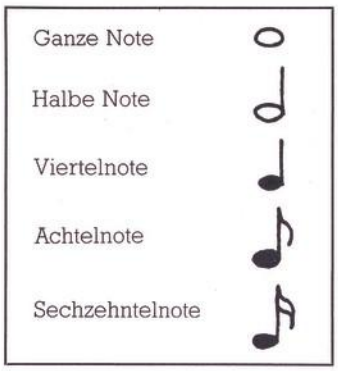
Dem Klang auf der Spur (Teil 9)

In diesem Teil wird gezeigt, wie man dreistimmige Musikstücke programmgesteuert, schnell und zeitexakt auf dem C 64 wiedergeben kann. So ganz nebenbei erfahren Sie eine Menge über die Interrupttechnik.

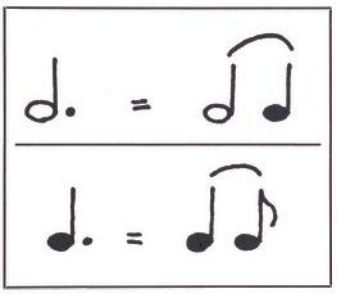
Zunächst einige Grundlagen zum Sequenzer. Unter einem Sequenzer versteht man ein Gerät oder Programm, das einen Synthesizer mit einer vorprogrammierten Tonfolge ansteuert. Die zentrale Rolle spielt dabei das genaue Einhalten eines programmierbaren Zeitmaßes.

Musikstücke werden üblicherweise in Takte von etwa 1 bis 4 Sekunden Länge eingeteilt. Am gebräuchlichsten ist der 4/4-Takt, der die Länge einer ganzen Note hat. Andere gebräuchliche Taktarten sind 2/4, 3/4, 6/8, 4/4, 5/4, 3/2. Diese Angaben betreffen allerdings nur die Zählweise der Takte und nicht das Tempo eines Musikstücks. So sind zum Beispiel 3/4- und 6/8-Takt bis auf die Zählweise vollkommen identisch.

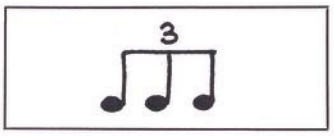
Die Notenlängen werden in Bruchteilen der ganzen Note angegeben:



Es kommen auch ungeradzahlige Vielfache dieser Notenlängen vor. Durch Punktierung kennzeichnet man die Verlängerung einer Note um die Hälfte ihrer ursprünglichen Länge:



Alle diese Notenlängen passen in ein Raster, welches eine ganze Note in 16 oder 32 gleiche Zeitabschnitte teilt. Es werden aber häufig auch sogenannte Triolen (Drittelnoten) eingesetzt. Zum Beispiel Achteltriolen,



das sind drei gleichlange Noten mit der Länge einer Viertelnote. Aus diesem Grund sollte das Zeitraster (die Anzahl der Zeitabschnitte, in die der Sequenzer eine ganze Note einteilt) auch den Faktor 3 enthalten. Ein sinnvolles Zeitraster ist zum Beispiel 96 (= 3x32).

Das Tempo wird in der Musik in Schlägen pro Minute (beats per minute: bpm) gemessen. Ein Schlag entspricht dabei einer Viertelnote. Der sinnvolle Bereich für dieses Maß liegt bei etwa 40 bis 240 bpm. Beim schnellen Tempo 240 bpm dauert eine ganze Note genau eine Sekunde. Der Sequenzer muß dann 96 Schritte pro Sekunde ausführen.

Programmtechnik

Ein Sequenzer ist von der zu erbringenden Funktion her eigentlich ein sehr einfaches Programm. Seine Leistungen sind schnell aufgezählt:

— Tonhöhen steuern
— Triggerung der einzelnen Stimmen (GATE ON und GATE OFF)

Diese Steuerungen müssen zeitgenau und unabhängig voneinander für drei Stimmen erfolgen. Darüber hinaus wären einige Zusatzfunktionen sinnvoll:

— programmierbare Tempoänderungen
— programmierbare Soundwechsel
— programmierbare Änderung der Inhalte beliebiger Speicherplätze (Parameter-Änderung)

Es soll zunächst ein einfacher Basis-Sequenzer entwickelt werden, der sich dann leicht um die genannten Zusatzfunktionen erweitern läßt. Die Erweiterungen sollen über Vektoren, also ohne Änderung des Grundprogramms, an dieses angeschlossen werden können.

Ein Sequenzer ist, ähnlich wie der in dieser Reihe veröffentlichte Modulator, ein Programm, das in regelmäßigen Zeitabständen eine Leistung erbringen muß. Der Aufruf per Interrupt, ausgelöst durch einen Zeitgeber, bietet sich also auch hier an. Jeder CIA (Complex Interface Adapter) ist mit zwei 16-Bit-Timern ausgestattet, die sich für diese Aufgabe eignen. Timer A in CIA1 wird bereits für den Systeminterrupt eingesetzt. Ein Systeminterrupt findet konstant 60-mal pro Sekunde statt und kann mit dem Aufruf eines Modulatorschritts gekoppelt werden.

