Dem Klang auf der Spur (Teil 5)

Dieser Teil des Musikkurses ist auch für all jene interessant, die sich nicht ausschließlich für Musik interessieren. Es werden Algorithmen zur Generierung verschiedener Signale vorgestellt.

Dabei wird anhand des Source-Listings des Programms Modulator gezeigt, wie man diese Algorithmen unter zeitkritischen Nebenbedingungen programmieren kann.

Im zweiten Teil dieser Reihe wurde schon erwähnt, daß man jeden Signalverlauf durch eine Folge von Stützwerten beschreiben kann. Da man diese Stützwerte digital codieren kann, wird so die Signalerzeugung und -verarbeitung mit dem Computer möglich. Man muß dabei allerdings mit einer Abtastfrequenz arbeiten, die mindestens doppelt so hoch ist wie die höchste Frequenz, die im verarbeiteten Signal vorkommt. Für Audio-Signale in HiFi-Qulität ist somit eine Abtastfrequenz von mindestens 40 kHz erforderlich. Stellen wir dieser Frequenz einmal die Taktfrequenz von 1 MHz in unserem C 64 gegenüber: Eine Abtastperiode dauert bei 40 kHz 25 μs. Diese Zeit entspricht genau 25 Taktzyklen im C 64. Ein 6510-Maschinenbefehl dauert zwischen zwei und sieben Taktzyklen, das heißt, daß die CPU während einer Abtastperiode gerade vier bis maximal zwölf Befehle abarbeiten kann; zuwenig, um damit schon sinnvoll einen Signalabtastwert weiterzuverarbeiten. Die digitale Verarbeitung von Audiosignalen bleibt also zunächst einmal Hochleistungsrechnern und Spezialprozessoren vorenthalten.

Dreh- und Angelpunkt: Integer-Arithmetik

Wenn man sich aber wie bei dem in der letzten Folge vorgestellten Programm Modulator auf eine Abtastfrequenz von 60 Hz beschränkt, dann sieht die Sache schon sehr viel günstiger aus. Während einer Abtastperiode von 16,6 ms (entsprechend 16600 Taktzyklen) kann man bei geeigneter Programmierung schon eine ganze Menge machen. Die Abtastfrequenz ist aber nicht das einzig wichtige Kriterium bei der Signalverarbeitung. Eine Rolle spielt auch die **Genauigkeit**, mit der die Abtastwerte dargestellt und verrechnet werden. Natürlich gilt hier: Je genauer, desto besser. Genauigkeit kostet aber wieder Rechenzeit, sobald man die Wortlänge des verfügbaren Prozessors (hier leider nur 8 Bit) überschreitet. Die Arithmetik-Routinen des Basic-Interpreters arbeitet zum Beispiel im Fließkommaformat mit 32-Bit-Mantisse. Sie bietet damit eine Genauigkeit, die selbst für sehr anspruchsvolle Probleme aus der Signalverarbeitung mehr als genug sein dürfte. Diese Routinen sind jedoch so langsam, daß sie auch in 16,6 ms nichts Vernünftiges tun können. Wir werden also unsere eigenen Arithmetik-Befehlsfolgen programmieren müssen und dabei einen Kompromiß zwischen Genauigkeit und Geschwindigkeit machen. Gleitkomma-Arithmetik ist zu aufwendig und für unsere Zwecke auch gar nicht erforderlich. Eine Genauigkeit von nur 8 Bit reicht allerdings auch nicht immer aus. So haben ja auch manche SID-Parameter eine Länge von 12 oder 16 Bit. Im Programm Modulator wird größtenteils mit 16-Bit-Zweierkomplex-Größen gerechnet. Es sei in diesem Zusammenhang auf dem Assembler-Kurs (Teil 3 im 64'er, Ausgabe 11/84) verwiesen, wo ausführlich beschrieben wird, wie man negative Zahlen im Zweierkomplement darstellt. Addition und Subtraktion von Zweierkomplement-Größen werden direkt durch die CPU-Befehle ADC und SBC sowie durch drei Flaggen (Negativ, Carry und Overflow) unterstützt. Für die Multiplikation gibt es dagegen keinen Maschinenbefehl. Da die Multiplikation in Modulator aber eine zentrale Rolle spielt, benötigen wir für sie ein effizientes (das heißt möglichst schnelles) Maschinenprogramm.

Die Multiplikation

Was ist 43 x 13? Die wenigsten Menschen dürften die Antwort auf einen Schlag parat haben, so wie zum Beispiel auf die Frage, was 3 x 7 ist. Wir brauchen 3 x 7 nicht auszurechnen, weil wir es auswendig wissen. Den Wert des Produkts 43 x 13 werden die wenigsten auswendig kennen und daher zu rechnen anfangen. Eine solche Rechnung könnte (ausführlich) so aussehen:

```
(1)      43 x 13
(2)         43
(3)        129
(4)        559
```

In Zeile (1) stehen dabei noch einmal die Faktoren. Zeile (2) stellt das Teilprodukt 43 x 10 = 430 dar, Zeile (3) das Teilprodukt 43 x 3 = 129. Durch geeignetes Einrücken braucht man die Null bei 430 in Zeile (2) nicht mitschreiben. In Zeile (4) werden schließlich die Teilprodukte addiert. Dieser vertrauten Rechenweise liegt das **Distributivgesetz** zugrunde, welches die Bildung von Teilprodukten erlaubt:

(5) 43x(10+3)=(43x10)+(43x3)

Die »komplizierte« Multiplikation 43 x 13 wird also auf »einfachere« Multiplikationen 43 x 10 und 43 x 3 zurückgeführt. Bezeichnen wir in unserem Beispiel die Zahl 43 als **Multiplikant (MD)** und 13 als **Multiplikator (MR)**, so können wir das Multiplikationsschema so formulieren: »Multipliziere MD mit den einzelnen Dezimalstellen von MR und addiere die Teilproduktion, die mit den entsprechenden Zehnerpotenzen 1,10,1000 etc. zu skalieren sind. Die Skalierung erreicht man aber einfach durch Linksverschieben um 0,1,2 etc. Dezimalstellen.«

Man kann den Multiplikator aber auch anders zerlegen, zum Beispiel in Zweierpotenzen:

13 = 8 + 4 + 1

und so multiplizieren:

```
(6)  43x13 = 8x43 + 4x43 + 1x43
           =  344 +  172 +   43
           =  599
```

Man benötigt hier als Summanden Produkte von MD mit Zweierpotenzen, die man leicht durch wiederholtes Verdoppeln von MD erhalten kann. Auf diese Weise sollen übrigens schon die alten Ägypter multipliziert haben. Wenn man nun MD und MR im Binärsystem darstellt, kann man besonders einfach multiplizieren: Die Produkte von MD mit Zweierpotenzen erhält man ganz einfach durch wiederholtes Linksverschieben. In unserem Beispiel gilt in binärer Schreibweise: MD = 101011, MR = 1101. Es ergibt sich das Schema:

```
                MD      MR
(7)        101011 x 1101
(8)        101011
(9)       101011
(10)    +  101011
(11)      1000101111
```

In Zeile (7) stehen MD und MR. Die Zeilen (8), (9) und (10) entsprechen den Teilpodukten 43 x 8, 43 x 4 und 43 x 1, die durch Linksverschiebung aus MD hervorgehen. Die Summe in Zeile (11) ist genau die Binärdarstellung von 559 (nachrechnen!).

Nach diesem Schema kann man nun einen Algorithmus formulieren: Es sei dazu N die Zahl der Binärstellen von MR:

```
1 SUM:=0;
2 FOR I:=N—1 DOWNTO 0 DO
3 BEGIN
4 SUM := LINKS(SUM);
5 IF MR(I)=1 THEN SUM:=SUM+MD
6 END
```

Der Algorithmus ist hier formal in einem Pascal-ähnlichen Stil dargestellt. SUM wird zunächst mit 0 vorbesetzt und dann N-mal nach links geschoben. Immer wenn dabei, von links nach rechts gezählt, in MR eine Eins auftritt, wird MD zu SUM addiert.

Sehen wir uns für unser Beispiel einen Trace des Programms an:
MD = 101011 MR = 1101 N = 4

Zeile	I	MR(I)=1	SUM	AKTION
1	undef	undef	0	
4	3	true	0	LINKS
5	3	true	101011	+
4	1	true	1010110	LINKS
5	2	true	10000001	+
4	1	false	100000010	LINKS
5	1	false	100000010	nichts
4	0	true	1000000100	LINKS
5	0	true	1000101111	+

Wir wollen diesen Algorithmus nun konkret in Maschinensprache realisieren. Wenn MD und MR zunächst auf 8 Bit begrenzt werden, kann man auf sie mit einem einzigen Maschinenbefehl zugreifen. Die Variable SUM hält man am besten im Akkumulator, weil man sie zum Addieren sowieso dorthin laden müßte. Auch die Linksverschiebung des Akkumulators ist wesentlich schneller als die einer Speicherzelle. Beim Addieren in den Akkumulator und beim Linksverschieben treten allerdings Überträge auf, die nicht verloren gehen dürfen, da diese je gerade die höherwertigen Bits von SUM darstellen. Das Endprodukt, das sich in SUM bildet, kann bis zu 16 Bit lang werden. (In unserem Beispiel sind es immerhin schon 9 Bit.) Bild 1 zeigt eine elegante Realisierung des Algorithmus, die mit nur zwei Speicherplätzen auskommt.

