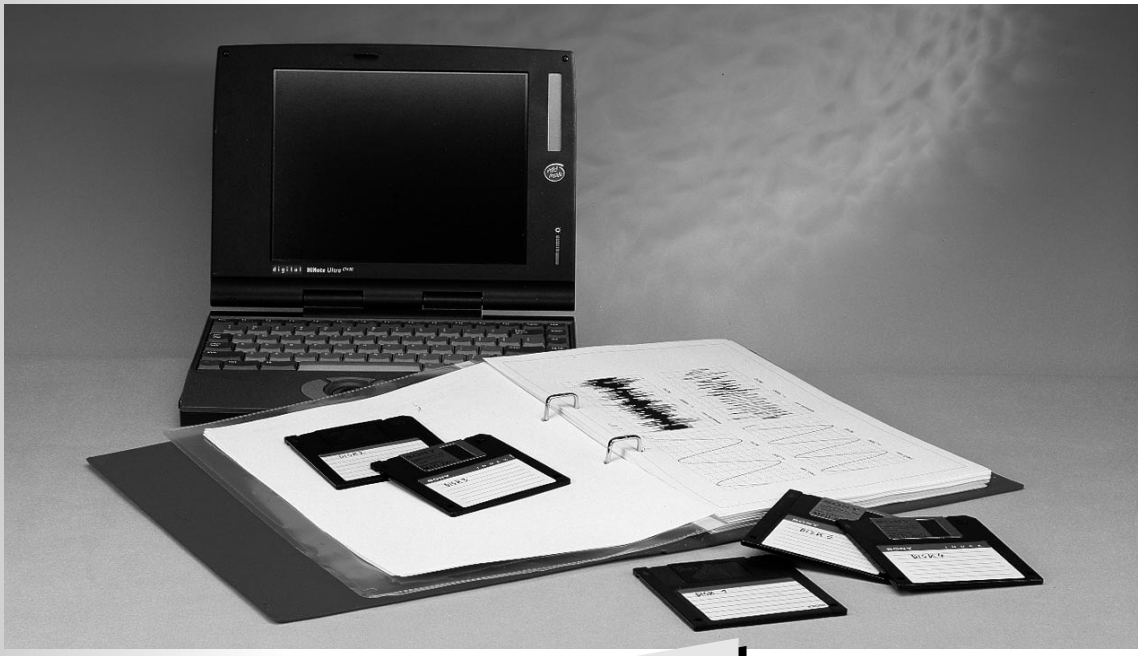
Digital Signal Processing

Kurs Teil 1: Einführung und erste Experimente



Die Idee, einen Kurs über Signalverarbeitung zu veröffentlichen, hatte der Autor dieses Kurses schon lange. Doch erst jetzt ist für viele Elektor-Leser die geeignete Hardware verfügbar, nämlich der PC mit Soundkarte und CD-ROM. Dadurch ist ein Einstieg ohne den Einsatz spezieller und komplizierter Signalprozessor-Hardware möglich.

Highlights

- Alle Beispiele direkt ausführbar
- Viele Wave-Dateien zur Analyse
- Programmbibliothek für viele Anwendungen
- PASCAL-Quellcodes aller Programme auf der CD
- Batch-Dateien für komplexe Experimente
- Simulation komplexer Übertragungssysteme
- Kein Bau von Hardware nötig

Dieser Kurs soll dem Leser wichtige Grundlagen der digitalen Signalverarbeitung in einfachen Beispielen und Experimenten zugänglich machen. Dabei wird auf spezielle Hardware verzichtet, zum Einsatz kommt ein ganz normaler PC (der nicht einmal besonders schnell sein muß) und eine Soundkarte.

