecker haben, der in die Video-/Audio-Buchse des Commodore 64 paßt. Es gibt drei- und fünfpolige DIN-Stecker, es reicht hier auch ein dreipoliger. Die Masse des Kabels löten Sie an den mittleren Pin an (das ist Pin 2 der Video-/Audio-Buchse). Die Pins einer DIN-Buchse sind halbmondförmig angeordnet. Wenn Sie von hinten auf die Rückseite des Commodore 64 sehen und die Video-/Audio-Buchse betrachten, so sehen Sie, daß die Pins einen nach oben hin offenen Halbkreis bilden. Der Pin am linken Rand dieses Halbkreises trägt das Audio Out-Signal des Sound-Chips.

Stecken Sie Ihren DIN-Stecker probeweise an die Video-/Audio-Buchse Ihres Commodore 64. Sie sehen dann, an welchen Pin des Steckers Sie die Signalleitung des Kabels anlöten müssen.

Mit dem Stecker für Ihre Stereoanlage ist es da schon komplizierter, da es hier die unterschiedlichsten Steckernormen gibt. Die beiden häufigsten sollen hier beschrieben werden:

1. Die Cinch-Buchsen, die bei den meisten japanischen und amerikanischen Geräten vorherrschen, gibt es für jeden Kanal einmal. Wenn Sie einen Cinch-Stecker an das andere Ende Ihres Überspielkabels anlöten und in eine der beiden Cinch-Buchsen Ihrer Anlage stecken, müssen Sie damit rechnen, daß man den Ton des Sound-Chips nachher nur über eine Box hört. Um zu erreichen, daß der Ton auf beiden Boxen hörbar ist, gibt es zwei Möglichkeiten:

a) Falls Ihre Anlage einen Stereo-/Mono-Umschalter besitzt, drücken Sie einfach auf Mono. Der Ton des Sound-Chips kommt dann auf beiden Boxen Ihrer Stereoanlage.

b) Sie löten an das andere Ende des Überspielkabels zwei Cinch-Stecker an, für jeden Kanal einen.

2. Die DIN-Buchsen bei den meisten deutschen Stereoanlagen haben für jeden Kanal einen Pin. Besorgen Sie sich einen fünfpoligen DIN-Stecker, löten Sie die Leitung mit dem Signal an einen der entsprechenden Pins an und verbinden Sie die beiden nebeneinanderliegenden Pins einfach durch eine Lötbrücke. So wird das Tonsignal an beide Boxen weitergegeben.

An welche Buchsen Ihrer Stereoanlage schließen Sie nun Ihr fertiges Kabel an? Viele Stereoanlagen haben einen sogenannten AUX-Eingang (AUX steht für auxiliary input = Hilfseingang). Falls Ihre Stereoanlage einen solchen Eingang besitzt, sollten Sie Ihr Kabel daran anschließen. In allen anderen Fällen sollten Sie den Eingang für den Cassettenrecorder benutzen.

Der Audio In-Pin des Sound-Chips liegt an der Video-/Audio-Buchse Ihres Commodore 64 direkt neben dem Audio Out-Pin. Wenn Sie also den Halbkreis, den die fünf Pins dieser Buchse bilden und der nach oben hin offen ist, betrachten, dann ist dies der zweite Pin von links.

Dieser Pin ist direkt mit dem Audio In-Pin des Sound-Chips verbunden. Es ist hier möglich, ein externes Tonsignal in den Sound-Chip einzuspeisen.

Was für ein Tonsignal bietet sich hier an? Wenn Sie an diesen Eingang ein Mikrofon anschließen, so wird man kaum einen Ton hören. Das liegt daran, daß das Tonsignal vorverstärkt werden muß.

Die Vorverstärkung eines Signals erfolgt am besten über Ihre Stereoanlage, die ja einen Verstärker enthält. Nehmen wir als Beispiel eines externen Tonsignals den Ton eines Radiosenders. Um das verstärkte Signal der Stereoanlage zu entnehmen, müssen Sie einen Ausgang wählen, dessen Lautstärke vom Lautstärkeregler der Stereoanlage eingestellt werden kann, und dies ist (neben den Lautsprecherausgängen) die Buchse für einen Kopfhörer. Die Kopfhörerbuchse ist hier am geeignesten, da man nicht immer die Lautsprecher abklemmen muß, wenn man das Tonsignal zum Commodore 64 leiten will.

Das Problem ist also zunächst, die Verbindung zwischen Kopfhörerbuchse an der Stereoanlage und Audio In-Pin der Video-/Audio-Buchse des Commodore 64 herzustellen.

Nun gibt es bei den Kopfhörerbuchsen der verschiedenen Stereoanlagen die verschiedensten Steckernormen: Buchsen für kleine Klinkenstecker (2,5 und 3,5 mm), große Klinkenstecker und fünfpolige Würfelstecker. Gehen Sie mit Ihrem Kopfhörer am besten in ein Radio-Fachgeschäft und verlangen Sie einen Stecker, wie er am Kopfhörer für Ihre Stereoanlage ist.

Wenn Sie einen entsprechenden Stecker haben, öffnen Sie den Stecker Ihres Kopfhörers und schauen sich an, an welchen Pins Kabel angelötet sind. Wenn der Stecker Ihres Kopfhörers sich nicht öffnen läßt, was zuweilen vorkommt, dann muß man den entsprechenden Pin eben durch Ausprobieren finden.

Das nächste Problem ist der Stecker für die Video-/Audio-Buchse des Commodore 64. Hier braucht man einen fünfpoligen DIN-Stecker. Zum Verbinden beider Stecker reicht ein einfaches Überspielkabel (Signal + Masse). Löten Sie nun das eine Kabelende an den Stecker für den Commodore 64.

Beim anderen Kabelende ist die Sache etwas schwieriger, da die Pinbelegung von Gerät zu Gerät anders ist. Stereokopfhörer haben zwei Signal-Pins, für jeden Kanal einen. Diese beiden Pins sollten Sie durch eine Lötbrücke verbinden, damit das Tonsignal mono zum Commodore 64 weitergeht. An einen der beiden Pins lötet man die Signalleitung an.

Wenn Ihre Stereoanlage beim Einstecken eines Kopfhörers die beiden Lautsprecher automatisch abschaltet, so sollten Sie hier das Kabel erst einstecken, wenn Sie ein entsprechendes Tonsignal gefunden haben. Suchen Sie also einen Radiosender, stellen auf eine geringe Lautstärke ein und schließen Sie dann erst Ihr Kabel an die Stereoanlage an.

Es ist nicht möglich, den Ton des Commodore 64 über eine Stereoanlage zu leiten und gleichzeitig das Signal der Stereoanlage, etwa einen Radiosender, in den Commodore 64 einzuleiten. Dazu bräuchte man schon zwei Stereoanlagen. Entfernen Sie daher gegebenenfalls ein Kabel, das vom Commodore 64 zur Stereoanlage geht, von der Video-/Audio-Buchse, bevor Sie Ihr neues Kabel einstecken. Stecken Sie dann das neue Kabel an diese Buchse. Da hiermit auch das Video-Signal nicht mehr verfügbar ist, müssen Sie, um das Bild und den Ton des Commodore 64 zum Fernseher zu leiten, den HF-Modulator benutzen. Verbinden Sie also Fernseher und Commodore 64 durch das normale Kabel und nicht durch ein Video-Kabel.

Gehen wir nun davon aus, daß die Verbindung geglückt ist. Wenn Sie den Commodore 64 einschalten, wird man den Radio-Sender nicht hören. Hüten Sie sich davor, den Lautstärkeregler an der Stereoanlage allzuweit aufzudrehen! Grund hierfür ist nicht eine zu geringe Ausgangsleistung der Stereoanlage. Die Lautstärke des externen Tonsignals wird wie die Lautstärke der internen Oszillatoren über die vier VOL-Kontrollbits (siehe Kapitel 4) gesteuert. Um diese z.B. auf Maximallautstärke zu setzen, geben Sie ein: POKE 54296,15. Nun müßte das Tonsignal, wenn auch verzerrt, über den Lautsprecher des Fernsehers zu hören sein.

Es ist möglich, durch Anlöten eines weiteren Kabels an den Stecker für die Video-/Audio-Buchse des Commodore 64 das Audio Out-Signal zu einer weiteren Stereoanlage oder auch zu einem kleinen Radio zu leiten, wenn Sie mit der Tonqualität des HF-modulierten Signals nicht zufrieden sind.

Sie können neben einem Radio-Signal natürlich auch andere Tonsignale in Ihren Commodore 64 schicken, soweit sie Ihre Stereoanlage verarbeitet. Es ist möglich, Cassetten über den Cassettenrecorder der Stereoanlage abzuspielen und an den Commodore 64 weiterzuleiten. Auch Schallplatten kann man so zum Commodore 64 leiten. Wenn Ihr Cassettenrecorder Mikrofoneingänge hat, so können Sie auch ein Mikrofon anschließen, den Cassettenrecorder auf Vorderbandkontrolle ("Source", "Monitor" u.ä.) einstellen und so über den Commodore 64 Ihre Stimme zum Fernsehlautsprecher leiten.

Nun kann man sich fragen: Wozu das alles, dafür brauche ich doch nicht den Commodore 64. Man kann das alles doch auch mit der Stereoanlage allein machen.

Aber es gibt ja noch das Bit FILTEX (Bit 3 von Register 23, siehe Kapitel 4), mit dem man das externe Tonsignal durch den Filter leiten kann. Und hier geht dann der Spaß erst richtig los.

Der Befehl zum Setzen des FILTEX-Bits lautet: POKE 54295,8. Der Befehl zum Löschen des FILTEX-Bits lautet: POKE 54295,0. Falls das Bit gelöscht ist, wird das externe Tonsignal ausschließlich durch die Gesamtlautstärke in seiner Lautstärke beeinflußt, ansonsten bleibt es unverändert. Wenn das Bit gesetzt ist, sind alle Möglichkeiten des Filters anwendbar, die im vierten Kapitel besprochen wurden.

Im folgenden Programm wollen wir uns einmal die besonderen Anwendungsmöglichkeiten des Filters für das externe Tonsignal anschauen:

```
0 PRINT CHR$(147);:INPUT "FILTERMODUS (1-7)";F
1 POKE 54296,F*16+15:POKE 54295,8:POKE 650,128
2 PRINT CHR$(19):PRINT:PRINT"FILTERFREQUENZ:"I;CHR$(157)" "
3 POKE 54294,I:GET A$
4 IF A$="+" THEN IF I<255 THEN I=I+1
5 IF A$="-" THEN IF I>0   THEN I=I-1
6 IF A$=" " THEN END
7 IF A$=CHR$(13) THEN RUN
8 GOTO 2
```

Nachdem Sie das Programm eingegeben haben und mit RUN gestartet haben, müssen Sie den Filtermodus einstellen (siehe die Tabelle der Filtertypen auf der nächsten Seite). Danach können Sie die Filtergrenzfrequenz mit den Tasten "+" und "-" verstellen. Achten Sie darauf, daß die Werte 0 nicht unter- und 255 nicht überschreiten können. Mit RETURN können Sie das Programm neu starten, um einen anderen Filtermodus auszuprobieren. Mit Space beenden Sie das Programm.

Um das Programm anzuwenden, müssen Sie natürlich ein externes Tonsignal angeschlossen haben. Da das Programm die drei internen Oszillatoren nicht anspricht, kann es nötig sein, vor Programmstart RUN/STOP RESTORE zu drücken. Durch Drücken dieser beiden Tasten können Sie das externe Tonsignal auch wieder abschalten.

Im folgenden Programm wird der Vorgang Filterfrequenz erhöhen - Filterfrequenz erniedrigen kontinuierlich durchgeführt. Der Wert aus dem vorigen Programm, der von 0 bis 255 durch Tastendruck eingestellt werden konnte, wird also vom Programm laufend erhöht und erniedrigt.

```
1 POKE 54296,31:POKE 54295,8
2 FOR I=0 TO 255:POKE 54294,I:NEXT
3 FOR I=255 TO 0 STEP -1:POKE 54294,I:NEXT
4 IF PEEK (198)=0 THEN RUN
5 POKE 54295,0:POKE 198,0
```

Das Programm läuft solange, bis Sie eine Taste drücken. Wenn dann ein Durchlauf beendet ist, stoppt das Programm und löscht zuvor noch das FILTEX-Bit.