Die langen Rechtecke stellen die Speicherstellen MD, MR und

den Akkumulator dar, fette Linien stehen für Bytepfade, dünne für Bitpfade. »+« versinnbildlicht die Addition A := A + MD. In diesem Schema werden MR und A zusammen (wie ein 16-Bit-Register) nach links geschoben. Dadurch erscheinen die Bits von MR nacheinander in der Carry-Flagge und können so leicht abgefragt werden. MD wird nur dann zum Akku addiert, wenn das durch Linksverschiebungen aus MR gewonnene Bit Eins ist. Ein Übertrag bei der Addition in den Akku muß natürlich nach MR weitergegeben werden, da MR gleichzeitig auch die höherwertigen Bits von SUM enthält. Durch die doppelte Nutzung der Speicherstelle MR wird der Multiplikator zwar durch das höherwertige Byte von SUM überschrieben, man spart sich dadurch aber einen Schiebebefehl und einen Speicherplatz. Da wir wegen schnelleren Zugriffs MD und MR in der Zero-Page plazieren werden, und da der freie Platz dort knapp ist, ist die Einsparung von Speicherplatz durchaus gerechtfertigt. Hier das Programm, das ausführlich besprochen werden soll:

```
MUL   LDA #0      (2)   SUM:=0
      LDX #8      (2)   Schleifen-
                        zähler
LOOP  ASL A       (2)   Register-
                        paar
      ROL MR      (5)   (MR.A)
                        n. links
      BCC NEXT (2/3)
      CLC         (2)
      ADC MD      (3)   SUM:=
                        SUM+MD
      BCC NEXT (2/3)
      INC MR      (5)   Übertrag
                        nach MR
NEXT  DEX         (2)
      BNE LOOP (2/3)  nächster
                      Lauf
```

Die Zahlen in Klammern geben die Ausführungszeiten der Befehle in Taktzyklen an. Sie sind aus Tabelle 1 entnommen. Bei den Verzweigungen nehmen wir der Einfachheit halber an, daß keine Page-Grenzen übersprungen werden, sonst müßte man im Falle eines Sprunges vier statt drei Takte in Rechnung stellen. Zur Arbeitsweise des Programms: Zuerst wird SUM mit 0 vorbesetzt. Dazu genügt es, den Akku mit 0 zu besetzen, da die höherwertigen Bits von SUM erst durch den Schiebeprozeß entstehen. Das X-Register zählt die Schleifendurchläufe. Innerhalb der Schleife wird zunächst das Registerpaar (MR.A) durch das Befehlspaar ASL,ROL nach links verschoben. Beide Befehle schieben nach links, wobei Bit 7 in die Carry-Flagge geschoben wird. Der Unterschied der beiden Befehle besteht aber darin, daß ASL das Bit 0 immer mit Null besetzt, während ROL Bit 0 mit dem Wert besetzt, den die Carry-Flagge vor dem ROL-Befehl hatte. In unserem Fall ist das gerade das aus dem vorhergehenden ASL stammende Bit 7 vom Akku. Nach der Verschiebung zeigt das aus MR stammende Carry-Bit an, ob MD zu SUM addiert werden soll oder nicht. Falls nicht, wird mit BCC NEXT die Addition übersprungen. Die Addition selbst berücksichtigt durch ein weiteres BCC NEXT/INC MR einen eventuell auftretenden Übertrag nach MR. Die Speicherstelle MR enthält zwar gleichzeitig Teile vom Multiplikator und von SUM, man kann sich aber überlegen, daß ein Übertrag nach MR nur den SUM-Teil, aber nicht den Multiplikator-Teil beeinflußt. Nach dem Verlassen der Schleife steht schließlich das 16-Bit-Produkt (die Variable SUM) im Registerpaar (MR.A). Der Multiplikator wurde überschrieben, der Multiplikant in MD dagegen ist unverändert erhalten geblieben.

			Immediate	Zero Page	ZP,X ZP,Y Abs.	Abs., X/Y	(Ind, X)	(Ind), Y
LDA LDX LDY ADC	AND ORA EOR SBC	BIT CMP CPX CPY	2	3	4	4/5*	6	5/6*
	STA STX STX		—	3	4	5	6	6

			Akku	Zero Page	ZP,X Abs.	Abs.,X
ASL LSR	ROL ROR	INC DEC	2	5	6	7

*die größere Zahl gilt, wenn beim Indizieren eine Page-Grenze überschritten wird

	Absolut	(Indirekt)
JMP	3	5
JSR	6	—

				Relativ
BCC BCS	BEQ BNE	BMI BPL	BVC BVS	2/3/4**

						Implizit
CLC CLD CLI	SEC SED SEI	DEX DEY INX INY	TAX TAY TSX	TXA TYA TXS	NOP	2
		PHA	PHP			3
		PLA	PLP			4
		RTI	RTS			6

**2, wenn nicht gesprungen wird
3, wenn gesprungen wird und das Sprungziel auf der gleichen Page liegt
4, wenn auf ein Ziel in einer anderen Page gesprungen wird

Tabelle 1. Ausführungszeiten der 6502/6510-Maschinenbefehle (in Taktzyklen)

Zeitbedarf

Es soll hier exemplarisch gezeigt werden, wie man den genauen Zeitbedarf eines Maschinenprogramms ermittelt. Die Ausführungszeit des Multiplikationsprogramms ist nicht einheitlich. Sie hängt von den Anfangswerten von MR und MD ab, welche das Verhalten des Programms an den beiden Verzweigungsstellen (BCC NEXT) beeinflussen. Wir werden hier also den günstigsten (in bezug auf die Rechenzeit) und den ungünstigsten Fall untersuchen. Im ungünstigsten Fall muß bei jedem Schleifendurchlauf addiert werden, und zusätzlich tritt bei jeder Additon ein Übertrag auf. Der Zeitbedarf eines Schleifendurchlaufes beträgt dann:

2(ASL)+5(ROL)+2(BCC)
+2(CLC)+3(ADC)+2(BCC)+5
(INC)+2(DEX)+3(BNE)=
= 26 Takte

Die Gesamtdauer der Multiplikation ergibt sich dann so:
2(LDA)+2(LDX)+8*26(Schleife)
-1=**211**

Die -1 kommt dadurch zustande, daß beim letzten Schleifendurchlauf bei BNE nicht gesprungen wird und dadurch nur 2 statt 3 Takte benötigt werden.

Im günstigsten Fall (MR=0, die Addition wird immer übersprungen) braucht die Schleife:
2(ASL)+5(ROL)+3(BCC)
+2(DEX)+3(BNE)=
=16 Takte
Gesamtdauer:
2(LDA)+2(LDX)+8*16
(Schleife)-1=**131**

Die Ausführungszeit der Multiplikation liegt also immer zwischen 131 und 211 Takten, wobei die Grenzwerte wohl selten erreicht werden dürften. Man kann im Mittel wohl mit zirka 170 Takten rechnen. Es ist für unsere Zwecke sehr wichtig, diese Größe zu kennen. Wenn wir in unserem Programm Modulator für einen Schritt eine Zeit von maximal 16,6 ms zur Verfügung haben, so können wir daraus eine theoretische Obergrenze für die Anzahl der in einem Schritt ausführbaren Multiplikationen ableiten. Sie liegt bei unserer 8-mal-8-Bit-Multiplikation etwa bei 75, wenn man den ungünstigsten Fall zugrundelegt.

Multiplikation mit größerer Wortlänge

Bei Modulator wird die Multiplikation für folgende Zwecke benötigt: Die LFOs und der Hüllkurvengenerator erzeugen Werteverläufe mit maximaler Amplitude. Das bedeutet bei der 16-Bit-Zweierkomplement-Arithmetik, in der hauptsächlich gerechnet wird, daß die Werte den zur Verfügung stehenden Bereich von -32768 bis +32767 meistens voll ausschöpfen. Nun möchte man aber oft das Modulationsziel, zum Beispiel die Frequenz einer SID-Stimme, nur um einige Hertz nach oben und unten modulieren. Man möchte die Tiefe dieser Modulation aber auch möglichst kontinuierlich steuern können, so daß zum Beispiel auch Modulationstiefen von einer Quinte oder gar einer Oktave möglich sind. Aus diesem Grund muß das Modulationssignal erst mit einem geeigneten Skalierungsfaktor multipliziert werden. Anschließend kann es durch einfache Addition zur Zielgröße diese in dem gewünschten Sinn modulieren.