Der Kurs kann auf mehreren Niveaus verfolgt werden. Am einfachsten ist es, die beschriebenen Experimente nachzuvollziehen, ohne die Grundlagen genau verstehen zu wollen. Die nächste Option besteht darin, anhand der mitgelieferten Programme eigene Experimente durchzuführen. Dazu gibt der Kurs Anregungen. Wieder einen Schritt weiter geht der Leser, der anhand der Programm-Quelltexte und der Erklärungen die Arbeitsweise der Programme verstehen will. Er kann anschließend selbst Programme zur Signalverarbeitung schreiben und auch eigene Vorstellungen realisieren. Zuletzt kann er auch versuchen, die hier vorgestellten Verfahren und Programme

auf einen "realen" Signalprozessor zu übertragen. Dies ist aber erfahrenen Programmierern vorbehalten, denn man stößt auf viele unerwartete Probleme, zum Beispiel Skalierungen, Integerarithmetik, Initialisierungen von Hardwarekomponenten, die nicht Thema des Kurses sind, da sie auch nicht zur eigentlichen Signalverarbeitung gehören. Wer dennoch diesen Weg beschreiten will, kann sicher nicht auf das Studium weitergehender Literatur verzichten.
Die Themen des Kurses sind in Tabelle 1 dargestellt. Viele der Verfahren der Signalverarbeitung beruhen auf mathematischen Konzepten, auf die wir aber verzichten wollen, soweit dies möglich ist. Manchmal kommen wir aber nicht darum herum. Wem das zu kompliziert ist, kann den Kurs durchaus verfolgen, ohne diese Abschnitte zu verstehen !

Die Software

Programme
In diesem Kursus werden zahlreiche Programme zur Verarbeitung von Signalen vorgestellt und verwendet. Alle notwendigen Programme sind als .EXE Dateien auf einer Kurs-CD vorhanden, die ab Februar beim Verlag erhältlich ist (EPS 986004-1). Mit diesen Programmen kann der Anwender zahlreiche Aufgaben selbst durchführen sowie eigene Daten analysieren und darstellen. In Tabelle 2 sind die Programme einmal aufgelistet.

Installation
Die Software wird installiert, wie auf der Kurs-CD in der Datei INSTALL.DOC beschrieben. Im Grunde werden einfach alle Programme in ein eigenes Arbeitsverzeichnis kopiert und für die einzelnen Kursteile die langen WAV-Dateien bei Bedarf hinzugefügt. Auf der CD befinden sich auch noch einige Hinweise, um sich mit der Software besser vertraut zu machen, die hier der Kürze halber fehlen.

Quellcodes
Die meisten Programme sind in TURBO-PASCAL 5.0 geschrieben. Der Quellcode liegt mit auf der CD vor, so daß der Leser die Programme selbst erweitern und verändern kann. Manchmal wird anhand der Quellcodes innerhalb des Kurses die Arbeitsweise der Programme erläutert, um die besprochenen Konzepte zu konkretisieren. Durch die Verwendung einer Bibliothek zur Signalverarbeitung können diese Programme sehr kurz und einfach ausfallen, so daß der Leser sich auf die zum Verständnis wesentlichen Dinge konzentrieren kann.

Geplante Themen des Kurses

- ✓ *Wave-Dateien*
- ✓ *Abtasttheorem*
- ✓ *Signalgeneratoren*
- ✓ *Filteranalyse mit Frequenzsweep*
- ✓ *Diskrete Fouriertransformation*
- ✓ *Fensterung*
- ✓ *Echoerzeugung*
- ✓ *Filterdesign*
- ✓ *Filteranalyse mit Rauschen*
- ✓ *Fouriersynthese*
- ✓ *Frequenzmodulation*
- ✓ *Quadraturverfahren*
- ✓ *RDS-Modulation*
- ✓ *Abgetastete Signale*
- ✓ *Aliasing, Downsampling*
- ✓ *Rekursive Tiefpaß- und Bandpaßfilter*
- ✓ *Spektrumanalyser*
- ✓ *FFT*
- ✓ *Sprungantwort und Frequenzgang von Filtern*
- ✓ *FIR-Filter*
- ✓ *Rauschsignale*
- ✓ *periodische Signale*
- ✓ *Amplitudenmodulation/demodulation*
- ✓ *Phasenmodulation*
- ✓ *Funkfernschreiben*