In diesem zweiten Programm zur Filterung des externen Tonsignals wurde der Filtermodus Tiefpass-Filter verwendet. Es ist hier, wie auch im ersten Programm, möglich, alle sieben Filtermodus-Kombinationen einzustellen. In folgender Tabelle finden Sie alle Kombinationen und die entsprechenden Werte für das erste und das zweite Programm:

Wert Programm 1	Wert Programm 2	resultierender Filtertyp
1	31	Tiefpass
2	47	Bandpass
3	63	Tief- und Bandpass
4	79	Hochpass
5	95	Tief- und Hochpass
6	111	Band- und Hochpass
7	127	All Pass

Diese Werte finden Sie im vierten Kapitel im Abschnitt zum Filter des Sound-Chips wieder. Es gibt jedoch noch viele andere Wege, die Filtergrenzfrequenz zu verändern, als nur durch bloßes Herauf- und Herunterzählen. Im folgenden Einzeiler wird die Filterfrequenz aus dem Inhalt eines Timers von einem I/O-Baustein ermittelt.

```
0 POKE 54296,15:POKE 54295,8:POKE 54294,PEEK(56324):RUN
```

Ergebnis: eine vollkommene "Zerstückelung" des Tonsignals. Weitere Möglichkeiten zur Beeinflussung des externen Tonsignals haben Sie durch das Programm "Synthimat 64" (siehe Anhang 1).

Die Filterung des externen Tonsignals durch Synthimat 64

In Synthimat 64 gibt es zur Veränderung der Filterfrequenz einen eigenen LFO (Low Frequency Oszillator; Erklärung siehe Anhang 3). Sie können hier das Tonsignal genauso filtern, wie es in den vorigen Programmen getan wurde. Es gibt jedoch durch den LFO Nr. 7 noch viele weitere Möglichkeiten: die Filterfrequenz langsam oder schnell verändern, kontinuierlich oder "abgehackt", über große oder kleine Frequenzbereiche.

Alle "normalen" Filtereinstellungen und LFO-Einstellungen lassen sich als Klangregister abspeichern. In Sekundenschnelle kann man dann die verschiedenen Variationsmöglichkeiten abrufen. Man kann sie natürlich auch auf Diskette speichern und später wieder einladen, so daß man die Werte nicht immer neu eingeben muß.

Der Anschluß einer externen Tastatur

Es ist möglich, an den Commodore 64 ein externes Keyboard anzuschließen. Durch geeignete Software läßt sich dann der Synthesizer in Ihrem Commodore 64 auf dieser Tastatur spielen.

Eine solche Tastatur, das sogenannte "Home Computer-Keyboard", bietet inzwischen die Firma Wersi an. Der Anschluß dieser Tastatur an den Commodore 64 erfolgt über einen Zwischenstecker zum normalen Cannon-Tastaturstecker des Commodore 64.

Da diese spezielle Tastatur eine 8*8-Matrixabfrage ermöglicht, können die gedrückten Tasten über Character-Strings in einem Programm abgefragt werden.

Leider gibt es einen solchen Zwischenstecker speziell für den Commodore 64 noch nicht im Handel, so daß man also die erforderliche Hardware sich selbst bauen muß. Auch wenn der Anschluß der Tastatur an den Commodore 64 geglückt ist, muß erst noch entsprechende Software entwickelt werden.

Es ist allerdings zu erwarten, daß sich bestimmte Bastlerkreise in Kürze auf dieses Problem stürzen und es auch lösen werden.

Soweit die Fakten. Über andere Möglichkeiten, ein Keyboard an den Commodore 64 anzuschließen, gibt es nur mehr Gerüchte. Es soll eine weitere Tastatur mit nur drei Oktaven (die Wersi-Tastatur hat fünf Oktaven mit oberstem C, also 61 Tasten) existieren, die an den User-Port angeschlossen wird. Entsprechende Software hierfür zu entwickeln, dürfte aufgrund der recht verzwickten User-Port-Abfrage wesentlich schwieriger sein.

Die "Verkettung" von Sound-Chips

Es ist laut "Programmer's Reference Guide" von Commodore möglich, Sound-Chips zu "verketten". Das geht so: der Audio Out-Pin des ersten Sound-Chips wird mit dem Audio In-Pin des nächsten Sound-Chips verbunden, dessen Audio Out-Pin mit dem Audio In-Pin des übernächsten usw.

Allerdings ist diese Möglichkeit mehr theoretischer Natur, denn wer hat schon zwei Commodore 64. Und Sound-Chips als Einzelteile von Commodore zu beziehen, ist ein Ding der Unmöglichkeit. Commodore ist froh, wenn die Sound-Chips, die produziert werden, für die zu liefernden Commodore 64er ausreichen.

Eine weiteres Problem beim Koppeln von Sound-Chips ist mehr hardwaremäßiger Natur. Der Sound-Chip, der die Töne ja auf digitaler Basis erzeugt, produziert sogenannten "digital noise" (engl. digitales Rauschen), der in der Struktur des Sound-Chips begründet ist. Denn die Oszillatoren des Sound-Chips sind nicht die besten, mit analogen Oszillatoren können Sie schon gar nicht mithalten. Bei einem Sound-Chip fällt dieser "digital noise" nicht weiter auf, aber beim Koppeln von mehreren Sound-Chips potenziert er sich.

Es soll angeblich (das ist wieder ein Gerücht) in Amerika eine Tastatur geben, die zusätzliche Sound-Chips enthält und die dann mehrstimmig polyphon spielbar ist. Doch aufgrund des obengenannten "digital noise" wird die Tonqualität nicht gerade die beste sein.

Der Anschluß des Commodore 64 an polyphone Synthesizer

Zum Abschluß dieses Anhangs soll auf eine musikalische Anwendung des Commodore 64 OHNE Anwendung des internen Sound-Chips eingegangen werden. Es handelt sich hierbei um die Möglichkeit, polyphone Synthesizer an den Commodore 64 anzuschließen.

Polyphone Synthesizer von Firmen wie Roland, Yamaha, Korg, Sequential Circuits u.v.a.m. besitzen ein sogenanntes "MIDI"-Interface zum Anschluß an weitere Synthesizer und auch Home-Computer. Nun hat der Commodore 64 kein MIDI-Interface, aber es existieren bereits zusätzliche Interfaces, die zwischen MIDI-Interface am polyphonen Synthesizer und User Port des Commodore 64 gesteckt werden.

Für diese Interfaces wurde bereits Software entwickelt. Jedoch ist es hier mit der Auswahl nicht sehr weit her, da polyphone Synthesizer doch recht teure Geräte sind und das ganze nicht sehr verbreitet ist.

Da es abzusehen ist, daß Interfaces und Software für die Verbindung von polyphonen Synthesizern und Home-Computern sich noch mehrfach ändern werden, weil die Entwicklung hier zur sehr im Fluß ist, soll an dieser Stelle nicht auf bestimmte Interfaces und Software eingegangen werden. Beides ist, wie gesagt, nur kurzlebig. Es soll vielmehr von den immensen Möglichkeiten die Rede sein, die Computer als Zusatzgeräte zu polyphonen Synthesizern bieten.

Die trivialste Anwendung eines an einen polyphonen Synthesizer gekoppelten Computers ist, den Computer zum Sequenzer zu machen. Hier muß am Computer mühselig Note für Note eingegeben werden, was sehr viel Zeit kostet. Wenn das Stück fertig ist, kann man es nur schwer wieder verändern.

Eine Stufe weiter ist der sogenannte "Real-Time-Sequenzer", bei dem es möglich ist, ein Musikstück stimmenweise einzugeben und später einzelne Stimmen wieder zu verändern. Die jeweligen Stimmen werden hierbei über die Tastatur des polyphonen Synthesizers eingegeben, was ein enormer Fortschritt ist, da man sich nicht mehr mit Tondauern und Tonhöhen herumschlagen muß. Allerdings liegt hier viel an der Software, wie bequem dann die Stimmeneingabe abläuft. Insbesondere die Editiermöglichkeiten sind bei vielen Programmen sehr eingeschränkt (um nicht zu sagen beschränkt).

Die am weitesten entwickelte Form der Speicherung der musikalischen Information besteht darin, daß man den Computer zum "Tonbandgerät" macht. Der Computer nimmt dann alles auf, was man auf der Tastatur des Synthesizers spielt (auch mehrstimmig polyphon), und speichert es. Später kann man das ganze wieder abrufen, indem die Informationen vom Computer zum Synthesizer zurückübertragen werden. Das Stück wird dann genauso abgespielt, wie es aufgenommen wurde. Von einer Klangverschlechterung kann nicht die Rede sein, da ja nicht der Ton, wie er aus dem Lautsprecher kommt, aufgenommen wurde, sondern sämtliche Tastendruckinformationen. Für diese Anwendung gibt es noch keinen Namen, geschweige denn entsprechende Software.

Anhang 3: Kleines Lexikon zur Computermusik

Viele neue und unbekannte Begriffe stürmten in diesem Buch auf Sie ein. Anstelle eines Registers finden Sie in diesem Anhang alle wichtigen Begriffe, die aus dem Bereich der Computermusik stammen, noch einmal kurz erläutert.

Verweise der Begriffe untereinander werden durch (s.d.), das heißt "siehe dort", gekennzeichnet.

A

Adresse	die eindeutige Identifizierung von Speicherplätzen, an denen Daten (s.d.) vorliegen
ADSR-Werte	(Attack Decay Sustain Release-Werte) Parameter, die den lautstärkemäßigen Verlauf eines Tons angeben
A/D-Wandler	(Analog/Digital-Wandler) Schaltung zur Umwandlung eines in analoger Form vorliegenden Signals (z.B. Musik) in einen entsprechenden Digitalwert
Algorithmus	eine Rechenvorschrift zur Lösung einer Aufgabe in einer Reihe von Einzelschritten
analog	bezeichnet kontinuierlich ablaufende Vorgänge, wie z.B. Zeit (Gegensatz: digital)
AND-Verknüpfung	binäre Rechenoperation. Das Ergebnis der AND-Verknüpfung von zwei Binärzahlen ist das logische UND der einzelnen Bits (Beispiel: 100100 AND 001001 = 101101)

Arrangement	Umarbeitung eines bereits komponierten Musikstücks für eine bestimmte Besetzung, kann auch Änderung der Abfolge und Wiederholungen einzelner Abschnitte eines Musikstücks bedeuten
Array	Datenstruktur, bei der jedes Datenelement eindeutig durch einen Index (s.d.) bestimmt ist
ASCII	(American Standard Code for Information Interchange = engl. amerikanischer Standardcode für den Informationsaustausch) ist der am häufigsten in Kleincomputern angewandte Code zur Darstellung von Zeichen in Form von Zahlenwerten (z.B. A=64, X=88)
Assembler	1. maschinenorientierte Programmiersprache (Einsatz z.B. bei der Herstellung von zeitkritischen Programmen), abhängig vom verwendeten Mikroprozessor 2. ein Programm, das einen mithilfe eines Editors (s.d.) erstellten Quelltext (s.d.) in Maschinensprache (s.d.) übersetzt
Attack	ein Zeitmaß für die Einschwingphase eines Tons
B	
Bandpass	ein Filtertyp, der ein Frequenz-"Band" aus einem Tonsignal ausfiltert
BASIC	(Beginner's All purpose Symbolic Instruction Code) die häufigste Programmiersprache im Bereich der Home Computer

Baßschlüssel	Zeichen am Anfang von Notenlinien, das angibt, daß die folgenden Töne tiefe Töne sind
Bildschirmspeicher	Speicherbereich im Computer, der alle auf dem Bildschirm abgebildeten Zeichen in Form von Bildschirmcodes enthält
Bit	(Binary digIT) kleinste Informationseinheit, Basis des Dualsystems, kann nur zwei Zustände annehmen (0=aus, 1=an)
C	
Chip	allgemeine Bezeichnung für ICs, Mikroprozessoren, I/O-Bausteine usw.
Coda	Schlußteil eines Musikstücks
Counter	(Zähler) spezielle Variable (s.d.), die laufend erhöht (inkrementiert) oder erniedrigt (dekrementiert) wird
Cover-Version	wenn ein Musiker/eine Gruppe ein Stück, das von einem anderen Musiker/Gruppe komponiert wurde, nachspielt und auf Platte aufnimmt, so ist dieses Musikstück eine "Cover-Version" des Originalstücks
D	
Daten	(Einzahl: Datum) für Computer aufbereitete Informationen
Datenfeld	1. Array (s.d.) 2. Datenblock (s.d.)
Datenblock	Programmteil, der vom Programm benötigte feste Daten (s.d.) enthält