Musik per Interrupt

Die Aufruffrequenz der Sequenzer-Schritte soll dagegen im Bereich von zirka 20-100 Hz, abhängig vom Tempo des Musikstücks, variabel sein. (Man erinnere sich: 240 bpm entsprechen 96 Hz bei einem Zeitraster von 96 Schritten pro ganzer Note.) Das legt den Einsatz eines weiteren unabhängigen Timers nahe. In Frage kommen dafür:

Timer B in CIA1 (IRQ)
Timer A in CIA2 (NMI)
Timer B in CIA2 (NMI)

Die Auswahl des Timers ist willkürlich. Im vorliegenden Programm wird Timer B in CIA1 eingesetzt. Dadurch bleiben die Timer in CIA 2 noch vollkommen frei für Zwecke, die nichts mit der Musikprogrammierung zu tun haben müssen. Da nun Timer A und Timer B beide unabhängig voneinander Interrupts auslösen können, muß die angesprungene Interrupt-Serviceroutine die Interrupt-Quelle ermitteln, also feststellen, welcher Timer den Interrupt ausgelöst hat und abhängig davon weiterverzweigen. Zu diesem Zweck wird im sogenannten Interrupt-Control-Register (ICR) $DC0D bei einem Timer-A-Interrupt Bit 0 und bei einem Timer-B-Interrupt Bit 1 gesetzt.

Programmierung des CIA

Zur Steuerung von CIA-Interrupts dient das schon erwähnte Interrupt-Control-Register (ICR) $DC0D. Dieses Register hat zwei Funktionen, je nachdem, ob schreibend oder lesend darauf zugegriffen wird. Bei Lesezugriff zeigt es an, ob, und wenn ja, woher ein Interrupt ausgelöst wurde. Zugleich wird das Register gelöscht und die Interrupt-Anforderung zurückgenommen (Die IRQ-Leitung geht von low auf high). Die Bits 0-4 sind dabei verschiedenen Interruptquellen zugeordnet. Uns interessieren hier nur die Bits 0 und 1, welche zu den Timer-Interrupts gehören. Durch einen Schreibzugriff wird dagegen ein Masken-Register angesprochen. Damit lassen sich die Interruptquellen einzeln freigeben oder sperren. Die Bits 0-4 kann man einzeln setzen oder zurücksetzen. Ist im geschriebenen Byte Bit 7 gesetzt, wird jedes mit einer 1 beschriebene Bit gesetzt, während die anderen Bits unverändert bleiben. Ist Bit 7 rückgesetzt, so wird jedes mit einer 1 beschriebene Bit zurückgesetzt, während die anderen Bits wieder unverändert bleiben. Gesetzte Bits ermöglichen eine Interrupterzeugung durch die jeweilige Quelle. Die Freigabe der Interrupterzeugung durch Timer B sieht also so aus:

```
LDA #%10000010
STA $DD0D       ;ICR Bit 1 setzen
```

Der Timer selbst wird durch drei Register gesteuert.

Das Registerpaar TIMER B ($DC06/$DC07) liefert bei Lesezugriff den aktuellen 16-Bit-Zählerstand. Dieser Wert wird kontinuierlich heruntergezählt. Bei Erreichen von Null stoppt der Timer entweder (One-Shot-Mode) oder lädt einen Wert aus einem Timer-Latch (Latch = Zwischenspeicher) nach und zählt von neuem herunter (Continous Mode). Bei diesem Timer-Unterlauf wird ein Interrupt erzeugt, wenn Bit 1 im ICR gesetzt ist. Ein Schreibzugriff auf TIMERA bezieht sich dagegen auf das 16-Bit-Latch. Mit dem Latch-Wert kann man die Zeit zwischen zwei Interrupts im Bereich von 1 bis 65535 Mikrosekunden steuern.

Das Register CRB (Control Register B, $DC0F) steuert die Betriebsart des Timers (Start/Stop, One Shot/Continous, u.a.)

```
Durch LDA #%00010001
STA $DD0E
```

wird der Zählerstand mit dem Latch-Wert geladen und der Timer gestartet.

Die Interrupt-Service-Routine

Sie fragt zunächst ab, ob der Interrupt von Timer A (Systeminterrupt, Modulatorschritt) oder von Timer B (Sequenzerschritt) kommt. Bei einer möglichen gleichzeitigen Interruptanforderung durch beide Timer, wird der Timer-B-Interrupt bevorzugt behandelt. Das hat folgende Gründe:

— Für ein exaktes Sequenzer-Timing sollten anzuspielende Noten möglichst wenig verzögert werden.
— Die Abarbeitung eines Sequenzer-Schritts benötigt viel weniger Rechenzeit als ein Modulatorschritt (zeitaufwendige Multiplikationen) oder eine Systeminterrupt-Behandlung.
— Die Aufruffrequenz kann bei den Sequenzer-Schritten sehr hoch sein (96 Hz bei 240 bpm, aber auch über 200 Hz sind technisch leicht möglich).