Bei der Modulation von Tonhöhen ergibt sich außerdem noch

ein weiteres Problem: Dort kommt es nicht auf absolute, sondern auf relative Frequenzverschiebungen an. Ein Beispiel: Ein 500-Hz-Ton wird um ±5 Hz moduliert. Um bei einem 1 000-Hz-Ton den gleichen Effekt zu erzielen, muß man ihn um ±10 Hz modulieren. Der Modulationsbetrag muß also bei Tonhöhen zusätzlich mit der zu modulierenden Frequenz selbst skaliert werden, was eine weitere Multiplikation erforderlich macht.

Die Wortlängen der Modulationsziele sind:

Tonfrequenzen	16 Bit
Pulsweiten	12 Bit
Filterfrequenz	8 Bit
Lautstärke	4 Bit

Die Filterfrequenz ist beim SID zwar eine 11-Bit-Größe, da aber feine Frequenzunterschiede in der Filterfrequenz nicht hörbar sind, werden nur die oberen 8 Bit moduliert.

Zur Steuerung der Modulationstiefe genügen 8-Bit. Eine fein gestufte Modulation ist ohnehin nur bei der Tonhöhenmodulation erforderlich. Hier genügt es aber, wenn das Modulationssignal selbst einen fein gestuften Verlauf (16 Bit) hat. Die durch 8 Bit realisierbaren 255 verschiedenen Modulationstiefen reichen aus, um alles vom feinsten Vibrato über Tonhöhensprünge in allen musikalisch sinnvollen Intervallen bis hin zur Sirene mit weitem Frequenzbereich zu verwirklichen.

Wir benötigen also eine 16 x 8-Bit-Multiplikation. Der Algorithmus von Bild 1 ließe sich in diese Richtung leicht erweitern. Man kann etwa entweder MR auf 16 Bit verlängern und benötigt dann 16 statt 8 Schleifendurchläufe oder man verlängert MD und A auf 16 Bit. In letzterem Fall benötigt man weiterhin nur 8 Schleifendurchläufe wobei aber, im Falle einer Eins aus MR, zwei 16-Bit-Größen (MD und A) addiert werden müssen. Natürlich braucht man für das höherwertige Byte von A einen weiteren Speicherplatz in der Zero-Page.

In Modulator wird ein anderer Weg eingeschlagen. Er wird durch Bild 2 beschrieben. Dieses erscheint zwar zunächst sehr kompliziert, das zugehörige Programm benötigt aber eine geringere Ausführungszeit. Vorgegeben sind ein 16-Bit-Multiplikator im Registerpaar (MR + 1.MR) und ein 8-Bit-Multiplikant in MD. (Der Einfachheit halber werden hier Zero-Page-Speicherplätze »Register« genannt.) Das Ergebnis des Programms soll ein 24-Bit-Produkt im Register Tripel (MR + 1.MR.A) sein.

Zuerst wird das niederwertige Teilprodukt MR x MD gebildet. Das Rechteck mit dem Kreuz steht für das Verfahren aus Bild 1, welche wie schon beschrieben, ein 8 x 8-Bit-Produkt in (MR.A) liefert. A wird im Y-Register zwischengespeichert. Anschließend werden MR+1 und MD ebenfalls nach Bild 1 multipliziert. Das Ergebnis ist das höherwertige Teilprodukt in (MR+1.A). Schließlich müssen die Teilprodukte nur noch mit richtiger Skalierung addiert werden. Dazu wird das höherwertige Byte des niederwertigen Teilprodukts, das in MR steht, zum niederwertigen Byte des höherwertigen Teilprodukts, das sich schon im Akku befindet, addiert. Dabei muß ein eventueller Übertrag nach MR+1 berücksichtigt werden. Das niederwertige Byte des niederwertigen Teilprodukts wird nur noch vom Y-Register in den Akku übertragen, wo es den niederwertigsten Teil des Endprodukts darstellt. In Modulator werden allerdings grundsätzlich nur 16-Bit-Größen weiterverarbeitet, so daß diese untersten 8 Bit des Produktes unberücksichtigt bleiben.

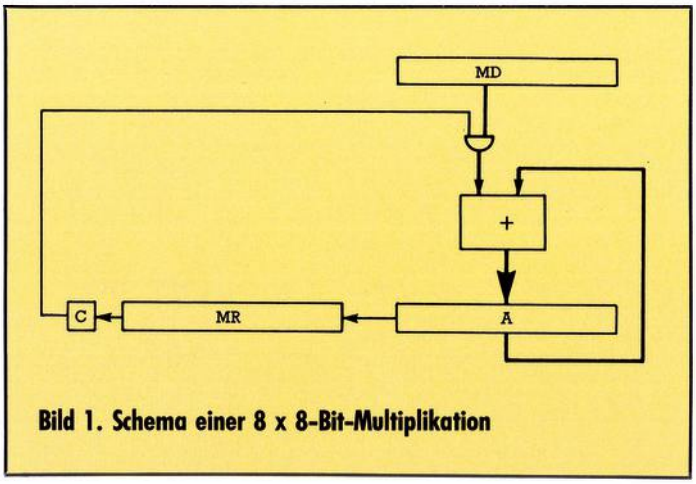

Bild 1. Schema einer 8 x 8-Bit-Multiplikation

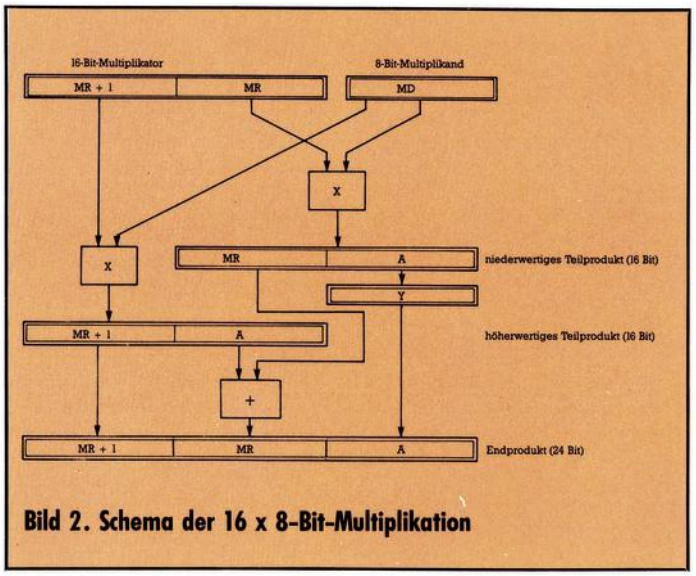

Bild 2. Schema der 16 x 8-Bit-Multiplikation

Im Source Listing zu Modulator steht das zugehörige Programm MULU in den Zeilen 1680 bis 1970. Zunächst steht dort zweimal hintereinander das schon vorgestellte 8 x 8 Bit-Multiplikationsprogramm, anschließend werden ab Zeile 1910 die Teilprodukte addiert. Eine Analyse ergibt eine Laufzeit von minimal **282** Takten und maximal **446** Takten.

Alle bisher beschriebenen Multiplizierer arbeiten nur dann korrekt, wenn man die Faktoren als positive Ganzzahlen interpretiert. Sie sind ohne Ergänzung nicht für Zweierkomplement-Größen geeignet. Das Programm MULS ab Zeile 2020 ist eine solche Ergänzung. Es berücksichtigt das Vorzeichen des Multiplikators. Ist dieser positiv, so wird sofort nach MULU verzweigt. Ein negativer Multiplikator wird zunächst negiert, wodurch er positiv wird (Zeile 2040 bis 2100), MULU wird als Unterprogramm aufgerufen, und schließlich wird das positive Produkt noch einmal negiert, was dann ein korrektes Resultat liefert. Der 8-Bit-Multiplikant wird aber nach wie vor nur als positive Zahl behandelt.

Die LFOs

Sie erzeugen die für Modulationen sinnvollen Kurvenverläufe als Folge von 16-Bit-Zweierkomplement-Zahlen. Am häufigsten wird die Dreieckskurve benötigt, da sie keine Sprünge macht und daher bei Anwendung auf Tonhöhen und auf Pulsweiten am angenehmsten klingt. Der Sägezahn eignet sich mehr für »härtere Effekte und für Videospiele, wo stark und schnell modulierte Töne oft zu hören sind. Die Rechteckkurve eignet sich für Triller (bei Frequenzmodulation), für mandolinenartige Effekte (bei Modulation von Lautstärke und Filterfrequenz) sowie für rhythmische Effekte (bei Frequenzmodulation mit größerer Modulationstiefe).