Decay	Zeitmaß für die Abschwingphase eines Tons
Deklarationen	Teil eines Programms vor der Übersetzung in den Maschinencode, das die Konstanten und Variablen eines Programms definiert
dekrementieren	Zahl/Variable/Register um 1 erniedrigen
Delimiter	letzter Wert eines Datenblocks (s.d.), der einen Datenblock begrenzt und ihn vom nächsten trennt
digital	lediglich aus Nullen und Einsen bestehend
Dreieck	Wellenform; dumpfer, flötenähnlicher Klang
Dynamik	musikalische Bezeichnung für die sich im Verlauf eines Musikstücks ändernde Lautstärke
E	
Editor	1. Programm zur Bildschirmverwaltung 2. Programm zum Erstellen von Quelltexten (s.d.)
Editieren	Eingeben und Bearbeiten von Programmen und Daten
enharmonische Verwechslung	musikalischer Fachbegriff für die Tatsache, daß in der sogenannten temperierten Stimmung bestimmte, nach musikalischer Funktion und Notation verschiedene Halbtöne (s.d.) die gleiche Tonfrequenz haben (Beispiel: die Halbtöne Cis und Des sind enharmonisch verwechselbar)

F

Filter — Schaltung, die dem Spektrum (s.d.) eines vorgegebenen Tonsignals bestimmte Frequenzen entfernt und andere durchläßt

fix — Programmzeilen in Musikprogrammen dieses Buchs, die nicht verändert werden dürfen, wenn man das Programm ein anderes Musikstück spielen lassen will, heißen fix

Flag — bestimmte Variable (s.d.), die anzeigt, ob eine Bedingung erfüllt ist oder nicht

Frequenz — Anzahl der Schwingungen einer Welle pro Zeiteinheit, in der Musik gleichbedeutend mit Tonhöhe

Fußlage — Bezeichnungsweise von Oktaven (s.d.), die von alten Pfeifenorgeln herstammt, wo die Länge der Pfeifen in Fuß gemessen wurde

G

Gate-Signal — Signal, das angibt, ob der Ton auf dem Sustain-Pegel (s.d.) verweilen soll (dann ist gate = 1) oder ob die Release-Phase (s.d.) gestartet werden soll (dann ist gate = 0)

Grundton — tiefste Komponente eines musikalischen Tons, der sich aus Grundton und Obertönen (s.d.) zusammensetzt. Der Grundton ist der lauteste Bestandteil eines musikalischen Tons, die Tonfrequenz ist die Frequenz des Grundtons.

H

Halbton	eine Oktave (s.d.) wird in der temperierten Stimmung in zwölf Halbtöne unterteilt und in acht Ganztöne. Diejenigen Halbtöne, die zugleich Ganztöne sind, sind auf einer Klaviertastatur als weiße Tasten angeordnet, diejenigen Halbtöne, die nicht zugleich Ganztöne sind, werden durch die schwarzen Tasten gespielt
high	logischer Wert eines Signals (logisch 1)
Hochpass	Filtertyp, bei dem tiefe Frequenzen abgeschnitten werden und hohe passieren. Ergebnis ist ein sehr heller Klang.
Hüllkurve	grafische Entsprechung für ADSR-Werte

I

Improvisation	musikalische Tätigkeit, bei der der Instrumentalist ein Musikstück im Augenblick des Spielens komponiert
Index	(Mehrzahl: Indizes) Wert, der die Position einer Variable (s.d.) innerhalb eines Arrays (s.d.) angibt
Initialisierung	Anfangsteil eines Programms, bei dem die Variablen eines Programms definierte Anfangswerte erhalten
inkrementieren	Zahl/Variable/Register um 1 erhöhen
Interface	(engl. Schnittstelle) Schaltung zur Datenübertragung von einem Computer zu Peripherie oder anderen Computern

Interpret	Musiker, der ein fremdes (d.h. nicht von ihm selbst komponiertes) Musikstück spielt
Interpreter	Programm, das einen Programmtext während der Abarbeitung des Programms Befehl für Befehl in Maschinencode übersetzt, diesen ausführt, aber nirgends abspeichert
Interrupt	Programm, das unabhängig vom Programm im Hauptspeicher abläuft und in bestimmten Intervallen, die von einem Timer (s.d.) eines I/O-Bausteins (s.d.) bestimmt werden, zur Ausführung kommt, währenddessen das Programm im Hauptspeicher aussetzt. Im Commodore 64 werden Cursorblinken und Tastaturabfrage durch einen Interrupt realisiert. Startadresse dieses Programms, das ausschließlich in Maschinencode erstellt ist, werden durch einen Vektor (s.d.) angegeben, der beim Commodore 64 die Speicherzellen $0314 (dez. 788) und $0315 (dez. 789) belegt
Intro	Anfangsteil eines Musikstücks
I/O-Baustein	spezieller Chip, der die Eingabe-/Ausgabe-Operationen (Input/Output) eines Computers abwickelt und somit die Peripherie steuert. Der Commodore 64 hat zwei I/O-Bausteine vom Typ 6526, die für die Tastaturabfrage, die Cassettenrecorder-Bedienung und die Steuerung der seriellen Schnittstelle, der Game-Ports (Joystick-Abfrage) und des User-Ports verantwortlich sind. Zur Kontrollierung der Funktionen der I/O-Bausteine besitzen diese eine bestimmte Anzahl Register (s.d.).

J

Jiffy Clock — englische Bezeichnung für die interruptgesteuerte Uhr des Commodore 64, die von Basic aus durch die Variablen TI und TI$ lesbar ist. Die Jiffy Clock belegt drei Speicherzellen der Zero Page (s.d.) von $A0 bis $A2 (dez. 160 bis 162)

K

Keyboard — englisches Wort für Tastaturen aller Art, insbesondere Klaviertastaturen und im übertragenen Sinn auch Tasteninstrument

L

Label — symbolische Adresse (s.d.) in einem Assembler-Quelltext (s.d.), deren absoluter Wert erst während der Übersetzung in den Maschinencode ermittelt wird

Leerstring — String (s.d.) der Länge 0

LFO — (Low Frequency Oszillator = engl. Tieffrequenzoszillator)
spezieller Oszillator (s.d.), der sehr tiefe Frequenzen (0,1 Hz bis 20 Hz) erzeugt, die als Töne unhörbar sind und die für die Modulation (s.d.) anderer Frequenzen benutzt werden, z.B. der Filterfrequenz oder der Frequenz eines anderen Oszillators

linear — ohne Wiederholung kontinuierlich fortlaufend

low — logischer Wert eines Signals (logisch 0)

LSB — (Least Significant Bit)
niederwertigstes Bit einer Binärzahl

M

Maschinencode — Folge von Hexadezimalzahlen, die vom Prozessor als Programm ausgeführt werden

Maschinensprache — Sprache des Mikroprozessors, setzt sich aus wenigen, hardwarespezifischen Befehlen zusammen. Auf der Ebene der Maschinensprache arbeitet man hauptsächlich mit den prozessoreigenen Registern (s.d.) und Befehlen zum Lesen und Beschreiben von Speicherinhalten

memory mapped I/O — (nicht übersetzbar) Prinzip, die Register von I/O-Bausteinen (s.d.) als Speicherzellen vom Prozessor aus ansprechen zu können, also die Register mit PEEK und POKE lesen und beschreiben zu können

MIDI — (Musical Instruments Digital Interface = engl. digitale Schnittstelle für Musikinstrumente)
Schnittstellennorm, auf die sich große Synthesizerhersteller wie Sequential Circuits (Prophet-Synthesizer), Roland, Yamaha, Korg und andere geeignet haben, zur Kopplung von Synthesizern untereinander und mit Home Computern. MIDI ist eine serielle Schnittstelle mit 3000 bit pro Sekunde und 16 anwählbaren Kanälen.

Modulation — Veränderung der Tonfrequenz eines Oszillators durch einen anderen.
Der eine Oszillator produziert hierbei ein Tonsignal, während die Frequenz des anderen laufend hinzuaddiert wird.

MSB (Most Significant Bit)
höchstwertigstes Bit einer Binärzahl

N

NMI (Non Maskable Interrupt)
Durch Drücken von RUN/STOP RESTORE lösen Sie einen NMI (Warmstart) aus. Ein NMI ist ein besonderer Interrupt, den der Benutzer auslösen kann, um den Registern von I/O-Bausteinen bestimmte definierte Werte zuzuweisen. So werden bei einem NMI z.B. alle Sound-Chip-Register auf 0 gesetzt, wodurch man ein Musikstück unterbrechen kann, ohne daß (wie bei RUN/STOP allein) die zuletzt gespielten Tonfrequenzen "stehenbleiben"

Note musikalische Bezeichnung für die Frequenz und die Dauer eines Tons

O

Obertöne Gemisch aus Sinustönen (s.d.), deren Frequenzen ganzzahlige Vielfache der Frequenz des Grundtons (s.d.) sind. Die Lautstärke der Obertöne untereinander und im Verhältnis zum Grundton sind für den "Klang" eines Tons verantwortlich. Die Lautstärke der Obertöne wird bei konventionellen Synthesizern durch die Wellenform der Oszillatoren bestimmt

Object-Code anderes Wort für Maschinencode (s.d.)

Oktave Abstand von acht Ganztönen oder zwölf Halbtönen. Die Frequenz eines Tons verdoppelt sich, wenn man ihn oktaviert

Option	englisches Wort für Auswahlmöglichkeit
Ostinato	sich immer wiederholende Baßlinie
Oszillator	elektronischer Schwingkreis
P	
Peak	kurzer Lautstärkeimpuls
Peripherie	Sammelbegriff für alle an einen Computer anschließbaren Geräte wie Drucker, Floppy usw.
Pin	1. "Beinchen" eines Chips 2. signalführendes Ende eines Steckers oder einer Buchse
Pointer	(engl. Zeiger) spezielle Variable (s.d.), die auf eine andere Variable verweist
Programm	Umsetzung eines Algorithmus' in eine einem Computer verständliche Form
Prozessor	(oder Mikroprozessor) aus Leit- und Rechenwerk bestehende Funktionseinheit einer digitalen Rechenanlage
Pulsbreite	Parameter der Rechteckschwingung, der die Obertonzusammensetzung bestimmt
Pulswelle	anderes Wort für Rechteck (s.d.)
PWM	(Pulse Width Modulation = engl. Pulsbreitenmodulation) Modulation der Pulsbreite (z.B. durch einen LFO)

Q

Quelltext — Bezeichnung für die Form, in der ein mit einem Editor (s.d.) erstelltes Programm im Speicher oder auf Diskette steht. Dies ist die lesbare und editierbare Form des Programms, die vor der Ausführung erst noch übersetzt werden muß

Q-Wert — anderes Wort für Resonanz (s.d.)