Experiment-Dateie
Viele Experimente bestehen darin, eine Reihe von Programmen in vorgegebener Weise hintereinander auszuführen. Zuerst werden zum Beispiel Signale erzeugt, anschließend verändert (mit mehreren Filtern) und dann angezeigt. So kann beispielsweise das vollständige Verhalten eines SSB-Senders mit der Datei SSB EMPFÄNGER nachgebildet werden. Da an einem Experiment oft eine ganze Anzahl von Dateien beteiligt ist (etwa Ablauf des Experiments, Filter-Definitionen, Voreinstellungen für Oszilloskop und Spektrumsanalyser), haben wir eine einfache Möglichkeit geschaffen, alles zu steuern, nämlich den

Simple-Pre-Prozessor SPP
Mit Hilfe dieses Pre-Prozessor-Programms SPP kann man komplette Experimentabläufe einfach in einer Datei festlegen. Im Prinzip macht der

Programme auf der Kurs-CD (vorläufig)

Signalgeneratoren	
SIN0	*Generator für sinusförmige Signale*
SIN1	*Generator für sinusförmige Signale*
PULSE1	*Generator für Einheitspuls*
STEP1	*Stufenform-Generator*
NOISE1	*Generator für Weißes Rauschen*
FMSWEEP1	*Sweep-Generator*
MUSICG1	*Tonleiter-Generator*
Filter	
SINFIL1	*einfaches Bandpaß-Filter*
BANDP1	*einfaches Bandpaß-Filter*
BUTTER1	*digitales Butterworth-Filter*
LP1	*einfacher Tiefpaß*
ECHO1	*Echogenerator*
FIRFIL1	*Universelles FIR-Filter*
SPECFIL1FIR	*Filter Synthese Programm*
Modulation, Demodulation, Mathematik	
DWNSMPL1	*Downsampling*
SUM1	*gewichtete Summe zweier Signale*
MUL1	*Produkt von Signalen (Mischerfunktion etc.)*
AMGEN1	*Amplitudenmodulation,synchrone Demodulation, Mischer*
FMGEN1	*Frequenzmodulator*
SCHMITT1	*Schmitt-Trigger Funktion*
SHORT1	*Signalausschnitt separieren*
RTTYRX1	*serielle Fernschreib-Daten dekodieren*
Analysatoren	
INFO1	*Allgemeine Information, Mittelwert, Energie von Signalen*
SCOPE1	*Mehrkanal-Oszilloskop*
SPEC1	*Mehrkanal-Spektrumsanalyser*
Diverses	
SHELP	*Hilfe-Funktion für die PASCAL/EXE Programme*
SPP	*Preprozessor für Experiment-Dateien*