Da das ICR beim Lesen gelöscht wird, muß sein Inhalt zwischengespeichert werden, damit beim Auftreten von zwei Interrupts die Behandlung des niedriger priorisierten Timer-A-Interrupts nachgeholt werden kann.

Bei Auftreten eines Interrupts wird immer das Interrupt-Bit im CPU-Statusregister gesetzt, damit die CPU nicht gleich wieder unterbrochen werden kann. Da die IRQ-Leitung so lange auf Low-Pegel bleibt, bis die CPU durch Auslesen des CIA-ICR die Interruptanforderung löscht, würde sich das System ohne gesetztes Interrupt-Bit durch einen Dauerinterrupt aufhängen. Es steht dem Programmierer allerdings frei, nach dem Auslesen des ICR das Interrupt-Bit durch den Befehl CLI (Clear Interrupt-Flag) zurückzusetzen, um damit das Programm wieder unterbrechbar zu machen. Beim vorliegenden Programm bleibt bei einem Sequenzer-Schritt das Interrupt-Bit gesetzt, während es zur Abarbeitung eines Timer-A-Interrupts rückgesetzt wird. Dadurch kann die CPU auch dann durch einen Timer-B-Interrupt unterbrochen werden.

Das Betriebssystem und das Programm Modulator machen beide intensiven Gebrauch von der Zero-Page. Die Inhalte der Zero-Page-Speicherplätze dürfen von einem interruptgetriebenen Programm nicht verändert werden. Das Sequenzer-Programm belegt daher nur zwei Zero-Page-Speicherplätze ($FE,$FF). Ihre Inhalte werden bei Programmbeginn zwischengespeichert und bei Programmende restauriert.

Die verwendeten Datenstrukturen

Um ein Musikstück in eine computergerechte Form zu bringen, muß man im wesentlichen die Tonhöhe und die Länge der einzelnen Noten codieren. Beim Einsatz mehrerer, verschieden klingender Stimmen, muß man außerdem jede Note eindeutig einer Stimme zuordnen. Die hier verwendete Datenstruktur (Bild 1) verfolgt mit ihrem etwas komplizierten Aufbau zwei Ziele:

— Sparsamer Umgang mit dem Speicher

— Gute Editiermöglichkeiten. (Ein Editorprogramm in Basic folgt in der nächsten Ausgabe)

Tracks

Die Steueranweisungen werden für die drei Stimmen getrennt in drei sogenannten Tracks (Tonspuren) gespeichert. Ein Track ist eine zusammenhängende Folge von 1-Byte-Kommandos. Das häufigste Kommando dürfte das Ton-Kommando sein. Die Tonhöhe wird aus einer Oktav-Nummer und einer Tonnummer (siehe Bild 2), die in den beiden Nibbles (= Halbbytes) eines Bytes stehen, ermittelt. Das Programm benötigt dazu lediglich eine Tabelle der Frequenzen der höchsten Oktave. Die Frequenzen der niedrigeren Oktaven werden durch Teilung durch Zweierpotenzen errechnet. Eine Division durch 2 wird durch einen einfachen Rechts-Shift realisiert. Die Dauer des Tones ist nicht Bestandteil des Ton-Kommandos. Sie wird durch das Zeit-Kommando voreingestellt. Da häufig mehrere Töne mit gleicher Länge aufeinanderfolgen, genügt ein einziges Zeit-Kommando (ein oder zwei Bytes), um die Tonlänge (siehe Bild 3) einzustellen. Dabei wird zwischen einer GATE-ON- und einer GATE-OFF-Phase unterschieden, deren Längen zusammengenommen die gewünschte Tonlänge ergeben.
Beispiel: GATE-ON-Zeit = 5
GATE-OFF-Zeit = 7
Gesamtzeit = 12

Das entspricht einer kurz angeschlagenen Achtelnote (bei 96 Zeitschritten pro ganzer Note). Die GATE-ON-Zeit ist im Bereich 1-96, die GATE-OFF-Zeit im Bereich 0-30 einstellbar. Der Sequenzer setzt nach Ablauf der GATE-ON-Zeit das GATE-Bit der entsprechenden Stimme im SID zurück und wartet dann die GATE-OFF-Zeit ab. Ist diese 0, so wird natürlich sofort der nächste Ton gespielt. Man kann aber auch explizit Pausen programmieren (Code $EF). Ihre Länge ist die Summe aus GATE-ON- und GATE-OFF-Zeit.