R

RAM — (Random Access Memory)
frei les- und beschreibbarer Speicher
enthält Benutzerprogramme und -daten

Rauschen — Gemisch aus vielen Sinustönen mit gleicher Lautstärke

Read-Write-Register — Register (s.d.), das vom Prozessor sowohl gelesen als auch beschrieben werden kann

Rechteck — Wellenform, die alle ungeradzahligen Obertöne enthält, deren Lautstärke durch die Pulsbreite kontrolliert wird

Register — Gruppe aus zumeist acht Bits, die
1. prozessorinterne Daten halten
2. Daten eines I/O-Bausteins lesen oder verändern können

Release — Zeitmaß für das entgültige Ausklingen eines Tons nach Löschen des Gate-Signals

Resonanz — Parameter, der bestimmt, in welchem Maß der Filter selbst als Oszillator arbeitet

Ring Modulator	Einrichtung vieler Synthesizer, die es erlaubt, mit einem normalen Oszillator, der im Audio-Bereich schwingt, einen anderen Oszillator zu modulieren. Die Modulation ergibt sich hier ausnahmsweise durch die multiplikative Verknüpfung der Frequenzen beider Oszillatoren
Ritardando	musikalischer Ausdruck für das Herabsetzen des Tempos im Verlauf eines Musikstücks
Routine	bestimmtes Programmstück, das oft zu wiederholende Operationen enthält

S

Sägezahn	Wellenform, bei der der n-te Oberton 1/n mal so laut wie der Grundton ist
Sequenzer	Gerät oder auch Programm, in dem die Tonhöhen und Tondauern eines Stücks exakt eingegeben werden müssen, und das nach Anschluß an einen Synthesizer dieses Stück über den Synthesizer spielt, ohne daß dazu eine Tastatur gebraucht wird
Sinuston	ein Grundton ohne Obertöne
Source	englische Bezeichnung für Quelltext (s.d.)
Speicherzelle	Element des Speichers, das durch die Adresse (s.d.) eindeutig bestimmt wird
Spektrum	Menge aller möglichen Frequenzen
String	englisches Wort für Zeichenkette

Sustain	englisches Wort für den Haltepegel, auf dem der Ton bei gesetztem Gate-Signal nach Durchlauf der Attack- und Decay-Phasen verbleibt
Synchronisation	Möglichkeit vieler Synthesizer, einen Oszillator durch einen anderen, der im Audio-Bereich schwingt, zu modulieren.
Synthesizer	Musikinstrument, das durch Oszillatoren, Filter, Hüllkurvengeneratoren und Verstärker den Klang natürlicher Musikinstrumente nachbilden und neue Klänge synthetisieren kann

T

Test-Bit	spezielles Kontrollbit im Sound-Chip, das es ermöglicht, jeden Oszillator getrennt ein- und auszuschalten
Tiefpass	Filtertyp, bei dem hohe Frequenzen abgeschnitten werden und tiefe passieren. Das Ergebnis ist oft ein dumpfer, bassiger Klang.
Timer	Register eines I/O-Bausteins, die einen oder mehrere Counter (s.d.) laufend hardwaremäßig inkrementieren (s.d.), um damit Zeitabläufe bei I/O-Operationen zu kontrollieren
Tool	(engl. Werkzeug) Kurzwort für Toolkit
Toolkit	(engl. Werkzeugkasten) Programmierhilfe, die neue Basic-Befehle, etwa für die Musikprogrammierung, zur Verfügung stellt

triggern — auf logisch 1 (high) setzen

V

variabel — veränderbar

Variable — Speicherzelle oder Gruppe von Speicherzellen, deren Inhalte sich laufend ändern und die vom Programm zur Speicherung von vorübergehenden Daten benutzt werden

Vektor — meistens zwei Speicherzellen, die eine Sprungadresse zu einem bestimmten Programmstück enthalten (meistens Betriebssystems-Unterroutinen). Durch "Verbiegen" der Vektoren kann man das Betriebssystem manipulieren

Violinschlüssel — Zeichen an Anfang von Notenlinien, das anzeigt, daß hohe Note folgen

Z

Zero Page — (engl. Seite Null)
besonderer Speicherbereich bei Computern, die von einem Mikroprozessor der Familie 65XX gesteuert werden. Der Vorteil der Zero Page äußert sich in besonderen, verkürzten Register-Lade- und Speicher-Befehlen für alle Adressen von 0 bis 255. Üblicherweise werden von einem 65XX-Computer diese Adressen zur Speicherung der Variablen des Betriebssystems und des Basic-Interpreters genutzt.
Durch verschiedene POKE-Befehle kann der Benutzer daher das Betriebssystem manipulieren.