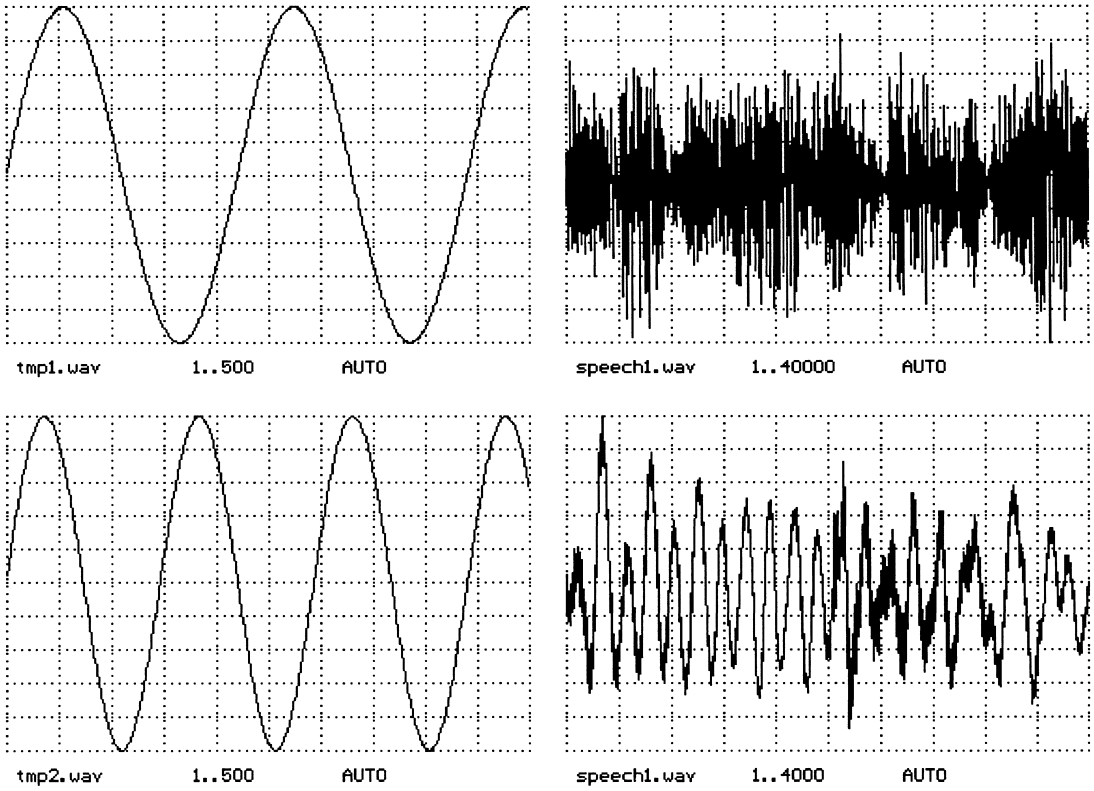


Bild 1. Erzeugte Signalformen werden wie im Oszilloskop dargestellt.

SPP dann nichts weiter, als aus dieser Datei aufgrund eingestreuter Kommandos mehrere Dateien zu erzeugen, die beim Experimentablauf notwendig sind. Die Experimente können dann mit einem einfachen Aufruf ausgeführt werden.
Um das in der Datei TEST3.SPP beschriebene Experiment auszuführen, muß man einfach
`DO TEST3.SPP <return>`
eingeben. Probieren Sie dies gleich einmal aus! Das Resultat ist in Bild 1 dargestellt. Genauso werden auch andere Simulationen gestartet.

SHELP-Funktion
Zu vielen Programmen kann man durch
SHELP Programmname <return>
eine Hilfe mit den wichtigsten Programmparametern auflisten lassen.
`SHELP SIN1.PAS <return>` bzw.
`SHELP SPP.PAS <return>`
klärt über den Sinusgenerator SIN1 beziehungsweise den SPP auf.

QUELLMATERIAL

Will man die verschiedenen Aufgaben der Signalverarbeitung "hautnah" selbst durchführen, benötigt man Signalmaterial, das man auch bearbeiten kann. Eine ganze Sammlung von Audio-Material ist auf der Kurs-CD vorhanden, um alle Experimente wirklich auszuführen. Es lassen sich aber auch selbst aufgenommene Signale verarbeiten. Alle Signale werden in Form von Wave-Dateien verarbeitet.
In unserem Kursus stellen Wave-Dateien (Endung .WAV unter DOS und Windows) das zentrale Mittel zum Austausch von Daten (Signalen) zwischen Programmen und der Außenwelt dar. Im Kopf (Header) der Wave-Datei werden verschiedene Parameter der Daten gespeichert. In unserem Kurs wird weitestgehend mit Wave-Dateien mit folgenden Parametern gearbeitet:

Sampling-Rate	*44100 oder 22050 oder 11025 Samples/s*
Bits/Sample	*16*
Format	*unkomprimiert*
Kanäle	*MONO*

Wiedergeben
Will man die Experimente, die in unserem Kurs durchgeführt werden, mit dem eigenen Ohr und nicht nur als Kurvenform auf dem Bildschirm verfolgen, so muß man über die Möglichkeit verfügen, Wave-Dateien abspielen zu können. Dazu gibt es eine Reihe von Programmen, meistens auch im Lieferumfang einer Soundkarte oder als Share- oder Freeware (im Internet). Zu bevorzugen ist ein unter DOS laufendes Programm.
Ist ein solches Programm aktiviert, sollte man einen Versuch durchführen: Versuchen Sie, die von der CD kopierte Datei SPEECH1.WAV abzuspielen. Es muß ein englischer Text mit Musik zu vernehmen sein