Rechnerisch kann man einen Sägezahnförmigen Wertverlauf besonders einfach erzeugen. Bei Modulator wird einfach ein 16-Bit-Wert zyklisch hochgezählt. Zyklisch bedeutet, daß immer wieder beim Minimalwert angefangen wird, wenn der Maximalwert überschritten wird. Das geschieht bei begrenzter Wortlänge automatisch durch Überlauf, den man hier absichtlich unberücksichtigt läßt. Im Modulator-Programm wird der Werteverlauf durch das Wort (= Bytepaar) SAWUP repräsentiert. SAWUP wird einfach um den Betrag im Wort LFOF hochgezählt. Dadurch ist die resultierende Frequenz der Sägezahnkurve direkt proportional zum Wert LFOF. Im Programm wird in Zeile 2300 bis 2420 erst das Steuerregister LFOC abgefragt. Im Falle des HOLD- oder RESET-Status braucht nichts berechnet zu werden. Im Falle des RUN-Status wird SAWUP in den Zeilen 2430 bis 2510 hochgezählt. Der aufsteigende Sägezahn wird dann gewissermaßen als »Master« für die anderen Kurvenformen herangezogen. Bild 3 zeigt, wie diese aus SAWUP gewonnen werden.

Interessant ist, daß gleichgültig, ob man die SAWUP-Werte im Zweierkomplement oder grundsätzlich positiv interpretiert, sich immer der gleiche Kurvenverlauf ergibt (gestrichelte und durchgezogene Kurve bei SAWUP).

Die Rechteckkurve entsteht dadurch, daß man, gesteuert durch SAWU, zwischen den Extremwerten +LFOA x 2 (hoch) 7 und -LFOA x 2 (hoch) 7 hin- und herschaltet. Man spart sich so die sonst anschließend fällige Multiplikation mit der LFO-Amplitude (= Modulationstiefe) LFOA. Hin- und hergeschaltet wird, wenn der Sägezahnwert einen vorgegebenen Schwellwert über- beziehungsweise unterschreitet. Dieser Schwellwert ist nichts anderes als die Pulsweite LFOP. SAWDOWN erhält man einfach durch Negieren von SAWUP. Bildet man das Maximum von SAWUP und SAWDOWN, so erhält man einen dreieckförmi-

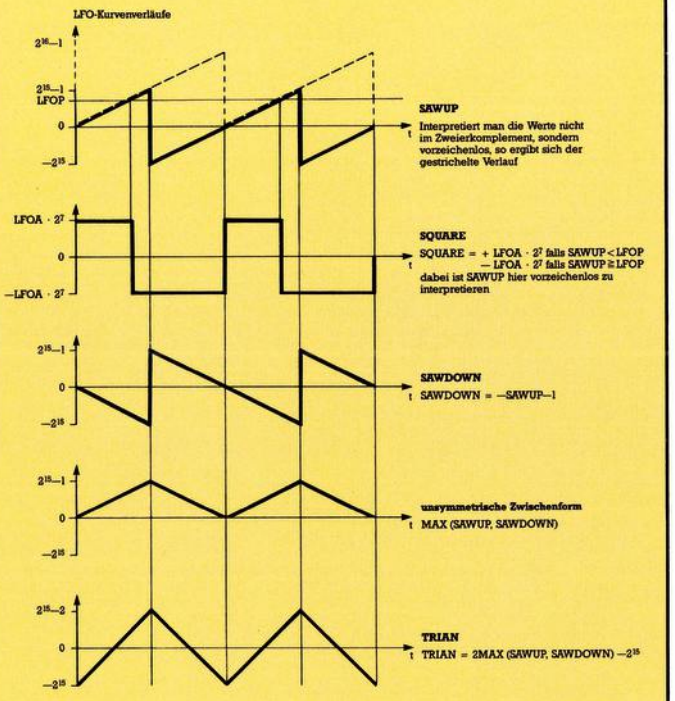

Bild 3. So gehen die von einem Software-LFO erzeugbaren Kurven aus SAWAP hervor

```
ASS.64 $7000 ENGAGED
2
1030:  C075                .OPT P,00
 ;*****************************************************
 ;*                                                   *
 ;*                  M O D U L A T O R                *
 ;*                                                   *
 ;* - PROGRAMMIERBARE SOFTWARE-LFOS UND HUELLKURVENGENERATOREN *
 ;*   ZUR MODULATION DER SID-PARAMETER                *
 ;*   FREQUENZ,PULSWEITE,FILTERFREQUENZ UND LAUTSTAERKE *
 ;* - DREISTIMMIGES FREQUENZ-PORTAMENTO               *
 ;*                                                   *
 ;* THOMAS KRAETZIG                    MAERZ   1985   *
 ;*                                                   *
 ;*****************************************************
 ;
 ; ZERO PAGE
 ;
1190:  C075        ZAEHLER  =   $9B     ;ADRESSVERSATZ FUER KSV
1200:  C075        MR       =   $FB     ;MULTIPLIKATOR
1210:  C075        MD       =   $FD     ;MULTIPLIKAND
1220:  C075        LFONR    =   $FE     ;ADRESSVERSATZ FUER LFO-BLOCK
1230:  C075        STINR    =   $FE     ;ADRESSVERSATZ FUER STIMMEN-BLOCK
1240:  C075        TEMP     =   $FF     ;ZWISCHENSPEICHER
1250:  C000                 *=   $C000   ;BASIS FUER STEUERPARAMETER
 ;
 ; STEUERPARAMETER
 ;
1290:  C002        F        *=   *+2     ;FREQUENZ
1300:  C004        PW       *=   *+2     ;PULSWEITE
1310:  C005        PORTA    *=   *+1     ;PORTAMENTO-RATE
1320:  C007        FP       *=   *+2     ;F IM PORTA-VERLAUF (DYNAMISCH)
1330:  C015                 *=   *+14    ;2 WEITERE SOLCHE BLOECKE
1340:  C017        FILT     *=   *+2     ;FILTERFREQUENZ
1350:  C018        MODLAUT  *=   *+1     ;FILTERMODUS/LAUTSTAERKE
1360:  C020        KSV      *=   *+8     ;KREUZSCHIENENVERTEILER
1370:  C022        LFOF     *=   *+2     ;LFO-FREQUENZ
1380:  C023        LFOP     *=   *+1     ;LFO-PULSWEITE
1390:  C024        LFOA     *=   *+1     ;LFO-AMPLITUDE
1400:  C025        LFOC     *=   *+1     ;LFO-STEUERREGISTER
1410:  C043                 *=   *+30    ;6 WEITERE LFO-STEUERBLOECKE
1420:  C044        A        *=   *+1     ;ATTACK-RATE
1430:  C045        D        *=   *+1     ;DECAY-RATE
1440:  C046        S        *=   *+1     ;SUSTAIN-PEGEL
1450:  C047        R        *=   *+1     ;RELEASE-RATE
1460:  C048        EGA      *=   *+1     ;HUELLKURVE-AMPLITUDE
1470:  C049        EGC      *=   *+1     ;HUELLKURVE-STEUERREGISTER
 ;
 ; DYNAMISCHE PARAMETER
```

Listing 1. Dokumentiertes Assemblerlisting von »Modulator«

(Fortsetzung auf Seite 156)

gen Kurvenverlauf, der allerdings nur positive Werte annimmt. Durch Verdoppeln dieser Werte und Verschiebung um 2 (hoch) 15 nach unten erhält man dann eine symmetrische Dreieckskurve maximaler Amplitude.