Anhang 4:

```
0 REM  "SGT. PEPPER'S LONELY HEARTS CLUB BAND"
1 REM  ARRANGED BY  THOMAS DACHSEL
2 REM
3 REM
4 FOR  I=54272 TO 54296
5 POKE I,0
6 NEXT I
7 REM
8 REM . DEFINITIONEN
9 REM
10 ER=54272
11 TL=ER
12 TH=ER+1
13 PL=ER+2
14 PH=ER+3
15 WE=ER+4
16 AD=ER+5
17 SR=ER+6
18 LF=ER+21
19 HF=ER+22
20 RE=ER+23
21 LA=ER+24
27 REM
28 REM  FREQUENZEN LESEN
29 REM
30 DIM FR$(11),FL(11,4),FH(11,4)
31 FOR I=0 TO 11
32     READ FR$(I)
33     FOR J=0TO 4
34         READ  FR
35         FH(I,J)=INT(FR/256)
36         FL(I,J)=FR-FH(I,J)*256
37     NEXTJ
38 NEXTI
47 REM
48 REM  NOTENDATEN LESEN
49 REM
50 DIMN1$(250),O1(250),D1(250)
```

```
51 DIMN2$(250),O2(250),D2(250)
52 DIMN3$(250),O3(250),D3(250)
53 I=0
54 READN1$(I),O1(I),D1(I):IFD1(I)>0THENI=I+1:GOTO54
55 I=0
56 READN2$(I),O2(I),D2(I):IFD2(I)>0THENI=I+1:GOTO56
57 I=0
58 READN3$(I),O3(I),D3(I):IFD3(I)>0THENI=I+1:GOTO58
59 REM
60 REM  UMRECHNUNGSROUTINE
61 REM
62 DIML1(250),H1(250),L2(250),H2(250),L3(250),H3(250)
63 I=0:PRINTCHR$(147)"1.STIMME:"
64 FOR J=0TO11
65 IF FR$(J)=N1$(I)THENL1(I)=FL(J,O1(I)-2):H1(I)=FH(J,O1(I)-2):PRINTN1$(I);
66 NEXT
67 I=I+1:IFD1(I)>0THEN64
68 I=0:PRINT:PRINT"2.STIMME:"
69 FOR J=0TO11
70 IF FR$(J)=N2$(I)THENL2(I)=FL(J,O2(I)-2):H2(I)=FH(J,O2(I)-2):PRINTN2$(I);
71 NEXT
72 I=I+1:IFD2(I)>0THEN69
73 I=0:PRINT:PRINT"3.STIMME:"
74 FOR J=0TO11
75 IF FR$(J)=N3$(I)THENL3(I)=FL(J,O3(I)-2):H3(I)=FH(J,O3(I)-2):PRINTN3$(I);
76 NEXT
77 I=I+1:IFD3(I)>0THEN74
87 REM
88 REM  ANFANGSKLANG EINSTELLEN
89 REM
90 POKEAD,41:POKEAD+7,41:POKEAD+14,7
91 POKESR,121:POKESR+7,121:POKESR+14,118
92 POKELA,15:POKERE,0:POKEPH,3
93 W1=64:W2=32:W3=32
94 FLAG=0
95 S1=0:S2=0:S3=0:REM INDIZES
96 C1=0:C2=0:C3=0:REM COUNTER
100 REM
101 REM STUECK SPIELEN
```

```
102 REM
103 POKETL,L1(S1):POKETH,H1(S1):POKEWE,W1OR1
104 POKETL+7,L2(S2):POKETH+7,H2(S2):POKEWE+7,W2 OR 1
105 POKETL+14,L3(S3):POKETH+14,H3(S3):POKEWE+14,W3 OR 1
110 C1=C1+1:IFC1>(D1(S1)*.5)THENPOKEWE,W1
111 IFC1<(D1(S1))THEN120
112 C1=0:S1=S1+1:POKETL,L1(S1):POKETH,H1(S1):POKEWE,W1OR1
120 C2=C2+1:IFC2>(D2(S2)*.7)THENPOKEWE+7,W2
121 IFC2<(D2(S2))THEN130
122 C2=0:S2=S2+1:POKETL+7,L2(S2):POKETH+7,H2(S2):POKEWE+7,W2OR1
130 C3=C3+1:IFC3>(D3(S3)*.8)THENPOKEWE+14,W3
131 IFC3<(D3(S3))THEN140
132 C3=0:S3=S3+1:POKETL+14,L3(S3):POKETH+14,H3(S3):POKEWE+14,W3OR1
140 IF S1=220 THEN IF C1=1 THEN FLAG=1:S1=28:S2=15:S3=26:GOTO96
141 IF S1=91 THEN IF FLAG=1 THEN IF C1=1 THEN 151
150 IFD1(S1)>0ORD2(S2)>0ORD3(S3)>0 THEN 110
151 PRINTCHR$(147)"NOCHMAL? (J)":POKE198,0:WAIT198,1:GETC$:IFC$<>"J"THENEND
152 GOTO90
200 REM FREQUENZEN
201 DATAC,1072,2145,4291,8583,17167
202 DATAC#,1136,2273,4547,9094,18188
203 DATAD,1206,2408,4817,9634,19269
204 DATAD#,1275,2551,5103,10207,20415
205 DATAE,1351,2703,5407,10814,21629
206 DATAF,1432,2864,5728,11457,22915
207 DATAF#,1517,3034,6069,12139,24278
208 DATAG,1607,3215,6430,12860,25721
209 DATAG#,1703,3406,6812,13625,27251
210 DATAA,1804,3608,7217,14435,28871
211 DATAA#,1911,3823,7647,15294,30588
212 DATAB,2025,4050,8101,16203,32407
1000 REM 1.STIMME
1001 DATAC,5,4
1002 DATAC,5,4
1003 DATAA,4,1
1004 DATAC,5,1
1005 DATAD,5,1
1006 DATAD#,5,1
1007 DATAD,5,1
```

```
1008 DATAC,5,3
1009 DATAC,5,4
1010 DATAC,5,4
1011 DATAA,4,1
1012 DATAC,5,1
1013 DATAD,5,1
1014 DATAD#,5,1
1015 DATAD,5,1
1016 DATAC,5,3
1017 DATA,0,4
1018 DATAA,4,1
1019 DATAC,5,1
1020 DATAD#,5,1
1021 DATAF,5,1
1022 DATAF,5,2
1023 DATAG,5,2
1024 DATAA#,4,2
1025 DATAC,5,2
1026 DATAG,4,14
1027 DATAC,5,1
1028 DATAB,4,1
1029 DATAC,5,1
1030 DATAB,4,1
1031 DATAC,5,1
1032 DATAB,4,1
1033 DATAC,5,2
1034 DATAC,5,1
1035 DATAC,5,6
1036 DATAB,4,1
1037 DATAC,5,1
1038 DATAC,5,1
1039 DATAC,5,1
1040 DATAC,5,1
1041 DATAC,5,1
1042 DATAC,5,1
1043 DATAC,5,2
1044 DATAC,5,1
1045 DATAC,5,7
1046 DATAC,5,1
1047 DATAB,4,1
1048 DATAC,5,1
1049 DATAC,5,1
1050 DATAC,5,1
1051 DATAC,5,1
1052 DATAC,5,2
1053 DATAB,4,1
1054 DATAC,5,7
1055 DATAC,5,1
1056 DATAC,5,1
1057 DATAC,5,1
1058 DATAC,5,1
1059 DATAC,5,1
1060 DATAC,5,1
1061 DATAC,5,2
1062 DATAB,4,1
1063 DATAC,5,7
1064 DATA,0,1
1065 DATAC,5,1
1066 DATAD,5,1
1067 DATAD,5,1
1068 DATAD,5,1
1069 DATAC,5,1
1070 DATAD,5,2
1071 DATAC,5,1
1072 DATAD,5,7
1073 DATA,0,1
1074 DATAC,5,1
1075 DATAD,5,1
1076 DATAC,5,1
1077 DATAD,5,1
1078 DATAC,5,1
1079 DATAD#,5,1
1080 DATAD#,5,2
1081 DATAD#,5,9
1082 DATAD#,5,2
1083 DATAD#,5,2
1084 DATAD#,5,1
1085 DATAD#,5,3
```

```
1086 DATAD#,5,2
1087 DATAD#,5,2
1088 DATAD#,5,2
1089 DATAD#,5,2
1090 DATAD#,5,8
1091 DATAC,5,6
1092 DATA,0,2
1093 DATA,0,4
1094 DATAD#,5,2
1095 DATAD,5,2
1096 DATAC,5,2
1097 DATAA,4,2
1098 DATAF,4,2
1099 DATAD#,4,2
1100 DATAD,4,2
1101 DATAF,4,2
1102 DATAA#,4,4
1103 DATAA#,4,8
1104 DATA,0,4
1105 DATAF,5,2
1106 DATAD#,5,2
1107 DATAD,5,2
1108 DATAC,5,2
1109 DATAA#,4,2
1110 DATAA,4,2
1111 DATAG,4,2
1112 DATAA,4,2
1113 DATAB,4,2
1114 DATAD,5,2
1115 DATAF,5,8
1116 DATAG,4,1
1117 DATAF#,4,1
1118 DATAG,4,1
1119 DATAA,4,1
1120 DATAB,4,1
1121 DATAG,4,1
1122 DATAB,4,1
1123 DATAD,5,1
1124 DATAG,5,6
1125 DATAE,4,2
1126 DATAG,4,2
1127 DATAG,4,2
1128 DATAG,4,2
1129 DATAG,4,2
1130 DATAC,5,4
1131 DATAC,5,2
1132 DATAA#,4,4
1133 DATAA,4,4
1134 DATAG,4,8
1135 DATAG,4,2
1136 DATAA,4,2
1137 DATAA,4,2
1138 DATAA,4,2
1139 DATAA,4,2
1140 DATAA,4,4
1141 DATAA,4,2
1142 DATAG,4,16
1143 DATAF#,4,2
1144 DATAG,4,2
1145 DATAG,4,2
1146 DATAG,4,2
1147 DATAG,4,2
1148 DATAC,5,4
1149 DATAC,5,2
1150 DATAA#,4,4
1151 DATAA,4,4
1152 DATAG,4,8
1153 DATAG,4,2
1154 DATAA,4,2
1155 DATAA,4,2
1156 DATAA,4,2
1157 DATAA,4,2
1158 DATAA,4,4
1159 DATAA,4,2
1160 DATAA#,4,4
1161 DATAG,4,10
1162 DATA,0,2
1163 DATAA,4,4
```

```
1164 DATAA,4,2
1165 DATAA,4,2
1166 DATAA,4,2
1167 DATAA,4,4
1168 DATAA,4,2
1169 DATAG,4,4
1170 DATAG,4,2
1171 DATAG,4,2
1172 DATAG,4,2
1173 DATAG,4,4
1174 DATAG,4,2
1175 DATAA,4,4
1176 DATAA,4,2
1177 DATAA,4,2
1178 DATAA,4,2
1179 DATAA,4,4
1180 DATAA,4,2
1181 DATAA,4,4
1182 DATAA,4,4
1183 DATAG,4,8
1184 DATAG,4,2
1185 DATAA,4,2
1186 DATAA,4,2
1187 DATAA,4,2
1188 DATAA,4,2
1189 DATAA,4,2
1190 DATAA,4,4
1191 DATAA,4,2
1192 DATAG#,4,2
1193 DATAG#,4,2
1194 DATAG#,4,2
1195 DATAG#,4,2
1196 DATAG#,4,6
1197 DATAG#,4,2
1198 DATAA,4,2
1199 DATAA,4,2
1200 DATAA,4,2
1201 DATAA,4,2
1202 DATAA,4,2
1203 DATAA,4,2
1204 DATAA,4,2
1205 DATAA,4,2
1206 DATAB,4,2
1207 DATAB,4,2
1208 DATAB,4,2
1209 DATAB,4,2
1210 DATAB,4,2
1211 DATAB,4,2
1212 DATAB,4,2
1213 DATAB,4,2
1214 DATAB,4,2
1215 DATAB,4,2
1216 DATAB,4,2
1217 DATAB,4,2
1218 DATAB,4,4
1219 DATA,0,1
1220 DATAC,5,1
1221 DATAC,5,2
1999 DATA,-1,-1
2000 REM 2.STIMME
2001 DATAF#,4,4
2002 DATAF#,4,4
2003 DATAF#,4,8
2004 DATAF#,4,4
2005 DATAF#,4,4
2006 DATAF#,4,8
2007 DATA,0,4
2008 DATAF,4,2
2009 DATAA,4,2
2010 DATAA,4,2
2011 DATAA,4,2
2012 DATAD#,4,2
2013 DATAD#,4,2
2014 DATAE,4,14
2015 DATAE,4,2
2016 DATAE,4,4
2017 DATAE,4,2
2018 DATAE,4,1
```

```
2019 DATAF#,4,7
2020 DATAF#,4,2
2021 DATAF,4,4
2022 DATAF,4,2
2023 DATAF,4,1
2024 DATAE,4,7
2025 DATAE,4,2
2026 DATAE,4,4
2027 DATAE,4,2
2028 DATAE,4,1
2029 DATAF#,4,7
2030 DATAF#,4,2
2031 DATAF,4,4
2032 DATAF,4,2
2033 DATAF,4,1
2034 DATAE,4,7
2035 DATA,0,2
2036 DATAF#,4,4
2037 DATAF#,4,2
2038 DATAF#,4,1
2039 DATAF#,4,7
2040 DATA,0,2
2041 DATAA,4,4
2042 DATAA,4,1
2043 DATAA,4,2
2044 DATAA,4,9
2045 DATAE,4,8
2046 DATAF,4,8
2047 DATAE,4,14
2048 DATA,0,2
2049 DATA,0,4
2050 DATAA,4,4
2051 DATAD#,4,4
2052 DATAA,3,4
2053 DATAA#,3,4
2054 DATAD,4,4
2055 DATAD,4,8
2056 DATA,0,4
2057 DATAA,4,4
2058 DATAA,4,4
2059 DATAD#,4,4
2060 DATAB,3,4
2061 DATAD,4,2
2062 DATAG,4,2
2063 DATAB,4,8
2064 DATAG,3,8
2065 DATAB,4,6
2066 DATA,0,2
2067 DATAE,4,8
2068 DATAG,4,4
2069 DATAF,4,2
2070 DATAD#,4,4
2071 DATAD#,4,4
2072 DATAE,4,8
2073 DATAE,4,2
2074 DATAD#,4,8
2075 DATAD#,4,4
2076 DATAD#,4,2
2077 DATAE,4,16
2078 DATA,0,2
2079 DATAE,4,8
2080 DATAG,4,4
2081 DATAF,4,2
2082 DATAD#,4,4
2083 DATAD#,4,4
2084 DATAE,4,8
2085 DATAE,4,2
2086 DATAF#,4,8
2087 DATAF#,4,4
2088 DATAF#,4,2
2089 DATAB,3,14
2090 DATA,0,2
2091 DATAC,4,4
2092 DATAC,4,2
2093 DATAC,4,2
2094 DATAC,4,2
2095 DATAC,4,4
2096 DATAC,4,2
```

```
2097 DATAE,4,4
2098 DATAE,4,2
2099 DATAE,4,2
2100 DATAE,4,2
2101 DATAE,4,4
2102 DATAE,4,2