Aufnehmen
Will man eigene Signale aufnehmen (zum Beispiel aus einem Empfänger oder aus elektronischen Schaltungen), muß man ein Programm aktivieren, um Aufnahmen von Wave-Dateien mit der Soundkarte zu machen. Auch dies gehört zum Lieferumfang von Soundkarten oder ist als Shareware zu haben. Das Aufnehmen von Wave-Dateien ist im Kurs nicht unbedingt notwendig, da alle benötigten Dateien auf der Kurs-CD vorhanden sind. Trotzdem sollte man sich die Technik aneignen, wie man eine Wave-Datei aufnimmt. Dann kann man viele Meßaufgaben mit den Kursprogrammen erledigen und so den Einsatzbereich seines PCs stark erweitern. Wichtig ist, die Aufnahmeparameter so einzustellen, daß die Programme die Dateien verarbeiten können.

ERSTE EXPERIMENTE

Ein digitaler Sinusgenerator
Um zu zeigen, wie wir in diesem Kurs arbeiten wollen, gleich ein praktisches Beispiel. Auf der CD befindet sich des Programm SIN1.EXE zur Erzeugung einer Wave-Datei, die ein Sinussignal enthält. Das können wir gleich ausprobieren. Wir starten es mit
`SIN1 <return>`
Die Default-Parameter sorgen dafür, daß ein Sinussignal von 1000 Hz erzeugt wird, das zwei Sekunden lang dauert (44100 Samples bei 22050 Samples pro Sekunde). Die Amplitude ist 10000 (Spitzenwert). Das Signal wird

in der Datei SIN1.WAV gespeichert. Diese Dateien können wir nun abspielen. Damit haben wir bereits die Möglichkeit, für Testzwecke in der Elektronikwerkstatt ein Sinussignal zu erzeugen. Mit den Parametern des Programms können wir andere Sinussignale erzeugen. Um beispielsweise ein Sinussignal von 500 Hz mit 100.000 Abtastwerten bei einer Samplingrate von 11025/s und einer Amplitude von 5000 zu erzeugen, ruft man das Programm wie folgt auf (Abspeichern der Daten erfolgt in der Datei SIN2.WAV).

```
SIN1 \scale=5000 \f0=500
\n=100000 \fs=11025
\out=sin2.wav <return>
```

Probieren Sie, verschiedenste Sinussignale zu erzeugen und sich anzuhören. Wie das Programm arbeitet, wird später erläutert.