Der Code $00 ist zur Kennzeichnung für das Track-Ende vorgesehen. Die Codes $F8 bis $FF sind für Sonderfunktionen reserviert, die für eine spätere Erweiterung des Sequenzers gedacht sind. Angesprungen werden sie über eine Tabelle von Vektoren, die im Moment

0	c
1	cis = des
2	d
3	dis = es
4	e
5	f
6	fis = ges
7	g
8	gis = as
9	a
10	ais = b
11	h

Bild 2. Tonnummer und Note

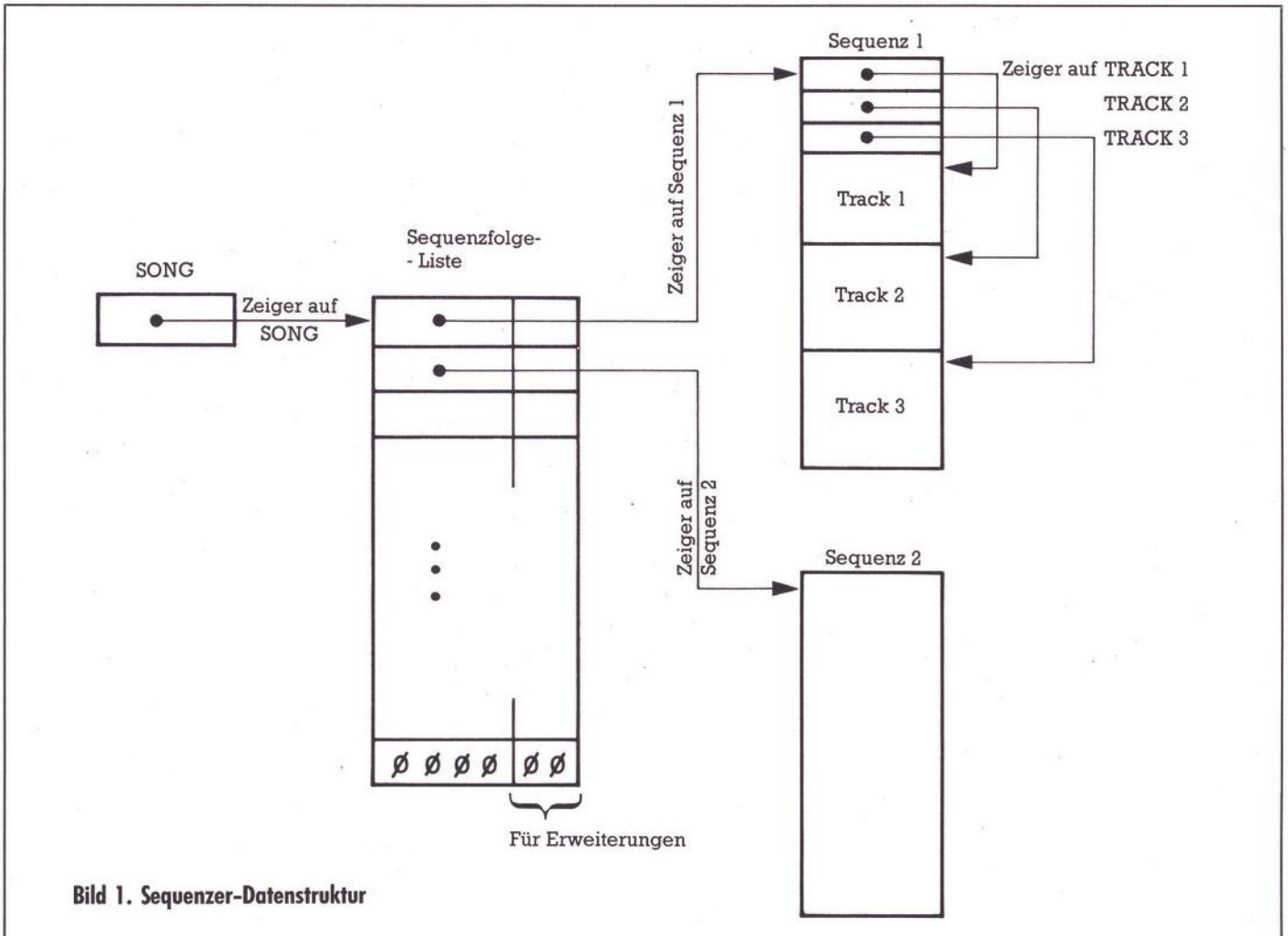


Bild 1. Sequenzer-Datenstruktur

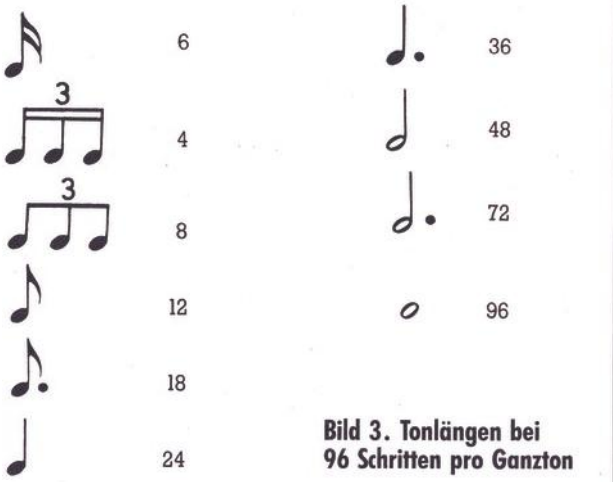


Bild 3. Tonlängen bei 96 Schritten pro Ganzton

nur in den Programmteil zur Ausführung des nächsten Kommandos führen, also nichts bewirken. Sinnvolle Sonderfunktionen sind:
— Änderung von Soundparametern
— Wahl eines ganzen Parametersatzes (Soundwechsel) im Zusammenhang mit dem Programm Modulator
— Tempowechsel
Diese Sonderfunktionen werden den Sequenzer in der nächsten Folge ergänzen.