2103 DATAF#,4,4
2104 DATAF#,4,2
2105 DATAF#,4,2
2106 DATAF#,4,2
2107 DATAF#,4,4
2108 DATAF#,4,2
2109 DATAC,4,4
2110 DATAC,4,4
2111 DATAE,4,8
2112 DATAE,4,2
2113 DATAC,4,2
2114 DATAC,4,2
2115 DATAC,4,2
2116 DATAC,4,2
2117 DATAC,4,2
2118 DATAC,4,4
2119 DATAD#,4,2
2120 DATAD,4,2
2121 DATAD,4,2
2122 DATAD,4,2
2123 DATAD,4,2
2124 DATAD,4,6
2125 DATAD,4,2
2126 DATAC,4,2
2127 DATAC,4,2
2128 DATAC,4,2
2129 DATAC,4,2
2130 DATAC,4,2
2131 DATAC,4,2
2132 DATAC,4,2
2133 DATAC,4,2
2134 DATAD,4,2
2135 DATAD,4,2
2136 DATAD,4,2
2137 DATAD,4,2
2138 DATAD,4,2
2139 DATAD,4,2
2140 DATAD,4,2
2141 DATAD,4,2
2142 DATAF,4,2
2143 DATAF,4,2
2144 DATAF,4,2
2145 DATAF,4,2
2146 DATAF,4,4
2147 DATA,0,1
2148 DATAF,4,1
2149 DATAF,4,2
2999 DATA,-1,-1
3000 REM 3.STIMME
3001 DATAD,3,2
3002 DATAD,3,2
3003 DATAD,3,2
3004 DATAD,3,2
3005 DATAD,3,8
3006 DATAD,3,2
3007 DATAD,3,2
3008 DATAD,3,2
3009 DATAD,3,2
3010 DATAD,3,8
3011 DATAF,2,2
3012 DATAF,2,2
3013 DATAF,2,2
3014 DATAF,2,2
3015 DATAF,2,2
3016 DATAF,2,2
3017 DATAF,2,2
3018 DATAF,2,2
3019 DATAC,3,2
3020 DATAC,3,2
3021 DATAC,3,2
3022 DATAC,3,2
3023 DATAC,3,2
```

```
3024 DATAC,3,2
3025 DATAC,3,2
3026 DATA,0,2
3027 DATAC,2,4
3028 DATAC,2,2
3029 DATAC,2,2
3030 DATAD,2,2
3031 DATAD,2,2
3032 DATAD,2,2
3033 DATAD,2,2
3034 DATAF,2,4
3035 DATAF,2,2
3036 DATAF,2,2
3037 DATAC,2,2
3038 DATAC,2,2
3039 DATAC,2,2
3040 DATAC,2,2
3041 DATAC,2,4
3042 DATAC,2,2
3043 DATAC,2,2
3044 DATAD,2,2
3045 DATAD,2,2
3046 DATAD,2,2
3047 DATAD,2,2
3048 DATAF,2,4
3049 DATAF,2,2
3050 DATAF,2,2
3051 DATAC,2,2
3052 DATAC,2,2
3053 DATAC,2,2
3054 DATAC,2,2
3055 DATAD,2,4
3056 DATAD,2,2
3057 DATAD,2,2
3058 DATAD,2,2
3059 DATAD,2,2
3060 DATAD,2,2
3061 DATAD,2,2
3062 DATAF,2,4
3063 DATAF,2,2
3064 DATAF,2,2
3065 DATAF,2,2
3066 DATAF,2,2
3067 DATAF,2,2
3068 DATAF,2,2
3069 DATAC,2,6
3070 DATAC,2,2
3071 DATAF,2,8
3072 DATAC,2,2
3073 DATAG,2,2
3074 DATAA,2,2
3075 DATAB,2,2
3076 DATAC,3,2
3077 DATAG,2,2
3078 DATAC,2,2
3079 DATA,0,2
3080 DATAF,2,2
3081 DATAF,2,2
3082 DATAF,2,4
3083 DATAF,3,4
3084 DATAF,2,4
3085 DATAA#,2,2
3086 DATAD,3,2
3087 DATAA#,2,2
3088 DATAF,3,2
3089 DATAA#,2,2
3090 DATAA#,2,2
3091 DATAF,3,2
3092 DATAA#,2,2
3093 DATAF,2,2
3094 DATAF,2,2
3095 DATAF,2,4
3096 DATAF,3,4
3097 DATAF,2,4
3098 DATAG,2,8
3099 DATAG,3,1
3100 DATAF#,3,1
3101 DATAG,3,1
```

```
3102 DATAA,3,1
3103 DATAB,3,1
3104 DATAG,3,1
3105 DATAB,3,1
3106 DATAD,4,1
3107 DATAF,4,8
3108 DATAG,3,2
3109 DATAD,3,2
3110 DATAG,2,2
3111 DATA,0,2
3112 DATAC,3,8
3113 DATAD#,3,4
3114 DATAD,3,2
3115 DATAC,3,4
3116 DATAF,2,4
3117 DATAC,3,4
3118 DATAG,2,2
3119 DATAC,3,2
3120 DATA,0,2
3121 DATAF,2,8
3122 DATAF,2,4
3123 DATAF,2,2
3124 DATAC,3,4
3125 DATAC,3,2
3126 DATAC,3,2
3127 DATAG,2,2
3128 DATAC,3,2
3129 DATAC,3,2
3130 DATAC,3,2
3131 DATA,0,2
3132 DATAC,3,8
3133 DATAD#,3,4
3134 DATAD,3,2
3135 DATAC,3,4
3136 DATAF,2,4
3137 DATAC,3,4
3138 DATAG,2,2
3139 DATAC,3,2
3140 DATA,0,2
3141 DATAD,2,8
3142 DATAD,2,4
3143 DATAD,2,2
3144 DATAG,2,6
3145 DATAG,2,2
3146 DATAD,2,2
3147 DATAG,2,2
3148 DATAG,2,2
3149 DATAG,2,2
3150 DATAF,2,4
3151 DATA,0,2
3152 DATAF,2,2
3153 DATA,0,2
3154 DATAF,2,4
3155 DATAF,2,2
3156 DATAC,2,4
3157 DATA,0,2
3158 DATAC,2,2
3159 DATA,0,2
3160 DATAC,2,4
3161 DATAC,2,2
3162 DATAD,2,4
3163 DATA,0,2
3164 DATAD,2,2
3165 DATA,0,2
3166 DATAD,2,4
3167 DATAD,2,2
3168 DATAF,2,4
3169 DATAF,2,4
3170 DATAC,3,4
3171 DATAG,2,2
3172 DATAC,3,2
3173 DATAG,3,2
3174 DATAF,3,2
3175 DATA,0,2
3176 DATAD#,3,2
3177 DATA,0,2
3178 DATAD,3,2
3179 DATA,0,2
```

```
3180 DATAC,3,2
3181 DATAF,3,2
3182 DATAA#,2,2
3183 DATA,0,2
3184 DATAF,3,2
3185 DATA,0,2
3186 DATAA#,2,2
3187 DATAA#,2,2
3188 DATAF,3,2
3189 DATAA#,2,2
3190 DATAF,3,2
3191 DATA,0,2
3192 DATAD#,3,2
3193 DATA,0,2
3194 DATAD,3,2
3195 DATA,0,2
3196 DATAC,3,2
3197 DATA,0,2
3198 DATAG,2,2
3199 DATA,0,2
3200 DATAD,3,2
3201 DATA,0,2
3202 DATAG,3,2
3203 DATA,0,2
3204 DATAD,3,2
3205 DATA,0,2
3206 DATAG,2,2
3207 DATA,0,2
3208 DATAD,3,2
3209 DATA,0,2
3210 DATAG,3,2
3211 DATAD,3,2
3212 DATAG,2,4
3999 DATA,-1,-1

READY.
```

```
0 REM "SHINE ON YOU CRAZY DIAMOND"
1 REM  ARRANGED BY THOMAS DACHSEL
2 REM ON JANUARY 30TH & 31ST, 1984
3 REM
9 FORI=54272TO54296:POKEI,0:NEXT
10 ER=54272
11 TL=ER
12 TH=ER+1
13 PL=ER+2
14 PH=ER+3
15 WE=ER+4
16 AD=ER+5
17 SR=ER+6
18 LF=ER+21
19 HF=ER+22
20 RE=ER+23
21 LA=ER+24
30 DIM FR$(11),FL(11,4),FH(11,4)
40 FORI=0TO11
41 READFR$(I)
42 FORJ=0TO4
43 READFR
44 FH(I,J)=INT(FR/256)
45 FL(I,J)=FR-FH(I,J)*256
46 NEXTJ,I
50 DIMN1$(320),O1(320),D1(320),N2$(169),O2(169),D2(169),N3$(526),O3(526),D3(52
51 I=0
52 READN1$(I),O1(I),D1(I):IFD1(I)>0THENI=I+1:GOTO52
53 I=0
54 READN2$(I),O2(I),D2(I):IFD2(I)>0THENI=I+1:GOTO54
55 I=0
56 READN3$(I),O3(I),D3(I):IFD3(I)>0THENI=I+1:GOTO56
60 DIML1(320),H1(320),L2(169),H2(169),L3(526),H3(526)
63 I=0:PRINTCHR$(147)"1.STIMME:"
64 FORJ=0TO11
65 IFFR$(J)=N1$(I)THENL1(I)=FL(J,O1(I)-2):H1(I)=FH(J,O1(I)-2):PRINTN1$(I);
66 NEXT
67 I=I+1:IFD1(I)>0THEN64
```

```
68 I=0:PRINT:PRINT"2.STIMME:"
69 FORJ=0TO11
70 IFFR$(J)=N2$(I)THENL2(I)=FL(J,O2(I)-2):H2(I)=FH(J,O2(I)-2):PRINTN2$(I)
71 NEXT
72 I=I+1:IFD2(I)>0THEN69
73 I=0:PRINT:PRINT"3.STIMME:"
74 FORJ=0TO11
75 IFFR$(J)=N3$(I)THENL3(I)=FL(J,O3(I)-2):H3(I)=FH(J,O3(I)-2):PRINTN3$(I);
76 NEXT
77 I=I+1:IFD3(I)>0THEN74
79 REM
80 REM  *** ANFANGSWERTE
81 REM
84 POKEAD,238:POKEAD+7,238:POKEAD+14,204
85 POKESR,175:POKESR+7,175:POKESR+14,238
86 POKEPH+14,3:POKEHF,200:POKERE,4:W1=32:W2=32:W3=64:POKELA,31
100 REM
101 REM *** STUECK SPIELEN
102 REM
103 POKETL,L1(S1):POKETH,H1(S1):POKEWE,W1OR1
104 POKETL+7,L2(S2):POKETH+7,H2(S2):POKEWE+7,W2 OR 1
105 POKETL+14,L3(S3):POKETH+14,H3(S3):POKEWE+14,W3 OR 1
110 C1=C1+1:IFC1>(D1(S1)*.5)THENPOKEWE,W1
111 IFC1<(D1(S1))THEN120
112 C1=0:S1=S1+1:POKEWE,W1OR1:POKETL,L1(S1):POKETH,H1(S1)
120 C2=C2+1:IFC2>(D2(S2)*.5)THENPOKEWE+7,W2
121 IFC2<(D2(S2))THEN130
122 C2=0:S2=S2+1:POKEWE+7,W2OR1:POKETL+7,L2(S2):POKETH+7,H2(S2)
130 C3=C3+1:IFC3>(D3(S3)*.5)THENPOKEWE+14,W3
131 IFC3<(D3(S3))THEN140
132 C3=0:S3=S3+1:POKEWE+14,W3OR1:POKETL+14,L3(S3):POKETH+14,H3(S3)
140 IFS3<>96THEN143
141 POKELA,79:POKEHF,90:POKEAD+14,5:POKESR+14,119:POKERE,244
142 POKEPH+14,1:POKEAD,0:POKEAD+7,0:POKESR,111:POKESR+7,111
143 IFS3<>188THEN145
144 POKELA,31:POKEHFILT,190:POKEAD+14,41:POKESR+14,175:POKEPH+14,4
145 IFS1<>22THEN148
146 POKEAD,0:POKESR,242:POKEAD+7,0:POKESR+7,242
147 W1=64:W2=64:POKEPH,5:POKEPH+7,4
```

```
148 IFS1<>118THEN150
149 POKEAD,7:POKESR,136:POKEAD+7,172:POKESR+7,136:W2=32
150 IFD1(S1)>0ORD2(S2)>0ORD3(S3)>0THEN110
151 PRINTCHR$(147)"NOCHMAL? (J)":POKE198,0:WAIT198,1:GETC$:IFC$<>"J"THENEND
152 C1=0:C2=0:C3=0:S1=0:S2=0:S3=0:GOTO80
200 REM FREQUENZEN
201 DATAC,1072,2145,4291,8583,17167
202 DATAC#,1136,2273,4547,9094,18188
203 DATAD,1206,2408,4817,9634,19269
204 DATAD#,1275,2551,5103,10207,20415
205 DATAE,1351,2703,5407,10814,21629
206 DATAF,1432,2864,5728,11457,22915
207 DATAF#,1517,3034,6069,12139,24278
208 DATAG,1607,3215,6430,12860,25721
209 DATAG#,1703,3406,6812,13625,27251
210 DATAA,1804,3608,7217,14435,28871
211 DATAA#,1911,3823,7647,15294,30588
212 DATAB,2025,4050,8101,16203,32407
10000 REM 1. STIMME
10001 DATAG,2,32
10002 DATAG,2,96
10003 DATAG,2,64
10004 DATAG,2,128
10005 DATAG,2,64
10006 DATAG,2,64
10007 DATAG,2,96
10008 DATAG,2,64
10009 DATAG,2,160
10010 DATAG,2,96
10011 DATAG,2,48
10012 DATAD,2,48
10013 DATAC,2,64
10014 DATAG,2,32
10015 DATAG,2,48
10016 DATAD,2,48
10017 DATAC,2,32
10018 DATAD,2,32
10019 DATAG,2,70
10020 DATAG,2,64
```

```
10021 DATAG,2,64
10022 DATAG,2,32
10023 DATAG,2,1
10024 DATAG,2,1
10025 DATAG,2,1
10026 DATAG,2,1
10027 DATAG,2,1
10028 DATAG,2,1
10029 DATAG,2,1
10030 DATAG,2,1
10031 DATAG,2,1
10032 DATAG,2,1
10033 DATAG,2,1
10034 DATAG,2,1
10035 DATAG,2,1
10036 DATAG,2,1
10037 DATAG,2,1
10038 DATAG,2,1
10039 DATAG,2,1
10040 DATAG,2,1
10041 DATAG,2,1
10042 DATAG,2,1
10043 DATAG,2,1
10044 DATAG,2,1
10045 DATAG,2,2
10046 DATAC,2,6
10047 DATAC,2,3
10048 DATAC,2,1
10049 DATAC,2,2
10050 DATAC,2,6
10051 DATAC,2,2
10052 DATAC,2,2
10053 DATAC,2,2
10054 DATAC,2,6
10055 DATAC,2,3
10056 DATAC,2,1
10057 DATAC,2,2
10058 DATAC,2,2
10059 DATAF,2,1