Erste Theorie

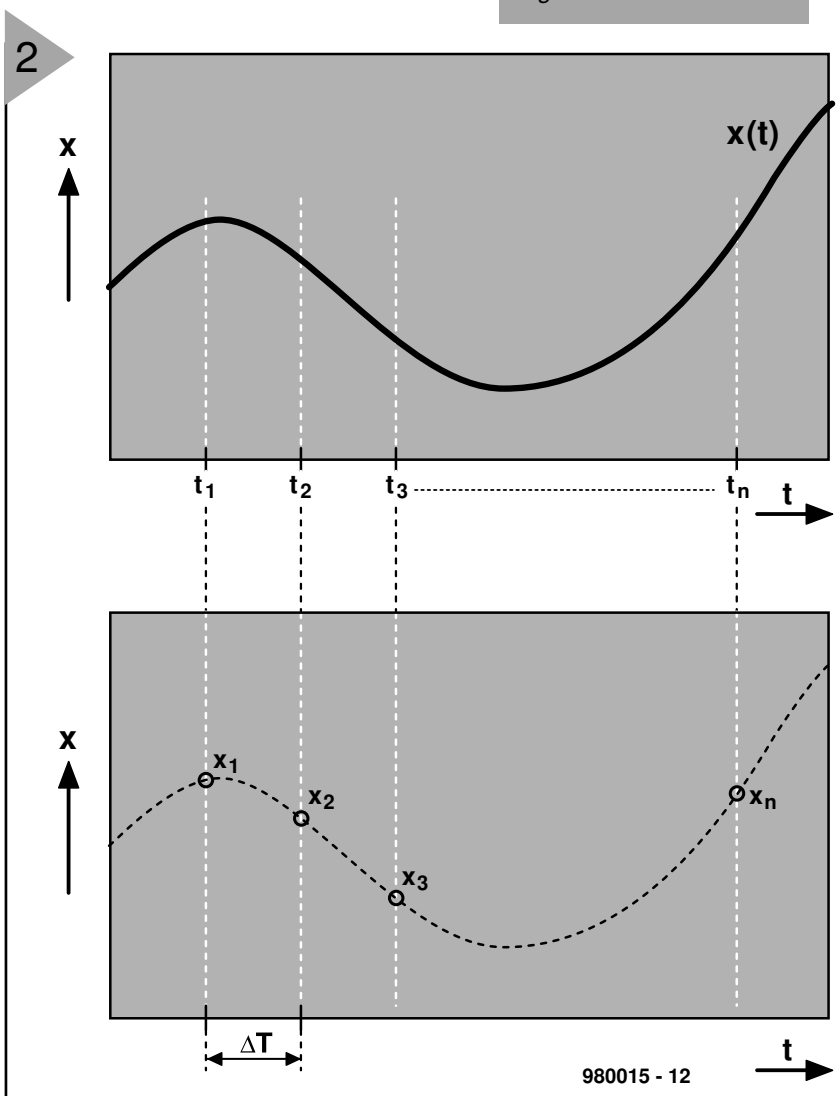
Nun geht es direkt mit ein wenig Theorie der digitalen Signalverarbeitung los. Wir kümmern uns darum, was bei der Abtastung und A/D-Wandlung von analogen Signalen passiert. In der analogen Technik kennt man normalerweise Signale x, etwa eine Spannung in einer elektrischen Schaltung, die zu jeder Zeit t einen Wert haben. Die Funktion der Spannung bezeichnet man mit x(t) (Bild 2).

Um Berechnungen mit Signalen anstellen zu können, tastet man das Signal ab. Dazu ermittelt man den Wert des Signals zu den Abtastzeitpunkten t_n. Dort hat das Signal den Wert $x(t_n)$, den wir von nun an mit x_n bezeichnen. In der Regel ist der Abstand zwischen zwei aufeinanderfolgenden Abtastwerten gleich. Die Zahl der Abtastungen pro Sekunde bezeichnen man als Sampling-Frequenz. Aus dem zeitkontinuierlichen Signal x(t) ist damit ein zeitdiskretes Signal geworden. Das Signal wird dargestellt als eine Folge von Zahlen. Die Temperaturangaben des Wetterdienstes sind ein Beispiel. Die Temperatur hat zu jedem Zeitpunkt einen Wert, Das Wetteramt teilt aber nur die Temperaturwerte zu bestimmten Zeiten mit.

In der nächsten Folge werden wir die Effekte bei Abtastung weiter besprechen und bereits in die digitale Filtertechnik einsteigen.

(980015-1)rg

Bild 2. Die Abtastung eines analogen Signals



Digital Signal Processing

Kursteil 2: Abtasten und digitale Filtertechnik

Nachdem wir in der letzten Folge das Abtasten von Signalen erklärt hatten, kommen wir diesmal darauf zurück, um die damit verbundenen Effekte kennenzulernen. Danach steigen wir dann in die digitale Filtertechnik ein.

Kennt man von einem abgetasteten Signal nur die Werte zu den Abtastzeiten, gibt das Abtasttheorem Auskunft, ob alle Signalinformationen in den abgetasteten Werten enthalten sind oder nicht:

Enthält ein Signal nur Signalanteile mit Frequenzen kleiner als f_{max}, so reichen die Abtastwerte zur Rekonstruktion des Signals aus, sofern sie mit einer Abtastrate größer als $2 \cdot f_{max}$ gewonnen wurden.