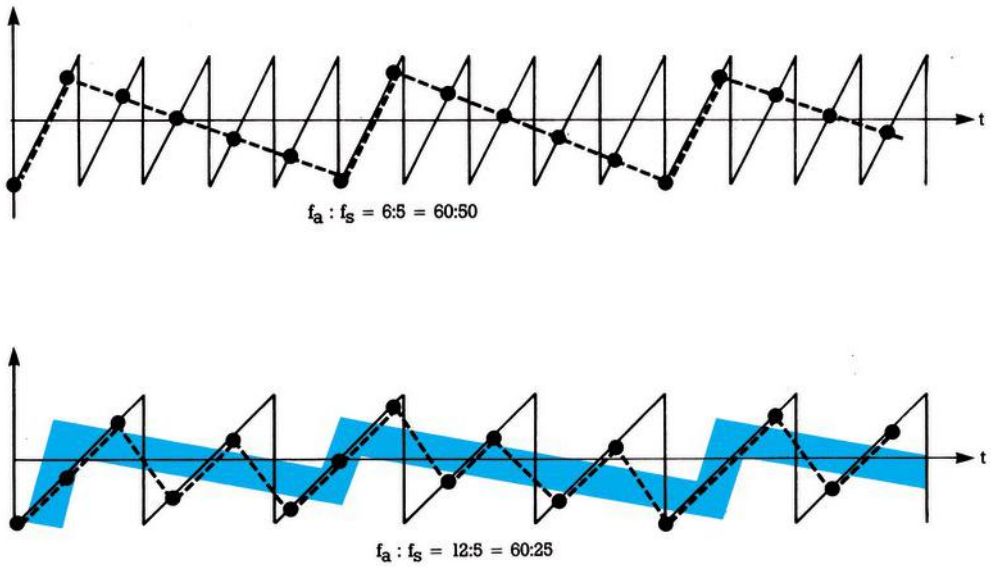

Bild 4. Demonstration des Aliasing-Effekts

Im Programm wird in Zeile 2530 bis 2590 aus LFOC ermittelt, welche Kurvenform überhaupt erzeugt werden soll, und entsprechend weiterverzweigt. Mit Ausnahme des Rechtecks, das schon mit seiner endgültigen Amplitude aufwartet, wird der errechnete Wert noch mit der Amplitude LFOA multipliziert (Zeile 3090 bis 3170). Das LFO-Programm rechnet alle 7 LFOs. Dabei wird auf die jeweiligen Parameter indiziert zugegriffen. Das Byte LFONR enthält dazu einen Adreß-Offset, der vom LFO-Programm in das X-Register geladen wird. Dieser Offset muß vom Programm, welches das LFO-Programm aufruft, korrekt zur Verfügung gestellt werden.

Aliasing-Parasitäre Frequenzen

Bei der eben beschriebenen Erzeugung der LFO-Kurvenformen tritt bei etwas höheren Frequenzen, etwa ab 10 Hz, noch ein interessantes Phänomen auf. Man kann die Erzeugung eines Sägezahnverlaufs durch zyklisches Hochzählen eines Wortes auch als Abtastung einer hypothetischen, kontinuierlichen Sägezahnkurfe auffassen. Die Abtastfrequenz ist in unserem Fall mit 60 Hz fest. Die Frequenz der hypothetischen Sägezahnkurve kann man aber durch den Parameter LFOF sehr feinstufig zwischen 0 und 60 Hz variieren. Wie soll aber zum Beispiel eine LFO-Kurve mit 50 Hz aussehen, wenn