10060 DATAF,2,1
10061 DATAF,2,1
10062 DATAF,2,1
10063 DATAF,2,1
10064 DATAF,2,1
10065 DATAF,2,1
10066 DATAF,2,1
10067 DATAF,2,1
10068 DATAF,2,1
10069 DATAF,2,1
10071 DATAG,2,5
10072 DATAG,2,3
10073 DATAG,2,1
10074 DATAG,2,2
10075 DATAG,2,6
10076 DATAG,2,6
10077 DATAG,2,6
10078 DATAG,2,3
10079 DATAG,2,1
10080 DATAG,2,2
10081 DATAG,2,6
10082 DATAG,2,6
10083 DATAD#,2,6
10084 DATAD#,2,3
10085 DATAD#,2,1
10086 DATAD#,2,2
10087 DATAD#,2,8
10088 DATAD#,2,2
10089 DATAD#,2,2
10090 DATAD,2,6
10091 DATAD,2,3
10092 DATAD,2,1
10093 DATAD,2,2
10094 DATAD,2,2
10095 DATAF,2,1
10096 DATAF,2,1
10097 DATAF,2,1
10098 DATAF,2,1
10099 DATAF,2,1
```

```
10101 DATAF,2,1
10102 DATAF,2,1
10103 DATAF,2,1
10104 DATAF,2,1
10105 DATAF,2,1
10106 DATAF,2,1
10107 DATAG,2,5
10108 DATAG,2,3
10109 DATAG,2,1
10110 DATAG,2,2
10111 DATAG,2,6
10112 DATAG,2,6
10113 DATAG,2,6
10114 DATAG,2,3
10115 DATAG,2,1
10116 DATAG,2,2
10117 DATAG,2,6
10118 DATAG,3,2
10119 DATAF,3,2
10120 DATAD,3,2
10121 DATAG,2,6
10122 DATAG,2,3
10123 DATAG,2,1
10124 DATAG,2,2
10125 DATAG,2,12
10126 DATAG,2,6
10127 DATAG,2,3
10128 DATAG,2,1
10129 DATAG,2,2
10130 DATAG,2,10
10131 DATAD,2,2
10132 DATAG,2,6
10133 DATAG,2,3
10134 DATAG,2,1
10135 DATAG,2,2
10136 DATAG,2,12
10137 DATAG,2,6
10138 DATAG,2,3
10139 DATAG,2,1
10140 DATAG,2,2
10141 DATAG,2,12
10142 DATAC,3,6
10143 DATAC,3,3
10144 DATAC,3,1
10145 DATAC,3,2
10146 DATAC,3,12
10147 DATAC,3,6
10148 DATAC,3,3
10149 DATAC,3,1
10150 DATAC,3,2
10151 DATAC,4,2
10152 DATAA#,3,2
10153 DATAG,3,2
10154 DATAC,3,2
10155 DATAA#,2,2
10156 DATAG,2,6
10157 DATAG,2,3
10158 DATAG,2,1
10159 DATAG,2,2
10160 DATAG,2,14
10161 DATAG,2,6
10162 DATAG,2,3
10163 DATAG,2,1
10164 DATAG,2,2
10165 DATAG,2,12
10166 DATAD#,2,9
10167 DATAD,2,1
10168 DATAD#,2,1
10169 DATAD,2,1
10170 DATAD#,2,8
10171 DATAD#,2,2
10172 DATAD#,2,2
10173 DATAD,2,10
10174 DATAD,2,2
10175 DATAD,2,2
10176 DATAA,2,1
10177 DATAC,3,1
10178 DATAD,3,2
```

```
10179 DATAD,3,2
10180 DATAA,2,2
10181 DATAD,3,2
10182 DATAG,3,10
10183 DATAG,3,2
10184 DATAF#,3,10
10185 DATAF#,3,2
10186 DATAF,3,12
10187 DATAE,3,12
10188 DATAD#,3,12
10189 DATAD,3,12
10190 DATAD#,3,12
10191 DATAD,3,8
10192 DATAD,4,4
10193 DATAG,2,9
10194 DATAG,2,1
10195 DATAG,2,2
10196 DATAG,2,12
10197 DATAG,2,9
10198 DATAG,2,1
10199 DATAG,2,2
10200 DATAG,2,12
10201 DATAG,2,9
10202 DATAG,2,1
10203 DATAG,2,2
10204 DATAG,2,12
10205 DATAG,2,9
10206 DATAG,2,1
10207 DATAG,2,2
10208 DATAG,2,12
10209 DATAC,3,9
10210 DATAC,3,1
10211 DATAC,3,2
10212 DATAC,3,9
10213 DATAC,3,1
10214 DATAC,3,2
10215 DATAC,3,9
10216 DATAC,3,1
10217 DATAC,3,2
10218 DATAC,3,12
10219 DATAG,2,9
10220 DATAG,2,1
10221 DATAG,2,2
10222 DATAG,2,12
10223 DATAG,2,9
10224 DATAG,2,1
10225 DATAG,2,2
10226 DATAG,2,6
10227 DATAG,2,1
10228 DATAA,2,1
10229 DATAA#,2,1
10230 DATAC,3,1
10231 DATAC#,3,1
10232 DATAD,3,1
10233 DATAD#,3,10
10234 DATAD#,3,2
10235 DATAD,3,9
10236 DATAD,3,1
10237 DATAD,3,2
10238 DATAD#,3,8
10239 DATAD#,3,2
10240 DATAD#,3,2
10241 DATAD,3,8
10242 DATAD,3,2
10243 DATAD,3,2
10244 DATAG,2,10
10245 DATAG,2,2
10246 DATAC,3,10
10247 DATAC,3,2
10248 DATAG,2,12
10249 DATAD,2,24
10250 DATAG,2,9
10251 DATAG,2,1
10252 DATAG,2,2
10253 DATAG,2,6
10254 DATAG,2,4
10255 DATAG,2,2
10256 DATAG,2,9
```

```
10257 DATAG,2,1
10258 DATAG,2,2
10259 DATAG,2,6
10260 DATAG,2,4
10261 DATAG,2,2
10262 DATAG,2,9
10263 DATAG,2,1
10264 DATAG,2,2
10265 DATAG,2,6
10266 DATAG,2,4
10267 DATAG,2,2
10268 DATAG,2,9
10269 DATAG,2,1
10270 DATAG,2,2
10271 DATAG,2,8
10272 DATAG,2,2
10273 DATAA#,2,2
10274 DATAC,3,9
10275 DATAC,3,1
10276 DATAC,3,2
10277 DATAC,3,10
10278 DATAC,3,2
10279 DATAC,3,9
10280 DATAC,3,1
10281 DATAC,3,2
10282 DATAC,3,10
10283 DATAC,3,2
10284 DATAG,2,9
10285 DATAG,2,1
10286 DATAG,2,2
10287 DATAG,2,10
10288 DATAG,2,2
10289 DATAG,2,9
10290 DATAG,2,1
10291 DATAG,2,2
10292 DATAG,2,6
10293 DATAG,2,1
10294 DATAA,2,1
10295 DATAA#,2,1
10296 DATAC,3,1
10297 DATAC#,3,1
10298 DATAD,3,1
10299 DATAD#,3,8
10300 DATAA#,2,2
10301 DATAD#,3,2
10302 DATAD,3,6
10303 DATAD,3,1
10304 DATAD,3,2
10305 DATAD,3,1
10306 DATAD,3,2
10307 DATAD#,3,8
10308 DATAA#,2,2
10309 DATAD#,3,2
10310 DATAD,3,6
10311 DATAD,3,6
10312 DATAG,2,8
10313 DATAD,2,2
10314 DATAG,2,2
10315 DATAC,2,8
10316 DATAC,2,2
10317 DATAC,2,2
10318 DATAG,2,8
10319 DATAG,2,2
10320 DATAG,2,2
10321 DATAD,2,24
10322 DATAG,2,24
19999 DATA,-1,-1
20000 REM 2. STIMME
20001 DATAA#,3,32
20002 DATAA#,3,64
20003 DATAC,3,32
20004 DATAA#,3,64
20005 DATAA#,3,128
20006 DATAA#,3,64
20007 DATAA#,3,64
20008 DATAA#,3,96
20009 DATAA#,3,64
20010 DATAA#,3,160
```

```
20011 DATAA#,3,96
20012 DATAA#,3,48
20013 DATAF,3,48
20014 DATAD#,3,64
20015 DATAA#,3,32
20016 DATAA#,3,48
20017 DATAF,3,48
20018 DATAD#,3,32
20019 DATAF,3,32
20020 DATAA#,3,70
20021 DATAA#,3,32
20022 DATAA,3,32
20023 DATAA#,3,32
20024 DATAA,3,32
20025 DATAA#,3,16
20026 DATAA,3,16
20027 DATAA#,3,1
20028 DATAA#,3,1
20029 DATAA#,3,1
20030 DATAA#,3,1
20031 DATAA#,3,1
20032 DATAA#,3,1
20033 DATAA#,3,1
20034 DATAA#,3,1
20035 DATAA#,3,1
20036 DATAA#,3,1
20037 DATAA#,3,1
20038 DATAA#,3,1
20039 DATAA#,3,1
20040 DATAA#,3,1
20041 DATAA#,3,1
20042 DATAA#,3,1
20043 DATAA#,3,1
20044 DATAA#,3,1
20045 DATAA#,3,2
20046 DATAA#,3,2
20047 DATAA#,3,2
20048 DATAC,3,6
20049 DATAE,5,12
20050 DATAA#,2,2
20051 DATAF,3,2
20052 DATAG,2,2
20053 DATAC,3,6
20054 DATAE,5,8
20055 DATAA,3,1
20056 DATAA,3,1
20057 DATAA,3,1
20058 DATAA,3,1
20059 DATAA,3,1
20061 DATAA,3,1
20062 DATAA,3,1
20063 DATAA,3,1
20064 DATAA,3,1
20065 DATAA,3,1
20066 DATAA,3,1
20067 DATAA#,3,5
20068 DATAA#,3,18
20069 DATAA#,3,24
20070 DATAG,3,6
20071 DATAG,3,16
20072 DATAG,3,2
20073 DATAF#,3,6
20074 DATAF#,3,8
20075 DATAA,3,1
20076 DATAA,3,1
20077 DATAA,3,1
20078 DATAA,3,1
20079 DATAA,3,1
20081 DATAA,3,1
20082 DATAA,3,1
20083 DATAA,3,1
20084 DATAA,3,1
20085 DATAA,3,1
20086 DATAA,3,1
20087 DATAA#,3,5
20088 DATAA#,3,18
20089 DATAA#,3,24
20090 DATAA#,3,18
```

```
20091 DATAA#,3,2
20092 DATAF,4,2
20093 DATAG,3,2
20094 DATAE,4,24
20095 DATAA#,3,18
20096 DATAA#,3,2
20097 DATAF,4,2
20098 DATAG,3,2
20099 DATAE,4,18
20100 DATAA#,3,2
20101 DATAF,4,2
20102 DATAG,3,2
20103 DATAE,4,18
20104 DATAA#,3,2
20105 DATAF,4,2
20106 DATAG,3,2
20107 DATAE,4,24
20108 DATAA#,3,18
20109 DATAA#,3,2
20110 DATAF,4,2
20111 DATAG,3,2
20112 DATAE,4,18
20113 DATAA#,3,2
20114 DATAF,4,2
20115 DATAG,3,2
20116 DATAG,4,22
20117 DATAG,4,2
20118 DATAF#,4,24
20119 DATAA#,4,12
20120 DATAA#,4,12
20121 DATAA#,4,12
20122 DATAD,4,12
20123 DATAG,4,12
20124 DATAF#,4,12
20125 DATAA,4,12
20126 DATAF#,4,12
20127 DATAA#,4,20
20128 DATAC,5,4
20129 DATAA#,4,20
20130 DATAC,5,4
20131 DATAA#,4,20
20132 DATAC,5,4
20133 DATAA#,4,20
20134 DATAA#,3,2
20135 DATAD,4,2
20136 DATAD#,4,20
20137 DATAF,4,4
20138 DATAD#,4,8
20139 DATAF,4,4
20140 DATAD#,4,12
20141 DATAA#,4,12
20142 DATAD,4,18
20143 DATAA#,3,18
20144 DATAG,4,12
20145 DATAF#,4,12
20146 DATAG,4,12
20147 DATAF#,4,12
20148 DATAA#,4,12
20149 DATAD#,4,12
20150 DATAA#,4,12
20151 DATAF#,4,24
20152 DATAA#,3,18
20153 DATAC,4,6
20154 DATAA#,3,12
20155 DATAC,4,12
20156 DATAA#,3,18
20157 DATAC,4,6
20158 DATAA#,3,24
20159 DATAD#,4,18
20160 DATAF,4,6
20161 DATAD#,4,24
20162 DATAA#,3,48
20163 DATAG,4,12
20164 DATAF#,4,12
20165 DATAG,4,12
20166 DATAF#,4,12
20167 DATAA#,3,12
20168 DATAD#,3,12
```

```
20169 DATAA#,3,12
20170 DATAF#,3,24
20171 DATAA#,3,24
29999 DATA,-1,-1
30000 REM 3. STIMME
30001 DATAG,4,32
30002 DATAG,4,40
30003 DATAA#,4,4
30004 DATAA,4,4
30005 DATAG,4,8
30006 DATAD,4,8
30007 DATAC,4,24
30008 DATAA#,3,4
30009 DATAF,3,4
30010 DATAG,3,16
30011 DATAA#,6,1
30012 DATAA,6,1
30013 DATAA#,6,1
30014 DATAA,6,1
30015 DATAG,6,1
30016 DATAF,6,1
30017 DATAD,6,1
30018 DATAC,6,1
30019 DATAA#,5,1
30020 DATAC,6,1
30021 DATAA#,5,1
30022 DATAC,6,1
30023 DATAA#,5,1
30024 DATAA,5,1
30025 DATAG,5,1
30026 DATAF,5,1
30027 DATAE,5,1
30028 DATAD,5,1
30029 DATAC,5,1
30030 DATAA#,4,1
30031 DATAA,4,1
30032 DATAF,4,1
30033 DATAA#,4,1
30034 DATAC,5,1
30035 DATAG,4,24
30036 DATAA#,4,27
30037 DATAA,4,5
30038 DATAA#,4,27
30039 DATAA,4,5
30040 DATAA#,4,32
30041 DATAC,5,26
30042 DATAA#,4,3
30043 DATAC,5,3
30044 DATAD,5,64