Ein Beispiel soll eine Verletzung dieses Theorems demonstrieren, wenn ein Signal zu hohe Frequenzen für eine gegebene Abtastrate enthält. `MUSICG1 <return>` erzeugt eine Tonleiter ab 40 Hz. Es werden 60 Töne erzeugt, die jeweils einen Halbton ansteigen, so daß ein Bereich von fünf Oktaven überstrichen wird. Der höchste Ton hat eine Frequenz von etwa 14 kHz. Die Töne werden in `MUSIC1.WAV` mit einer Abtastrate von 44,1 kHz gespeichert, das Abtasttheorem ist also erfüllt, was sich problemlos durch Anhören verifizieren läßt.

Downsampling

Nun tasten wir das gerade generierte Signal erneut ab, allerdings mit nur 11025 kHz, also mit einem Viertel der ursprünglichen Abtastfrequenz. Dieser Vorgang wird Downsampling genannt und vom Programm `DWNSMPL1.EXE` durchgeführt. Es verwandelt dazu `MUSIC1.WAV` in die Datei `MUSIC2.WAV`. Den Downsampling-Faktor gibt man im Aufruf

```
DWNSMPL1   \inp=MUSIC1.WAV
\out=MUSIC2.WAV   \factor=4
<return>
```

an. Nun sind lediglich niedrigere Töne als 11025 Hz/2 = 5512,5 Hz vorhanden. Die darüberliegenden Töne werden nach dem Abtasttheorem nicht korrekt rekonstruiert und deshalb in völlig falsche Frequenzen umgesetzt. Dies bezeichnet man als *Aliasing*.

1

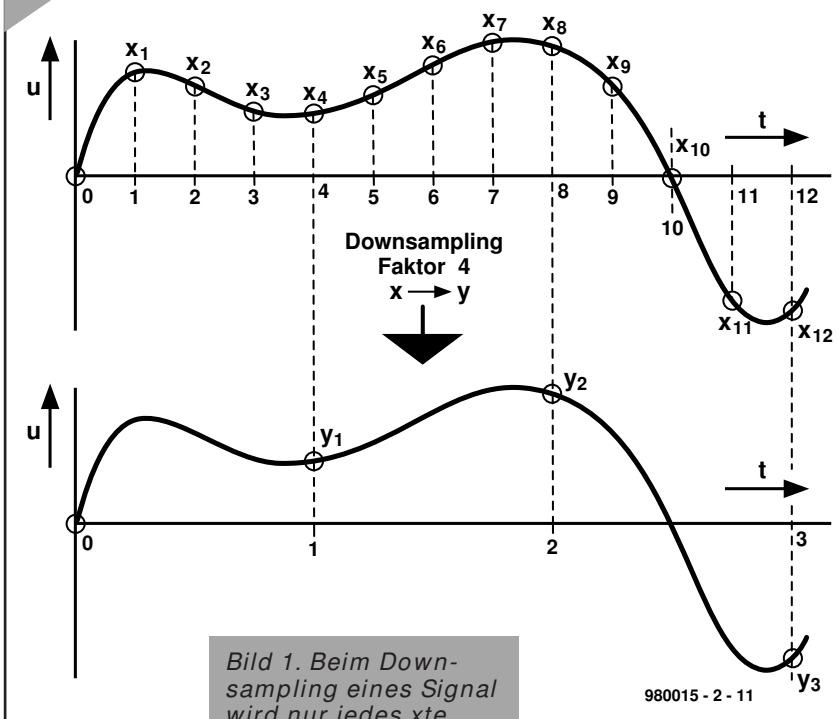


Bild 1. Beim Downsampling eines Signal wird nur jedes xte Sample genommen.

Alias-Frequenzen

Der Vorgang des Aliasing ist übrigens keineswegs willkürlich, sondern ziemlich einfach nachzuvollziehen. Wird ein sinusförmiges Signal mit der Frequenz $f_0 < f_s/2$ mit der Frequenz f_s abgetastet, so entstehen bestimmte Abtastwerte. Jedes Signal mit den Frequenzen $m \cdot f_s - f_0$ oder $m \cdot f_s + f_0$ ($m = 1, 2, 3, 4 \ldots$) erzeugt bis auf das Vorzeichen die gleichen Abtastwerte, die zu f_0 gehörenden Alias-Frequenzen (Bild 2). Nach dem Abtasten kann man Signale dieser Frequenzen nicht unterscheiden. Sie alle treten infolge der abgetasteten Werte mit Frequenz f_0 auf. Um das zu vermeiden, schaltet man vor einem A/D-Wandler ein Tiefpaßfilter, das die Alias-Frequenzen unterdrückt. Und damit wären wir beim Thema