Fortsetzung auf Seite 173

```
                ;
1510:  C04B     SAWUP   *=   *+2      ;AUFSTEIGENDER SAEGEZAHN
1520:  C04D     KURVE   *=   *+2      ;AKTUELLER LFO-WERT
1530:  C04E             *=   *+1      ;(UNBENUTZT)
1540:  C06C             *=   *+30     ;6 WEITERE DYNAMISCHE LFO-BLOECKE
1550:  C06E     E       *=   *+2      ;HUELLKURVENWERT
1560:  C070     EKURVE  *=   *+2      ;BEWERTETER HUELLKURVENWERT
1570:  C071     EPHASE  *=   *+1      ;0=ATTACK 1=DECAY
                ;
                ; KONSTANTEN UND SONSTIGE
                ;
1610:  C071     SID     =    $D400    ;SID-BASISADRESSE
1620:  C073     ZEIT    *=   *+2      ;FUER ZEITMESSUNG
1630:  C075     ZEIT1   *=   *+2
                ;-------------------------------------------------------
                ; MULTIPLIKATION 16 BIT (GANZZAHL=UNSIGNED) * 8 BIT
                ; (MR+1,MR,A)(24) = (MR+1,MR)(16) * MD (8)
                ;-------------------------
1680:  C075 A9 00    MULU    LDA  #0
1690:  C077 A2 08            LDX  #8
1700:  C079 0A       LOOP1   ASL  A        ;MR(8)*MD(8)
1710:  C07A 26 FB            ROL  MR
1720:  C07C 90 07            BCC  NEXT1
1730:  C07E 18               CLC
1740:  C07F 65 FD            ADC  MD
1750:  C081 90 02            BCC  NEXT1
1760:  C083 E6 FB            INC  MR
1770:  C085 CA       NEXT1   DEX
1780:  C086 D0 F1            BNE  LOOP1
1790:  C088 A8               TAY   ;ZWISCHENSPEICHERN
1800:  C089 8A               TXA   ;STATT LDA #0
1810:  C08A A2 08            LDX  #8
1820:  C08C 0A       LOOP2   ASL  A        ;MR+1(8)*MD(8)
1830:  C08D 26 FC            ROL  MR+1
1840:  C08F 90 07            BCC  NEXT2
1850:  C091 18               CLC
1860:  C092 65 FD            ADC  MD
1870:  C094 90 02            BCC  NEXT2
1880:  C096 E6 FC            INC  MR+1
1890:  C098 CA       NEXT2   DEX
1900:  C099 D0 F1            BNE  LOOP2
1910:  C09B 18               CLC   ;TEILPRODUKTE ADDIEREN
1920:  C09C 65 FB            ADC  MR
1930:  C09E 85 FB            STA  MR
1940:  C0A0 90 02            BCC  NEXT3
1950:  C0A2 E6 FC            INC  MR+1
1960:  C0A4 98       NEXT3   TYA
1970:  C0A5 60               RTS
                ;-------------------------------------------------------
                ; MULTIPLIKATION 16 BIT (ZWEIERKOMPLEMENT=SIGNED) * 8 BIT
                ; (MR+1,MR,A)(24) = (MR+1,MR)(16) * MD (8)
                ;-------------------------------------------------------
2020:  C0A6 A5 FC    MULS    LDA  MR+1
2030:  C0A8 10 CB            BPL  MULU     ;MR POSITIV, NICHTS WEITER ZU TUN
2040:  C0AA 38               SEC   ;MR NEGIEREN
2050:  C0AB A9 00            LDA  #0
2060:  C0AD E5 FB            SBC  MR
2070:  C0AF 85 FB            STA  MR
2080:  C0B1 A9 00            LDA  #0
2090:  C0B3 E5 FC            SBC  MR+1
2100:  C0B5 85 FC            STA  MR+1
2110:  C0B7 20 75 C0          JSR  MULU
2120:  C0BA 85 FF             STA  TEMP
2130:  C0BC 38               SEC   ;PRODUKT NEGIEREN
2140:  C0BD A9 00            LDA  #0
2150:  C0BF E5 FF            SBC  TEMP
2160:  C0C1 A8               TAY
2170:  C0C2 A9 00            LDA  #0
2180:  C0C4 E5 FB            SBC  MR
2190:  C0C6 85 FB            STA  MR
2200:  C0C8 A9 00            LDA  #0
2210:  C0CA E5 FC            SBC  MR+1
2220:  C0CC 85 FC            STA  MR+1
2230:  C0CE 98               TYA
2240:  C0CF 60               RTS
                ;-------------------------------------------------------
                ; LFO N UM EINEN SCHRITT WEITERSCHALTEN
                ; DAS PROGRAMM ERWARTET EINE GROESSE DER GESTALT
                ; 5 * N  (N=0...6) IN LFONR
                ;-------------------------------------------------------
2300:  C0D0 A6 FE    LFO     LDX  LFONR
2310:  C0D2 BD 24 C0          LDA  LFOC,X
2320:  C0D5 29 06             AND  #$06
2330:  C0D7 C9 04             CMP  #$04     ;HOLD "?"
2340:  C0D9 F0 12             BEQ  LFORTS   ;DANN NICHTS ZU TUN
2350:  C0DB C9 06             CMP  #$06     ;RUN "?"
2360:  C0DD F0 0F             BEQ  LFORUN
2370:  C0DF A9 00    LFORES  LDA  #0       ;RESET
2380:  C0E1 9D 49 C0          STA  SAWUP,X  ;DYNAMISCHE PARAMETER
2390:  C0E4 9D 4A C0          STA  SAWUP+1,X ;INITIALISIEREN
2400:  C0E7 9D 4B C0          STA  KURVE,X
2410:  C0EA 9D 4C C0          STA  KURVE+1,X
2420:  C0ED 60       LFORTS  RTS
2430:  C0EE 18       LFORUN  CLC
2440:  C0EF BD 49 C0          LDA  SAWUP,X
2450:  C0F2 7D 20 C0          ADC  LFOF,X
2460:  C0F5 9D 49 C0          STA  SAWUP,X
2470:  C0F8 85 FB             STA  MR
2480:  C0FA BD 4A C0          LDA  SAWUP+1,X
2490:  C0FD 7D 21 C0          ADC  LFOF+1,X
2500:  C100 9D 4A C0          STA  SAWUP+1,X
2510:  C103 85 FC             STA  MR+1     ;SAWUP=SAWUP+LFOF
                ; KURVENFORM ERMITTELN
2530:  C105 BD 24 C0          LDA  LFOC,X
2540:  C108 29 18             AND  #$18
2550:  C10A F0 27             BEQ  TRIAN
2560:  C10C C9 08             CMP  #$08
2570:  C10E F0 50             BEQ  LFOMUL   ;SAWUP, WENIG ZU TUN
2580:  C110 C9 10             CMP  #$10
2590:  C112 F0 40             BEQ  SAWDOWN
                ; SQUARE
                ; KURVE = + LFOA(8) * 2**7 , FALLS SAWUP < LFOP
                ; KURVE = - LFOA(8) * 2**7 , SONST
2630:  C114 A5 FC             LDA  MR+1
2640:  C116 DD 22 C0          CMP  LFOP,X
2650:  C119 90 09             BCC  SOPOS
2660:  C11B 38       SONEG   SEC   ;LFOA NEGIEREN
2670:  C11C A9 00             LDA  #0
2680:  C11E FD 23 C0          SBC  LFOA,X
2690:  C121 38               SEC
2700:  C122 B0 04             BCS  SQ1      ;(IMMER)
2710:  C124 18       SOPOS   CLC
2720:  C125 BD 23 C0          LDA  LFOA,X
2730:  C128 6A       SQ1     ROR  A        ;ARTITHMETISCHER RECHTS-SHIFT
2740:  C129 9D 4C C0          STA  KURVE+1,X
2750:  C12C A9 00             LDA  #0
2760:  C12E 6A                ROR  A
2770:  C12F 9D 4B C0          STA  KURVE,X
2780:  C132 60                RTS
                ; TRIAN
                ; BERECHNE MAX(SAWUP, -SAWUP-1) * 2 - 2**15
2810:  C133 A5 FC    TRIAN   LDA  MR+1
2820:  C135 10 13            BPL  TRPOS
2830:  C137 A5 FB    TRNEG   LDA  MR
2840:  C139 49 FF            EOR  #$FF
2850:  C13B 0A               ASL  A
2860:  C13C 85 FB            STA  MR
2870:  C13E A5 FC            LDA  MR+1
2880:  C140 49 FF            EOR  #$FF
2890:  C142 2A               ROL  A        ;(-SAWUP-1)*2
2900:  C143 49 80            EOR  #$80     ;-2**15
2910:  C145 85 FC            STA  MR+1
2920:  C147 4C 60 C1          JMP  LFOMUL
2930:  C14A 06 FB    TRPOS   ASL  MR
2940:  C14C 2A               ROL  A        ;SAWUP*2
2950:  C14D 49 80            EOR  #$80     ;-2**15
2960:  C14F 85 FC            STA  MR+1
2970:  C151 4C 60 C1          JMP  LFOMUL
                ; SAWDOWN
                ; BERECHNE -SAWUP-1
3000:  C154 A5 FB    SAWDOWN LDA  MR
3010:  C156 49 FF            EOR  #$FF
3020:  C158 85 FB            STA  MR
3030:  C15A A5 FC            LDA  MR+1
3040:  C15C 49 FF            EOR  #$FF
3050:  C15E 85 FC            STA  MR+1
                ; LFOMUL
                ; ZWEIERKOMPLEMENT-KURVENFORM IN MR(16)
                ; MIT LFOA(8) MULTIPLIZIEREN
3090:  C160 BD 23 C0 LFOMUL   LDA  LFOA,X
3100:  C163 85 FD             STA  MD
3110:  C165 20 A6 C0          JSR  MULS
3120:  C168 A6 FE             LDX  LFONR
3130:  C16A A5 FB             LDA  MR
3140:  C16C 9D 4B C0          STA  KURVE,X
3150:  C16F A5 FC             LDA  MR+1
3160:  C171 9D 4C C0          STA  KURVE+1,X
3170:  C174 60                RTS
                ;-------------------------------------------------------
                ; EG (ADSR) UM EINEN SCHRITT WEITERSCHALTEN
                ;-------------------------------------------------------
3210:  C175 AD 48 C0 EG       LDA  EGC
3220:  C178 29 06             AND  #$06
3230:  C17A C9 04             CMP  #$04     ;HOLD "?"