30045 DATAD,5,32
30046 DATAC,5,26
30047 DATAF,5,3
30048 DATAC,5,3
30049 DATAD,5,32
30050 DATAC,5,26
30051 DATAF,5,3
30052 DATAC,5,3
30053 DATAD,5,26
30054 DATAA#,4,3
30055 DATAA,4,3
30056 DATAG,4,16
30057 DATAA#,6,1
30058 DATAA,6,1
30059 DATAA#,6,1
30060 DATAA,6,1
30061 DATAG,6,1
30062 DATAF,6,1
30063 DATAD,6,1
30064 DATAC,6,1
30065 DATAA#,5,1
30066 DATAC,6,1
30067 DATAA#,5,1
30068 DATAC,6,1
30069 DATAA#,5,1
30070 DATAA,5,1
30071 DATAG,5,1
30072 DATAF,5,1
30073 DATAE,5,1
```

```
30074 DATAD,5,1
30075 DATAC,5,1
30076 DATAA#,4,1
30077 DATAA,4,1
30078 DATAF,4,1
30079 DATAA#,4,1
30080 DATAC,5,1
30081 DATAG,4,24
30082 DATAD,5,32
30083 DATAC,5,24
30084 DATAD,5,8
30085 DATAA#,4,3
30086 DATAA,4,3
30087 DATAG,4,20
30088 DATAC,5,3
30089 DATAF,4,3
30090 DATAD,4,44
30091 DATAA#,4,3
30092 DATAA,4,3
30093 DATAG,4,3
30094 DATAD,4,8
30095 DATAA#,3,3
30096 DATAG,4,64
30097 DATAG,5,12
30098 DATAF,5,4
30099 DATAG,5,16
30100 DATAG,4,2
30101 DATAA#,4,2
30102 DATAC,5,2
30103 DATAD,5,6
30104 DATAC,5,4
30105 DATAD,5,26
30106 DATAD,5,1
30107 DATAC,5,1
30108 DATAA#,4,2
30109 DATAC,5,2
30110 DATAD,5,6
30111 DATAC,5,2
30112 DATAD,5,12
30113 DATAA,4,1
30114 DATAA#,4,1
30115 DATAA,4,1
30116 DATAG,4,1
30117 DATAA,4,2
30118 DATAF,4,2
30119 DATAF,4,2
30120 DATAD,4,2
30121 DATAF,4,12
30122 DATAG,4,2
30123 DATAA,4,2
30124 DATAA#,4,10
30125 DATAA,4,1
30126 DATAG,4,1
30127 DATAF,4,2
30128 DATAG,4,2
30129 DATAA#,4,16
30130 DATAA,4,12
30131 DATAF,4,2
30132 DATAG,4,2
30133 DATAG,4,12
30134 DATAD,5,2
30135 DATAF,5,2
30136 DATAG,5,10
30137 DATAG,5,2
30138 DATAA#,5,2
30139 DATAC,6,2
30140 DATAA#,5,10
30141 DATAD,4,2
30142 DATAF,4,4
30143 DATAG,4,8
30144 DATAA,4,2
30145 DATAA#,4,2
30146 DATAA#,4,8
30147 DATAF,4,4
30148 DATAG,4,18
30149 DATAD,5,1
30150 DATAC,5,1
30151 DATAA#,4,2
```

```
30152 DATAC,5,2
30153 DATAD,5,12
30154 DATAD,5,2
30155 DATAF,5,6
30156 DATAG,5,2
30157 DATAG,5,2
30158 DATAG,5,8
30159 DATAG,5,1
30160 DATAF,5,1
30161 DATAD,5,1
30162 DATAD,5,9
30163 DATAD,5,1
30164 DATAC,5,1
30165 DATAA#,4,1
30166 DATAC,5,1
30167 DATAC,5,12
30168 DATAG,4,2
30169 DATAA#,4,10
30170 DATAD,5,2
30171 DATAC,5,2
30172 DATAA#,4,2
30173 DATAC,5,2
30174 DATAD,5,12
30175 DATAD,5,2
30176 DATAF,5,2
30177 DATAD,5,12
30178 DATAG,5,2
30179 DATAG,5,2
30180 DATAA#,5,12
30181 DATAD,5,2
30182 DATAA#,5,2
30183 DATAD,6,8
30184 DATAC,6,2
30185 DATAA#,5,2
30186 DATAA,5,2
30187 DATAF,5,2
30188 DATAG,5,32
30189 DATAA#,3,2
30190 DATAF,4,2
30191 DATAG,3,2
30192 DATAE,4,58
30193 DATAA#,3,2
30194 DATAF,4,2
30195 DATAG,3,2
30196 DATAE,4,58
30197 DATAA#,3,2
30198 DATAF,4,2
30199 DATAG,3,2
30200 DATAE,4,26
30201 DATAA#,3,2
30202 DATAF,4,2
30203 DATAG,3,2
30204 DATAE,4,18
30205 DATAA#,3,2
30206 DATAF,4,2
30207 DATAG,3,2
30208 DATAE,4,18
30209 DATAA#,3,2
30210 DATAF,4,2
30211 DATAG,3,2
30212 DATAE,4,14
30213 DATAC,4,1
30214 DATAC,4,1
30215 DATAC,4,1
30216 DATAC,4,1
30217 DATAC,4,1
30218 DATAC,4,1
30219 DATAC,4,1
30221 DATAC,4,1
30222 DATAC,4,1
30223 DATAC,4,1
30224 DATAC,4,1
30225 DATAD,4,5
30226 DATAF,5,12
30227 DATAA#,3,2
30228 DATAF,4,2
30229 DATAG,3,2
30230 DATAE,4,18
```

```
30231 DATAA#,3,2
30232 DATAF,4,2
30233 DATAG,3,2
30234 DATAD#,4,6
30235 DATAD#,4,16
30236 DATAD#,4,2
30237 DATAD,4,6
30238 DATAD,4,8
30239 DATAC,4,1
30241 DATAC,4,1
30242 DATAC,4,1
30243 DATAC,4,1
30244 DATAC,4,1
30245 DATAC,4,1
30246 DATAC,4,1
30247 DATAC,4,1
30248 DATAC,4,1
30249 DATAC,4,1
30250 DATAC,4,1
30251 DATAD,4,5
30252 DATAF,5,12
30253 DATAA#,3,2
30254 DATAF,4,2
30255 DATAG,3,2
30256 DATAE,4,6
30257 DATAE,4,18
30258 DATAC,6,1
30259 DATAD,6,1
30260 DATAG,5,13
30261 DATAD,5,1
30262 DATAG,5,1
30263 DATAF,5,1
30264 DATAD,5,4
30265 DATAC,5,1
30266 DATAA#,4,1
30267 DATAC,5,1
30268 DATAA#,4,1
30269 DATAG,4,19
30270 DATAA#,6,1
30271 DATAG,6,1
30272 DATAD,6,1
30273 DATAA#,6,10
30274 DATAA#,6,2
30275 DATAA,6,10
30276 DATAF,6,1
30277 DATAD,6,1
30278 DATAE,6,9
30279 DATAG,6,1
30280 DATAF,6,1
30281 DATAD,6,1
30282 DATAF,6,6
30283 DATAA#,6,1
30284 DATAG,6,1
30285 DATAD,6,1
30286 DATAA#,6,1
30287 DATAG,6,1
30288 DATAD,6,1
30289 DATAC,6,9
30290 DATAC,6,1
30291 DATAA#,6,1
30292 DATAG,6,1
30293 DATAA#,6,1
30294 DATAG,6,6
30295 DATAA#,5,1
30296 DATAE,6,2
30297 DATAD,6,2
30298 DATAG,5,1
30299 DATAA#,5,1
30300 DATAG,5,31
30301 DATAD,5,1
30302 DATAA#,5,1
30303 DATAF,5,1
30304 DATAG,5,9
30305 DATAA#,5,1
30306 DATAF,6,1
30307 DATAG,5,1
30308 DATAE,6,9
30309 DATAA,5,1
```

```
30310 DATAG,5,1
30311 DATAF,5,1
30312 DATAG,5,6
30313 DATAA#,5,2
30314 DATAD,6,1
30315 DATAC,6,1
30316 DATAD,6,2
30317 DATAD#,6,12
30318 DATAD#,6,8
30319 DATAD#,6,4
30320 DATAD,6,7
30321 DATAF#,5,1
30322 DATAG, 5,1
30323 DATAF#,5,1
30324 DATAE,5,1
30325 DATAF#,5,1
30326 DATAF#,5,10
30327 DATAG,6,1
30328 DATAG,6,1
30329 DATAG,6,8
30330 DATAG,6,2
30331 DATAG,6,2
30332 DATAA,6,2
30333 DATAA#,6,2
30334 DATAA,6,2
30335 DATAG,6,3
30336 DATAG,6,1
30337 DATAF,6,1
30338 DATAD,6,1
30339 DATAF,6,1
30340 DATAD,6,8
30341 DATAD,6,1
30342 DATAC,6,1
30343 DATAA#,5,1
30344 DATAG,5,7
30345 DATAG,5,1
30346 DATAA#,5,2
30347 DATAC,6,2
30348 DATAD#,6,7
30349 DATAD#,6,1
30350 DATAD,6,1
30351 DATAC,6,1
30352 DATAA#,5,1
30353 DATAC,6,1
30354 DATAD,6,8
30355 DATAD,6,2
30356 DATAF,6,2
30357 DATAD#,6,8
30358 DATAD,6,1
30359 DATAC,6,1
30360 DATAA#,5,1
30361 DATAC,6,1
30362 DATAD,6,12
30363 DATAG,5,20
30364 DATAF,5,2
30365 DATAA,5,2
30366 DATAG,5,23
30367 DATAA,5,1
30368 DATAA#,5,18
30369 DATAC,6,2
30370 DATAA,5,1
30371 DATAA#,5,1
30372 DATAC,6,1
30373 DATAA#,5,25
30374 DATAD#,6,9
30375 DATAD,6,1
30376 DATAC,6,2
30377 DATAG,5,8
30378 DATAA,5,1
30379 DATAA#,5,3
30380 DATAD#,6,8
30381 DATAD,6,1
30382 DATAC,6,3
30383 DATAD#,6,8
30384 DATAF,6,1
30385 DATAD#,6,1
30386 DATAD,6,1
30387 DATAC,6,1
```

```
30388 DATAD,6,10
30389 DATAA,5,1
30390 DATAA#,5,1
30391 DATAC,6,10
30392 DATAF,5,1
30393 DATAG,5,1
30394 DATAA#,5,1
30395 DATAA,5,1
30396 DATAG,5,22
30397 DATAA#,5,10
30398 DATAA,5,1
30399 DATAG,5,1
30400 DATAF#,5,12
30401 DATAA#,5,10
30402 DATAA,5,1
30403 DATAA#,5,1
30404 DATAC,6,12
30405 DATAD,6,8
30406 DATAA#,5,1
30407 DATAA,5,1
30408 DATAG,5,1
30409 DATAA,5,1
30410 DATAD#,5,10
30411 DATAF,5,2
30412 DATAG,5,12
30413 DATAF,5,21
30414 DATAD,6,1
30415 DATAG,6,2
30416 DATAG,6,4
30417 DATAG,6,1
30418 DATAA#,6,1
30419 DATAG,6,9
30420 DATAD,5,1
30421 DATAG,5,1
30422 DATAF,5,1
30423 DATAD,5,3
30424 DATAD,5,1
30425 DATAC,5,1
30426 DATAA#,4,1
30427 DATAC,5,1
30428 DATAA#,4,1
30429 DATAG,4,10
30430 DATAA#,5,1
30431 DATAC,6,1
30432 DATAD,6,8
30433 DATAG,5,1
30434 DATAA#,5,1
30435 DATAG,5,3
30436 DATAA#,5,7
30437 DATAG,6,1
30438 DATAA#,6,1
30439 DATAG,6,21
30440 DATAD,5,1
30441 DATAA#,5,1
30442 DATAA,5,1
30443 DATAG,5,6
30444 DATAA#,5,2
30445 DATAD,6,5
30446 DATAG,5,1
30447 DATAA#,5,1
30448 DATAD,6,1
30449 DATAC,6,1
30450 DATAA#,5,1
30451 DATAC,6,4
30452 DATAC,6,1
30453 DATAA#,5,1
30454 DATAC,6,1
30455 DATAC,6,1
30456 DATAC,6,1
30457 DATAA#,5,1
30458 DATAD,6,10
30459 DATAG,5,1
30460 DATAF,5,1
30461 DATAD,5,1
30462 DATAC,5,1
30463 DATAC,5,4
30464 DATAA#,4,1
30465 DATAG,4,1
```

```
30466 DATAA#,4,6
30467 DATAC,5,1
30468 DATAG,4,1
30469 DATAA#,4,1
30470 DATAA,4,1
30471 DATAF,4,1
30472 DATAC,4,1
30473 DATAD,4,9
30474 DATAD,4,1
30475 DATAA#,4,1
30476 DATAF,4,1
30477 DATAG,4,10
30478 DATAA#,5,1
30479 DATAC,6,1
30480 DATAD,6,10
30481 DATAG,6,1
30482 DATAG,6,1
30483 DATAG,6,3
30484 DATAG,4,5
30485 DATAG,6,1
30486 DATAG,6,1
30487 DATAA#,6,1
30488 DATAG,6,1
30489 DATAC,6,3
30490 DATAG,4,6
30491 DATAC,6,1
30492 DATAA#,5,1
30493 DATAG,5,1
30494 DATAA#,5,1
30495 DATAG,5,5
30496 DATAD,5,1
30497 DATAG,5,1
30498 DATAA#,5,1
30499 DATAC,6,1
30500 DATAD,6,2
30501 DATAD#,6,8
30502 DATAD#,6,4
30503 DATAF#,5,6
30504 DATAG,5,1
30505 DATAF#,5,1
30506 DATAE,5,1
30507 DATAF#,5,1
30508 DATAG,5,1
30509 DATAF#,5,1
30510 DATAG,5,9
30511 DATAD,5,1
30512 DATAG,5,1
30513 DATAF,5,1
30514 DATAD,5,9
30515 DATAD,5,1
30516 DATAC,5,1
30517 DATAA#,4,1
30518 DATAC,5,1
30519 DATAA#,4,1
30520 DATAG,4,2
30521 DATAA#,4,2
30522 DATAG,4,2
30523 DATAD,4,1
30524 DATAC,4,1
30525 DATAA#,3,1
30526 DATAC,4,1
30527 DATAF,4,24
30528 DATAG,5,24
39999 DATA,-1,-1

READY.
```