Sequenzen

Für jede der drei Stimmen gibt es eine Folge von Kommandos, einen Track. Die drei Tracks werden zu einer Sequenz zusammengefaßt. Eine Sequenz ist hier ein zusammenhängender Abschnitt eines Musikstücks, der einen einzigen Ton, einen Takt oder auch das ganze Stück umfassen kann. Den drei Tracks gehen drei Zeiger auf die Track-Startadressen voran. Obwohl es sich aus Gründen der Übersichtlichkeit empfiehlt, die Sequenzen wie in Bild 1 zusammenhängend in der Folge Zeiger-Track 1, -Track 2, -Track 3 zu speichern, besteht dazu kein Zwang. Es müssen lediglich die drei Track-Zeiger einer Sequenz und die Tracks in sich zusammenhängen.

Sequenzfolge-Liste

Um eine Sequenz zu wiederholen, muß man sie nicht zweimal programmieren, sondern kann sie wie ein Unterprogramm mehrmals aufrufen. Die Sequenzfolge-Liste enthält dazu die Startadressen der Sequenzen in der Reihenfolge, in der diese gespielt werden sollen. Dabei können die gleichen Adressen natürlich mehrfach auftreten. Unter der Startadresse einer Sequenz wird hier die Adresse des Zeigers auf Track 1 verstanden. Die Sequenzfolge-Liste enthält für jede Sequenz außer dem Zeiger noch ein drittes Byte, das für spätere Erweiterungen vorgesehen ist. Drei Nullen schließen die Liste ab.

Flexibilität durch Steuerflags und Vektoren

Im Normalfall wird man die drei Tracks einer Sequenz gleich lang programmieren. Macht man dagegen die Tracks unterschiedlich lang, so wiederholt das Programm die kürzeren Tracks so lange, bis der längste Track zu Ende gespielt ist. Erst dann geht das Programm zur nächsten Sequenz über. Dieses Verhalten kann bei manchen Musikstücken nützlich sein. Das MSE-Listing 2 enthält einen Musik-Datensatz, bei dem in der zweiten Sequenz der dritte Track aus nur vier Tönen besteht, die fortlaufend wiederholt werden.

Normalerweise hält der Sequenzer an, wenn alle Sequenzen gemäß Sequenzfolge-Liste durchgespielt sind. Nach dem Anhalten wird auch der Interruptvektor auf seinen ursprünglichen Wert zurückgestellt. Eine 1 im Flag REPMODUS bewirkt, daß das ganze Stück endlos wiederholt wird.

Eine 1 im Flag SEQMODUS bewirkt, daß die aktuelle Sequenz endlos wiederholt wird. Auch hier ist der längste Track der Sequenz maßgeblich.

Eine 1 im Flag LEGATO bewirkt, daß die GATE-Bits in den SID-Steuerregister nicht zurückgesetzt werden. Dadurch klingen die Töne gebunden. Dazu muß allerdings ein Sustain-Pegel ungleich Null eingestellt sein, sonst ist überhaupt nichts hörbar.

An allen wichtigen Stellen des Sequenzers wird der Programmfluß über Vektoren weitergeleitet. Damit soll die Möglichkeit, das Programm nachträglich leicht zu erweitern, offengehalten werden. Die Vektoren für die acht Sonderfunktionen wurden schon erwähnt. Außer diesen acht gibt es noch drei weitere Vektoren:
TONVEKTOR
Er führt das Programm weiter, nachdem die Frequenz für einen Ton-an-Befehl ermittelt wurde. Im vorliegenden Programm wird der Frequenzwert direkt in den SID geschrieben. Bei einem Einsatz zusammen mit dem Modulator muß die Frequenz dagegen in ein Modulator-Register geschrieben werden.
EXTRAVEKTOR
Über diesen Vektor kann man weitere Aktionen an einen Sequenzer-Schritt anhängen. Denkbar wäre zum Beispiel die Anzeige der gespielten Noten auf dem Bildschirm in Realtime.
IRQAVEKTOR
Führt zum Systeminterrupt $EA31. Dieser Vektor muß beim Einsatz mit dem Modulator auf die Startadresse des Modulatorschrittes zeigen.

Das vorliegende Sequenzerprogramm (Listing 1) belegt den Speicherbereich $C480-$C778. $C480 = 50304 ist gleichzeitig auch die Startadresse (SYS 50304). Tabelle 1 faßt die wichtigsten Routinen, Variablen und Vektoren des Sequenzer-Programms zusammen.