3240:  C17C F0 15             BEQ  EGRTS    ;DANN NICHTS ZU TUN
3250:  C17E C9 06             CMP  #$06     ;RUN "?"
3260:  C180 F0 12             BEQ  EGRUN
3270:  C182 A9 00    EGRES   LDA  #0       ;RESET
3280:  C184 8D 6C C0          STA  E        ;DYNAMISCHE PARAMETER
3290:  C187 8D 6D C0          STA  E+1      ;INITIALISIEREN
3300:  C18A 8D 6E C0          STA  EKURVE
3310:  C18D 8D 6F C0          STA  EKURVE+1
3320:  C190 8D 70 C0          STA  EPHASE
3330:  C193 60       EGRTS   RTS
3340:  C194 A9 01    EGRUN   LDA  #1       ;MASKE FUER BIT 0
3350:  C196 2C 48 C0          BIT  EGC      ;GATE "?"
3360:  C199 F0 52             BEQ  RELEASE
                ; ATTACK ODER DECAY
3380:  C19B 2C 70 C0          BIT  EPHASE
3390:  C19E D0 39             BNE  DECAY
3400:  C1A0 AD 6C C0 ATTACK   LDA  E
3410:  C1A3 49 FF             EOR  #$FF
3420:  C1A5 85 FB             STA  MR
3430:  C1A7 AD 6D C0          LDA  E+1
3440:  C1AA 49 FF             EOR  #$FF
3450:  C1AC 85 FC             STA  MR+1     ;MR(16)=2**15-1-E(16)
3460:  C1AE AD 43 C0          LDA  A
3470:  C1B1 0A                ASL  A        ;*2
3480:  C1B2 B0 05             BCS  ATTACK2  ;FALLS A)=128,DANN MR * 1
3490:  C1B4 85 FD             STA  MD
3500:  C1B6 20 75 C0          JSR  MULU     ;INC(16)=MR(16)*2*A(8)/2**8
3510:  C1B9 18       ATTACK2  CLC   ;MR(16)=INC(16)
3520:  C1BA AD 6C C0          LDA  E
3530:  C1BD 65 FB             ADC  MR
3540:  C1BF 8D 6C C0          STA  E
3550:  C1C2 85 FB             STA  MR
3560:  C1C4 AD 6D C0          LDA  E+1
3570:  C1C7 65 FC             ADC  MR+1
3580:  C1C9 8D 6D C0          STA  E+1
3590:  C1CC 85 FC             STA  MR+1     ;E(16)=E(16)+INC(16)
3600:  C1CE C9 FF             CMP  #$FF     ;EG(16))=$FF00 "?"
3610:  C1D0 90 47             BCC  EGMUL    ;<
3620:  C1D2 A9 01             LDA  #1
3630:  C1D4 8D 70 C0          STA  EPHASE   ;UEBERGANG ZU DECAY
3640:  C1D7 D0 40             BNE  EGMUL    ;(IMMER)
3650:  C1D9 AD 6C C0 DECAY    LDA  E
3660:  C1DC 85 FB             STA  MR
3670:  C1DE 38                SEC
3680:  C1DF AD 6D C0          LDA  E+1
3690:  C1E2 ED 45 C0          SBC  S
3700:  C1E5 85 FC             STA  MR+1     ;MR(16)=E(16)-S(8)*2**8
3710:  C1E7 AD 44 C0          LDA  D
3720:  C1EA 4C FF C1          JMP  DECREL
3730:  C1ED A9 00    RELEASE  LDA  #0
3740:  C1EF 8D 70 C0          STA  EPHASE   ;BEI GATE=1 WIEDER ATTACK
3750:  C1F2 AD 6C C0          LDA  E
3760:  C1F5 85 FB             STA  MR
3770:  C1F7 AD 6D C0          LDA  E+1
3780:  C1FA 85 FC             STA  MR+1     ;MR(16)=E(16)
3790:  C1FC AD 46 C0          LDA  R
3800:  C1FF 85 FD    DECREL   STA  MD       ;GEMEINSAMER TEIL FUER DECAY UND RELEASE
3810:  C201 20 75 C0          JSR  MULU     ;MR(16)=DEC(16)
3820:  C204 38                SEC
3830:  C205 AD 6C C0          LDA  E
3840:  C208 E5 FB             SBC  MR
3850:  C20A 8D 6C C0          STA  E
3860:  C20D 85 FB             STA  MR
3870:  C20F AD 6D C0          LDA  E+1
3880:  C212 E5 FC             SBC  MR+1
3890:  C214 8D 6D C0          STA  E+1
3900:  C217 85 FC             STA  MR+1
3910:  C219 AD 47 C0 EGMUL    LDA  EGA      ;HUELLKURVE SKALIEREN
3920:  C21C 85 FD             STA  MD
3930:  C21E 20 75 C0          JSR  MULU     ;MR(16)=E(16)*EGA(8)/2**8
3940:  C221 A9 08             LDA  #$08     ;MASKE FUER BIT 3
3950:  C223 2C 48 C0          BIT  EGC      ;+/- "?"
3960:  C226 F0 10             BEQ  EGPLUS
3970:  C228 38       EGMINUS  SEC   ;EKURVE NEGIEREN
3980:  C229 A9 00             LDA  #0
3990:  C22B E5 FB             SBC  MR
4000:  C22D 8D 6E C0          STA  EKURVE
4010:  C230 A9 00             LDA  #0
4020:  C232 E5 FC             SBC  MR+1
4030:  C234 8D 6F C0          STA  EKURVE+1
4040:  C237 60                RTS
4050:  C238 A5 FB    EGPLUS   LDA  MR       ;EKURVE(16)=MR(16)
4060:  C23A 8D 6E C0          STA  EKURVE
4070:  C23D A5 FC             LDA  MR+1
```

```
4080:  C23F 8D 6F C0          STA  EKURVE+1
4090:  C242 60                RTS
 ;-----------------------------------------------------------
 ; SUMMIERE MODULATIONSBEITRAEGE GEMAESS EINER KSV-ZEILE
 ; DAS PROGRAMM ERWARTET EIN KSV-BYTE IN AKKU A
 ; UND LIEFERT MODULATIONS-SUMME IN MR(16) AB
 ;-----------------------------------------------------------
4150:  C243 49 FF     SUMMOD  EOR  #$FF       ;KSV-BYTE INVERTIEREN
4160:  C245 85 FF             STA  TEMP
4170:  C247 A9 00             LDA  #0
4180:  C249 85 FB             STA  MR
4190:  C24B 85 FC             STA  MR+1
4200:  C24D A0 08             LDY  #8         ;SCHLEIFENZAEHLER
4210:  C24F 46 FF     SUMLOOP LSR  TEMP
4220:  C251 B0 11             BCS  SUMNEXT    ;BEI 1 NICHTS SUMMIEREN
4230:  C253 AA                TAX  ;ADRESSVERSATZ FUER LFO-BLOECKE
4240:  C254 A5 FB             LDA  MR
4250:  C256 7D 4B C0          ADC  KURVE,X
4260:  C259 85 FB             STA  MR
4270:  C25B A5 FC             LDA  MR+1
4280:  C25D 7D 4C C0          ADC  KURVE+1,X
4290:  C260 85 FC             STA  MR+1
4300:  C262 8A                TXA
4310:  C263 38                SEC
4320:  C264 69 04     SUMNEXT ADC  #4         ;+5
4330:  C266 88                DEY
4340:  C267 D0 E6             BNE  SUMLOOP
4350:  C269 60                RTS
 ;-----------------------------------------------------------
 ; PORTAMENTO-EINZELSCHRITT FUER STIMME N
 ; PROGRAMM ERWARTET 7 * N (N=0,1,2) IN STINR
 ;-----------------------------------------------------------
4400:  C26A A6 FE     PORT    LDX  STINR
4410:  C26C BD 04 C0          LDA  PORTA,X
4420:  C26F D0 0D             BNE  PORTRUN
4430:  C271 BD 00 C0          LDA  F,X        ;PORTA=0, FREQUENZ UEBERNEHMEN
4440:  C274 9D 05 C0          STA  FP,X
4450:  C277 BD 01 C0          LDA  F+1,X
4460:  C27A 9D 06 C0          STA  FP+1,X
4470:  C27D 60                RTS
4480:  C27E BD 01 C0 PORTRUN  LDA  F+1,X
4490:  C281 DD 06 C0          CMP  FP+1,X
4500:  C284 90 3A             BCC  PMINUS     ;F<FP
4510:  C286 D0 0B             BNE  PPLUS      ;F>FP
 ; GLEICHHEIT, LOW-BYTES VERGLEICHEN
4530:  C288 BD 00 C0          LDA  F,X
4540:  C28B DD 05 C0          CMP  FP,X
4550:  C28E 90 30             BCC  PMINUS     ;F<FP
4560:  C290 D0 01             BNE  PPLUS      ;F>FP
4570:  C292 60                RTS  ;F=FP, NICHTS ZU TUN
4580:  C293 38        PPLUS   SEC
4590:  C294 BD 00 C0          LDA  F,X
4600:  C297 FD 05 C0          SBC  FP,X
4610:  C29A 85 FB             STA  MR
4620:  C29C BD 01 C0          LDA  F+1,X
4630:  C29F FD 06 C0          SBC  FP+1,X
4640:  C2A2 85 FC             STA  MR+1       ;DIF(16)=F(16)-FP(16)
4650:  C2A4 BD 04 C0          LDA  PORTA,X
4660:  C2A7 85 FD             STA  MD
4670:  C2A9 20 75 C0          JSR  MULU       ;INC(16)=DIF(16)*PORTA(8)/2**8
4680:  C2AC A6 FE             LDX  STINR
4690:  C2AE 38                SEC
4700:  C2AF BD 05 C0          LDA  FP,X
4710:  C2B2 65 FB             ADC  MR
4720:  C2B4 9D 05 C0          STA  FP,X
4730:  C2B7 BD 06 C0          LDA  FP+1,X
4740:  C2BA 65 FC             ADC  MR+1
4750:  C2BC 9D 06 C0          STA  FP+1,X     ;FP(16)=FP(16)+INC(16)+1
4760:  C2BF 60                RTS
4770:  C2C0 38        PMINUS  SEC
4780:  C2C1 BD 05 C0          LDA  FP,X
4790:  C2C4 FD 00 C0          SBC  F,X
4800:  C2C7 85 FB             STA  MR
4810:  C2C9 BD 06 C0          LDA  FP+1,X
4820:  C2CC FD 01 C0          SBC  F+1,X
4830:  C2CF 85 FC             STA  MR+1       ;DIF(16)=FP(16)-F(16)
4840:  C2D1 BD 04 C0          LDA  PORTA,X
4850:  C2D4 85 FD             STA  MD
4860:  C2D6 20 75 C0          JSR  MULU       ;DEC(16)=DIF(16)*PORTA(8)/2**8
4870:  C2D9 A6 FE             LDX  STINR
4880:  C2DB 18                CLC
4890:  C2DC BD 05 C0          LDA  FP,X
4900:  C2DF E5 FB             SBC  MR
4910:  C2E1 9D 05 C0          STA  FP,X
4920:  C2E4 BD 06 C0          LDA  FP+1,X