Das Programmieren von Musikstücken mit Hilfe der Tabelle 2 ist noch etwas mühsam. Ein Editor in der nächsten Ausgabe wird diese Arbeit erleichtern. Mit dem Datensatz aus Listing 2 (»Kobold« aus den »Lyrischen Stücken« von Edvard Grieg) kann man den Sequenzer testen.

Die Verschmelzung des Sequenzers und des Modulators zu einer funktionellen Einheit wird in der nächsten Folge behandelt. (Thomas Krätzig/tr)

Routinen, Variablen, Vektoren

$C480	JMP TEST	Teststart, Zeiger initialisieren, die wichtigsten SID-Parameter setzen, Sequenzer starten.
$C49E	SIDCR (3 Byte)	SID-Control-Register-Bytes mit zurückgesetztem GATE-Bit
$C4A8	LEGATO (1 Byte)	Flag 0 = normaler Betrieb 1 = kein GATE-OFF
$C4A9	SEQMODUS (1 Byte)	Flag 0 = ganzes Stück spielen 1 = Sequenz wiederholen
$C4AA	REPMODUS (1 Byte)	Flag 0 = Stück einmal spielen 1 = Stück immer wieder spielen
$C4C3	FUNCTION (8*2 Byte)	Vektoren für Sonderfunktionen
$C4D3	TONVEKTOR (2 Byte)	Vektor zur Weiterverarbeitung der Frequenz bei Ton an
$C4D5	EXTRAVEKTOR (2 Byte)	Vektor für Zusatzaktion bei jedem Sequenzer-Schritt
$C4D7	IRQAVEKTOR (2 Byte)	Vektor für Timer-A-Interrupt
$C4D9	START	Sequenzer starten (Es werden keine Zeiger initialisiert)
$C51E	IRQSERVICE	Anlaufpunkt für alle IRQ-Interrupts
$C67D	NEXTAKT	Dorthin sollten alle Sonderfunktionen zurückspringen
$C716	STOP	Sequenzer unterbrechen/ausschalten. Er kann mit START jederzeit wieder gestartet werden.
$C739	TEST	siehe $C480

Tabelle 1. Die wichtigsten Routinen, Varianten und Vektoren des Sequenzers.

Kommandoformate innerhalb der Tracks

Kommando	Interpretation	
%0000 0000	Track-Ende	
%0ttt tttt	Zeitvorgabe	t = 1...96 GATE-ON-Zeit: = t-m t = 97...127 GATE-OFF-Zeit: = t-97
%1mmm nnnn	nächster Ton	m = 0...6 Oktave n = 0...11 Tonnummer n = 12...15 Pause
%1110 1111	Pause	(Standardcode $EF)
%1111 1fff	Sonderfunktion	f = 0...7 Funktionsnummer