4930:  C2E7 E5 FC             SBC  MR+1
4940:  C2E9 9D 06 C0          STA  FP+1,X     ;FP(16)=FP(16)-DEC(16)-1
4950:  C2EC 60                RTS
 ;-----------------------------------------------------------
 ; HAUPTPROGRAMM
 ; SCHALTE ALLE MODULATINSQUELLEN UM EINEN SCHRITT WEITER
 ; MODULIERE ALLE PARAMETER GEMAESS KREUZSCHIENENVERTEILER (KSV)
 ;-----------------------------------------------------------
 ; 7 LFOS WEITERSCHALTEN
5020:  C2ED A9 1E     MODUL   LDA  #30
5030:  C2EF 85 FE     LFOLOOP STA  LFONR
5040:  C2F1 20 D0 C0          JSR  LFO
5050:  C2F4 38                SEC
5060:  C2F5 A5 FE             LDA  LFONR
5070:  C2F7 E9 05             SBC  #5
5080:  C2F9 10 F4             BPL  LFOLOOP
 ; EG (ADSR) WEITERSCHALTEN
5100:  C2FB 20 75 C1          JSR  EG
 ;
 ; 3 STIMMEN BEARBEITEN
 ;
5140:  C2FE A9 02             LDA  #2
5150:  C300 A2 0E             LDX  #14
5160:  C302 85 9B             STA  ZAEHLER
5170:  C304 86 FE             STX  STINR
 ; FREQUENZ MODULIEREN
5190:  C306 20 6A C2 FMOD     JSR  PORT       ;FP WEITERSCHALTEN
5200:  C309 A6 9B             LDX  ZAEHLER
5210:  C30B BD 18 C0          LDA  KSV,X
5220:  C30E D0 11             BNE  FMOD1
 ; KEINE FREQUENZMODULATION, PARAMETER UEBERNEHMEN
5240:  C310 A6 FE             LDX  STINR
5250:  C312 BD 05 C0          LDA  FP,X
5260:  C315 9D 00 D4          STA  SID,X
5270:  C318 BD 06 C0          LDA  FP+1,X
5280:  C31B 9D 01 D4          STA  SID+1,X
5290:  C31E 4C 41 C3          JMP  PMOD
5300:  C321 20 43 C2 FMOD1    JSR  SUMMOD     ;LIEFERT MODULATIONSWERT IN MR(16)
5310:  C324 A6 FE             LDX  STINR
5320:  C326 BD 06 C0          LDA  FP+1,X
5330:  C329 85 FD             STA  MD
5340:  C32B 20 A6 C0          JSR  MULS       ;MODULATIONSWERT MIT FP HIGH SKALIEREN
5350:  C32E A6 FE             LDX  STINR
5360:  C330 18                CLC
5370:  C331 BD 05 C0          LDA  FP,X
5380:  C334 65 FB             ADC  MR
5390:  C336 9D 00 D4          STA  SID,X
5400:  C339 BD 06 C0          LDA  FP+1,X
5410:  C33C 65 FC             ADC  MR+1
5420:  C33E 9D 01 D4          STA  SID+1,X
 ; PULSWEITE MODULIEREN
5440:  C341 A6 9B     PMOD    LDX  ZAEHLER
5450:  C343 BD 18 C0          LDA  KSV+3,X
5460:  C346 D0 11             BNE  PMOD1
 ; KEINE PW-MODULATION, PARAMETER UEBERNEHMEN
5480:  C348 A6 FE             LDX  STINR
5490:  C34A BD 02 C0          LDA  PW,X
5500:  C34D 9D 02 D4          STA  SID+2,X
5510:  C350 BD 03 C0          LDA  PW+1,X
5520:  C353 9D 03 D4          STA  SID+3,X
5530:  C356 4C B1 C3          JMP  NEXTSTI
5540:  C359 20 43 C2 PMOD1    JSR  SUMMOD     ;LIEFERT MODULATIONSWERT NACH MR(16)
 ; MR(12)=MR(16)/2**4, ERGIBT 12-BIT ZWEIERKOMPLEMENTGROESSE
5560:  C35C 46 FC             LSR  MR+1
5570:  C35E 66 FB             ROR  MR
5580:  C360 46 FC             LSR  MR+1
5590:  C362 66 FB             ROR  MR
5600:  C364 46 FC             LSR  MR+1
5610:  C366 66 FB             ROR  MR
5620:  C368 46 FC             LSR  MR+1
5630:  C36A 66 FB             ROR  MR
5640:  C36C A6 FE             LDX  STINR
5650:  C36E 18                CLC
5660:  C36F BD 02 C0          LDA  PW,X
5670:  C372 65 FB             ADC  MR
5680:  C374 9D 02 D4          STA  SID+2,X
5690:  C377 BD 03 C0          LDA  PW+1,X
5700:  C37A 65 FC             ADC  MR+1
5710:  C37C 29 0F             AND  #$0F       ;BIT 7-4 AUSBLENDEN
5720:  C37E 9D 03 D4          STA  SID+3,X
5730:  C381 8A        NEXTSTI TXA  ;(STINR)
5740:  C382 38                SEC
5750:  C383 E9 07             SBC  #7
5760:  C385 85 FE             STA  STINR
5770:  C387 C6 9B             DEC  ZAEHLER
5780:  C389 30 03             BMI  FILMOD
5790:  C38B 4C 06 C3          JMP  FMOD       ;NAECHSTE STIMME
 ; FILTERFREQUENZ MODULIEREN, NUR HIGH-BYTE
5810:  C38E AD 15 C0 FILMOD   LDA  FILT       ;LOW-BYTE UEBERNEHMEN
5820:  C391 8D 15 D4          STA  SID+21
5830:  C394 AD 1E C0          LDA  KSV+6
5840:  C397 D0 09             BNE  FILMOD1
 ; KEINE FILTERMODULATION, PARAMETER UEBERNEHMEN
5860:  C399 AD 16 C0          LDA  FILT+1
5870:  C39C 8D 16 D4          STA  SID+22
5880:  C39F 4C AE C3          JMP  LAUTMOD
5890:  C3A2 20 43 C2 FILMOD1  JSR  SUMMOD     ;LIEFERT MODULATIONSWERT IN MR(16)
5900:  C3A5 18                CLC
5910:  C3A6 AD 16 C0          LDA  FILT+1
5920:  C3A9 65 FC             ADC  MR+1
5930:  C3AB 8D 16 D4          STA  SID+22
 ; LAUTSTAERKE MODULIEREN
 ; NUR DIE 4 OBEREN BITS VON MR+1 TRAGEN DAZU BEI
5960:  C3AE AD 1F C0 LAUTMOD  LDA  KSV+7
5970:  C3B1 D0 07             BNE  LAUMOD1
 ; KEINE LAUTSTAERKEMODULATION, PARAMETER UEBERNEHMEN
5990:  C3B3 AD 17 C0          LDA  MODLAUT
6000:  C3B6 8D 18 D4          STA  SID+24
6010:  C3B9 60                RTS
6020:  C3BA 20 43 C2 LAUMOD1  JSR  SUMMOD     ;LIEFERT MODULATIONSWERT IN MR(16)
6030:  C3BD AD 17 C0          LDA  MODLAUT
6040:  C3C0 29 F0             AND  #$F0       ;MODUS (BIT 7-4) EXTRAHIEREN
6050:  C3C2 85 FF             STA  TEMP
6060:  C3C4 A5 FC             LDA  MR+1
6070:  C3C6 4A                LSR  A
6080:  C3C7 4A                LSR  A
6090:  C3C8 4A                LSR  A
6100:  C3C9 4A                LSR  A          ;A(4)=MR+1(8)/2**4
6110:  C3CA 18                CLC
6120:  C3CB 6D 17 C0          ADC  MODLAUT
6130:  C3CE 29 0F             AND  #$0F       ;BIT 7-4 AUSBLENDEN
6140:  C3D0 05 FF             ORA  TEMP       ;MODUS EINBLENDEN
6150:  C3D2 8D 18 D4          STA  SID+24
6160:  C3D5 60                RTS
 ;-----------------------------------------------------------
 ; CIA#1 TIMER A ABFRAGEN, LOW-BYTE IN A, HIGH-BYTE IN X
 ;-----------------------------------------------------------
6200:  C3D6 AD 04 DC TIME     LDA  $DC04      ;TIMER A LOW
6210:  C3D9 AE 05 DC          LDX  $DC05      ;TIMER A HIGH
6220:  C3DC C9 04             CMP  #4
6230:  C3DE B0 01             BCS  TIME1
 ; TA LOW < 4, UNTERLAUF NACH TA HIGH KORRIGIEREN
6250:  C3E0 E8                INX
6260:  C3E1 60        TIME1   RTS
 ;-----------------------------------------------------------
 ; ERWEITERTES INTERUPTPROGRAMM
 ;-----------------------------------------------------------
6300:  C3E2 20 D6 C3 INTRPT   JSR  TIME       ;STARTZEIT LESEN
6310:  C3E5 8D 73 C0          STA  ZEIT1      ;UND FESTHALTEN
6320:  C3E8 8E 74 C0          STX  ZEIT1+1
6330:  C3EB 20 ED C2          JSR  MODUL      ;MODULATIONSSCHRITT
6340:  C3EE 20 D6 C3          JSR  TIME       ;ENDZEIT LESEN
6350:  C3F1 85 FF             STA  TEMP       ;DIFFERENZ BERECHNEN
6360:  C3F3 38                SEC
6370:  C3F4 AD 73 C0          LDA  ZEIT1
6380:  C3F7 E5 FF             SBC  TEMP
6390:  C3F9 8D 71 C0          STA  ZEIT
6400:  C3FC 86 FF             STX  TEMP
6410:  C3FE AD 74 C0          LDA  ZEIT1+1
6420:  C401 E5 FF             SBC  TEMP
6430:  C403 8D 72 C0          STA  ZEIT+1
6440:  C406 4C 31 EA          JMP  $EA31      ;KERNAL-SYSTEMINTERUPTROUTINE
 ;-----------------------------------------------------------
 ; INTERUPTVEKTOR UMSTELLEN (MODULATOR EINSCHALTEN)
 ;-----------------------------------------------------------
6480:  C409 78        START   SEI
6490:  C40A A9 E2             LDA  #<INTRPT
6500:  C40C 8D 14 03          STA  $0314
6510:  C40F A9 C3             LDA  #>INTRPT
6520:  C411 8D 15 03          STA  $0315
6530:  C414 58                CLI
6540:  C415 60                RTS
 ;-----------------------------------------------------------
 ; INTERUPTVEKTOR ZURUECKSTELLEN (MODULATOR AUSSCHALTEN)
 ;-----------------------------------------------------------
6580:  C416 78        AUS     SEI
6590:  C417 A9 31             LDA  #$31
6600:  C419 8D 14 03          STA  $0314
6610:  C41C A9 EA             LDA  #$EA
6620:  C41E 8D 15 03          STA  $0315
6630:  C421 58                CLI
6640:  C422 60                RTS
0C075-C423

READY.
```

Dokumentiertes Assemblerlisting von »Modulator« (Schluß)