Tabelle 2. So programmiert man einen »Track«.

```
programm : sequencer          c480 c779
-----------------------------------------

c480 : 4c 39 c7 ff 00 00 00 c8  ec
c488 : 0c c8 e0 c8 fd c8 20 c9  84
c490 : 43 c9 18 18 18 00 00 00  43
c498 : 0d 0d 0d 00 00 00 20 20  30
c4a0 : 20 01 02 04 00 00 00 00  42
c4a8 : 00 00 01 1e 86 18 8e 8b  27
c4b0 : 96 7e 9f fa a8 06 b3 ac  af
c4b8 : bd f3 c8 e6 d4 8f e1 f8  c1
c4c0 : ee 2e fd 7d c6 7d c6 7d  63
c4c8 : c6 7d c6 7d c6 7d c6 7d  1c
c4d0 : c6 7d c6 fc c6 95 c5 31  38
c4d8 : ea ad 14 03 8d d7 c4 ad  04
c4e0 : 15 03 8d d8 c4 a9 1e 8d  22
c4e8 : 14 03 a9 c5 8d 15 03 a9  82
c4f0 : e0 8d 06 dc a9 2e 8d 07  04
c4f8 : dc a9 82 8d 0d dc a9 11  7c
c500 : 8d 0f dc 60 ae 84 c4 bd  f6
c508 : 8d c4 85 ff bd 8c c4 85  b7
c510 : fe a0 00 b1 fe fe 8c c4  38
c518 : d0 03 fe 8d c4 60 ad 0d  fb
c520 : dc 48 29 02 d0 03 4c a1  45
c528 : c5 a5 fe 48 a5 ff 48 a5  4f
c530 : 01 29 fe 85 01 ad 0d dc  a2
c538 : a2 0e 8e 85 c4 a2 04 8e  c4
c540 : 84 c4 a2 02 8e 83 c4 20  68
c548 : 4d c6 ce 83 c4 30 12 ce  d0
c550 : 84 c4 ce 84 c4 ad 85 c4  d4
c558 : 38 e9 07 8d 85 c4 4c 47  37
c560 : c5 ad a7 c4 c9 07 d0 0c  ae
c568 : ad a9 c4 d0 07 20 df c5  b1
c570 : 90 c6 b0 1e a2 00 bd a4  be
c578 : c4 f0 03 20 b3 c5 a2 01  6f

c580 : bd a4 c4 f0 03 20 b3 c5  6a
c588 : a2 02 bd a4 c4 f0 03 20  4f
c590 : b3 c5 6c d5 c4 a5 01 09  8b
c598 : 01 85 01 68 85 ff 68 85  ae
c5a0 : fe 68 0d 0d dc 29 01 f0  b4
c5a8 : 04 58 6c d7 c4 68 a8 68  f1
c5b0 : aa 68 40 8e 83 c4 8a 0a  0d
c5b8 : 8d 84 c4 0a 0a 38 ed 83  1b
c5c0 : c4 8d 85 c4 ad 8a c4 85  92
c5c8 : fe ad 8b c4 85 ff ac 84  2c
c5d0 : c4 b1 fe 99 8c c4 c8 b1  d5
c5d8 : fe 99 8c c4 4c 7d c6 ad  86
c5e0 : 88 c4 18 69 03 8d 88 c4  46
c5e8 : 90 03 ee 89 c4 ad 88 c4  4c
c5f0 : 85 fe ad 89 c4 85 ff a0  4b
c5f8 : 00 b1 fe 8d 8a c4 c8 b1  97
c600 : fe f0 31 8d 8b c4 85 ff  69
c608 : ad 8a c4 85 fe a0 00 b1  34
c610 : fe 99 8c c4 c8 c0 06 d0  e3
c618 : f6 a9 00 8d 98 c4 8d 99  ae
c620 : c4 8d 9a c4 8d a7 c4 a9  66
c628 : 01 8d 9b c4 8d 9c c4 8d  5b
c630 : 9d c4 18 60 ad aa c4 f0  67
c638 : 0f ad 86 c4 8d 88 c4 ad  e3
c640 : 87 c4 8d 89 c4 4c ed c5  b0
c648 : 20 16 c7 38 60 ae 83 c4  7f
c650 : bd 98 c4 f0 22 de 98 c4  ad
c658 : f0 01 60 ad a8 c4 d0 0b  a1
c660 : bd 9e c4 29 fe ae 85 c4  c8
c668 : 9d 04 d4 ae 83 c4 bd 95  93
c670 : c4 f0 0a 9d 9b c4 60 de  02
c678 : 9b c4 f0 01 60 ae 83 c4  e5
c680 : a9 00 9d a4 c4 20 04 c5  0e
c688 : c9 00 d0 12 ae 83 c4 ad  3d
c690 : a7 c4 1d a1 c4 8d a7 c4  f6

c698 : a9 01 9d a4 c4 60 10 15  77
c6a0 : c9 f8 90 29 29 07 0a aa  77
c6a8 : bd c3 c4 85 fe bd c4 c4  a3
c6b0 : 85 ff 6c fe 00 c9 61 b0  65
c6b8 : 09 ae 83 c4 9d 92 c4 4c  ac
c6c0 : 7d c6 e9 61 ae 83 c4 9d  9c
c6c8 : 95 c4 4c 7d c6 a8 ae 83  f5
c6d0 : c4 bd 92 c4 9d 98 c4 98  93
c6d8 : 29 0f c9 0c 90 01 60 0a  23
c6e0 : aa 98 4a 4a 4a 4a a8 bd  c7
c6e8 : ac c4 85 fe bd ab c4 c0  05
c6f0 : 0e f0 06 46 fe 6a c8 d0  c9
c6f8 : f6 6c d3 c4 ae 85 c4 9d  17
c700 : 00 d4 a5 fe 9d 01 d4 ae  46
c708 : 83 c4 bd 9e c4 09 01 ae  26
c710 : 85 c4 9d 04 d4 60 a9 00  d6
c718 : 8d 0f dc a9 02 8d 0d dc  14
c720 : ad 9e c4 8d 00 d4 8d 07  ea
c728 : d4 8d 0e d4 ad d7 c4 8d  a9
c730 : 14 03 ad d8 c4 8d 15 03  5f
c738 : 60 78 a5 fe 48 a5 ff 48  60
c740 : ad 86 c4 8d 88 c4 ad 87  88
c748 : c4 8d 89 c4 20 ed c5 90  77
c750 : 02 b0 1e a9 08 a2 49 8d  3d
c758 : 05 d4 8e 06 d4 8d 0c d4  bf
c760 : 8e 0d d4 8d 13 d4 8e 14  96
c768 : d4 a9 0f 8d 18 d4 20 d9  e3
c770 : c4 68 85 ff 68 85 fe 58  29
c778 : 60 ff 00 ff 00 ff 00 00  d8
```

Listing 1. Der Sequenzer. Bitte mit dem MSE eingeben.

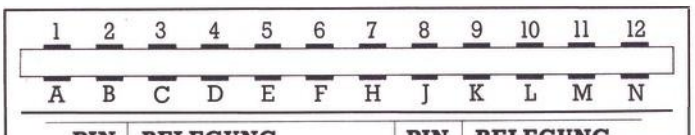
```
programm : musik              c800 c950
-----------------------------------------

c800 : 15 c8 00 31 c8 00 31 c8  82
c808 : 00 a4 c8 00 a4 c8 00 e9  f1
c810 : c8 00 00 00 00 1b c8 21  17
c818 : c8 27 c8 60 61 ef 48 ef  49
c820 : 00 60 61 ef 48 ef 00 0c  c3
c828 : 6d 93 9a 93 8a 93 9a 93  4f
c830 : 00 37 c8 69 c8 9d c8 18  f8
c838 : 61 ef 0c ba 24 ef 0c ba  53
c840 : 24 ef 0c 6d ba b6 b6 ba  be
c848 : b9 b5 b5 b9 18 61 b9 48  84
c850 : ef 0c c3 24 ef 0c c3 24  71
c858 : ef 0c 6d c3 bb bb c3 c2  4f
c860 : ba ba c2 18 61 c2 30 ef  f8
c868 : 00 0c 61 b3 b5 b6 b5 b3  8c

c870 : b5 b6 b5 b3 b5 6d b6 b3  6d
c878 : b3 b6 b5 b0 b0 b5 18 61  e6
c880 : b5 30 ef 0c b8 ba bb ba  90
c888 : b8 ba bb ba b8 ba 6d bb  72
c890 : b8 b8 bb ba b5 b5 ba 18  0f
c898 : 61 ba 30 ef 00 0c 6d 8a  8c
c8a0 : 93 9a 93 00 aa c8 b8 c8  cb
c8a8 : de c8 18 61 ef 79 d1 d1  d2
c8b0 : c8 c8 c6 c6 c1 61 c1 00  95
c8b8 : 0c 61 c6 c8 ca c8 c6 c8  df
c8c0 : 18 ca 0c c1 c3 c5 c3 c1  76
c8c8 : c3 18 c5 0c bb c1 c3 c1  e7
c8d0 : bb c1 18 c3 0c b6 b8 ba  b9
c8d8 : b8 b6 b8 18 ba 00 18 61  eb
c8e0 : ef 48 79 b6 b1 ab 61 a6  74
c8e8 : 00 ef c8 08 c9 2c c9 18  68

c8f0 : 61 ef 60 ef ef ef ef 06  a9
c8f8 : 67 9a a5 aa aa b5 ba ba  24
c900 : c5 18 61 ca ef ef ef 00  c1
c908 : 0c 61 a5 a6 a8 a6 a5 ef  39
c910 : a6 a5 a3 ef a5 a3 a1 ef  4d
c918 : a3 a1 a0 ef 60 ef 0c a3  af
c920 : a1 9b 3c ef 60 95 18 b5  1a
c928 : ef ef ef 00 0c 61 95 96  5a
c930 : 98 96 95 ef 96 95 93 ef  bb
c938 : 95 93 91 ef 93 91 90 ef  e1
c940 : 60 ef 0c 93 91 8b 3c ef  54
c948 : 60 8a 18 aa ef ef ef 00  87
```

Listing 2. Ein Musik-Demo. Bitte mit dem MSE eingeben.

C 64 extern — Der Weg nach draußen (Teil 3)

Nachdem wir in der letzten Folge die Programmierung der Control-Ports abgeschlossen haben, wenden wir uns heute einer der vielseitigsten Schnittstellen des C 64/VC 20 zu — dem User-Port.

Der User-Port führte neben dem Expansion-Port lange Zeit ein Schattendasein, denn an ihm können keine Programm-Module verwendet werden. Mittlerweile hat der User-Port aber weit aufgeholt. Er wird für Steuerzwecke, zum Anschluß eines Druckers, für die RS232 und auch zum Programmieren von EPROMs mit einem Zusatzgerät verwendet. Sehen wir uns diese interessante Schnittstelle etwas genauer an. Die Anschlußbelegung der User-Ports von C 64 und VC 20 zeigen die Bilder 1 und 2. Beachten Sie bitte, daß die Anschlüsse an der Ober- und Unterseite des User-Ports verschiedene Funktionen haben. Klemmen Sie deshalb dort niemals eine Krokodilklemme



PIN	BELEGUNG	PIN	BELEGUNG
1	GND	A	GND
2	+5V, max. 100 mA	B	FLAG2
3	RESET	C	PB0
4	CNT1	D	PB1
5	SP 1	E	PB2
6	CNT2	F	PB3
7	SP 2	H	PB4
8	PC2	J	PB5
9	SER. ATN IN	K	PB6
10	9V AC, max. 100 mA	L	PB7
11	9V AC, max. 100mA	M	PA2
12	GND	N	GND

Bild 1. Anschlußbelegung des C 64 User-Ports. (Bei Aufsicht auf die Computerrückseite)