TIEFPASSFILTERUNG

Dies muß nicht mit einem "gewöhnlichen" Filter geschehen, sondern läßt sich auch in einem digitalen Verfahren erledigen. Wir betrachten zuerst das analoge Tiefpaßfilter (Bild 3 und Bild 4) und versuchen, sein Verhalten digital nachzuvollziehen. Ein bißchen Mathematik läßt sich nicht vermeiden, obwohl es der Einfachheit halber nicht sehr "sauber" zugeht.

Während eines Abtastintervalls mit der Dauer $\Delta T = t_{k+1} - t_k$ ändert sich die Eingangsspannung u nur wenig, sondern behält während der Zeit den Anfangswert u_k. Auch die Ausgangsspannung wird sich nur wenig ändern, so daß durch den Widerstand R der fast konstante Strom $i = (u_k - v_k)/R$ fließt. Am Beginn des Abtastintervalls weist der Kondensator die Spannung v_k auf. Er wird Δt lang mit diesem Strom geladen und hat demnach die Spannung

$$v_{k+1} = v_k + i\,\Delta t/C$$
$$= v_k + (v_k - u_k)/RC \cdot \Delta t$$

Wir lösen nach u_{k+1} auf und erhalten

$v_{k+1} = r \cdot v_k + (1-r)u_k$ mit
$r = 1 - (\Delta t/RC)$

Das ist die Berechnungsvorschrift für unser erstes digitales Filter. Das Programm ist auf der CD-ROM unter dem Namen `LP1.EXE` zu finden, den Quellkode `LP1.PAS` kann man Bild 5 entnehmen.

Die Zeilen 1 bis 5 bilden den Programmkopf und vereinbaren, daß die Programmbibliothek `SIGLIB.PAS` verwendet werden soll. Die Zeilen 6 und 7 vereinbaren die notwendigen Variablen, während Zeile 10 die Initialisierung von `SIGLIB.PAS` aufruft. Die Zeilen 11 bis 14 besetzen die Parameter vor und gegebenenfalls (durch Prozeduren aus `SIGLIB.PAS`) mit aktuellen Parametern des Programmaufrufs. In den Zeilen 16 und 17 wird die Ein- und Ausgabe der WAV-Dateien aktiviert. Zeile 19 initialisiert den Filterwert. Die Schleife der Zeilen 20 bis 25 führt den eigentlichen Filtervorgang aus. In Zeile 22 wird der aktuelle Filterausgangswert in die Ausgabedatei geschrieben. Der eigentliche Filteralgorithmus ist in Zeile 23 enthalten. Die Anweisung in Zeile 26 schließt mit der bekannten Prozedur alle Dateien und damit das Programm.

Das Beispiel zeigt, daß Programme zur digitalen Signalverarbeitung nicht immer lang und undurchschaubar sein müssen. Den sonst vorhandenen Ballast nimmt uns in diesem Kurs die Programbibliothek `SIGLIB.PAS` ab. Zum Test des Filters bearbeiten wir die Datei `MUS1.WAV` (zunächst anhören !) von der CD-ROM. Dies geschieht durch den Aufruf

```
lp1  \r=0.995 \scale=10
\inp=mus1.wav \out=tmp.wav
<return>
```

Das entstehende Signal `tmp.wav` unterscheidet sich beim Hörtest deutlich vom Ursprung. Mit diesem Filter kann man nun etwas experimentieren, indem man zum Beispiel verschiedene Werte von `r` probiert. Größer als 1 darf man `r` allerdings nicht wählen, da das Filter sonst instabil wird.

Durch bloßes Zuhören kann man ein Filter natürlich nicht sehr präzise charakterisieren. Dazu benötigen wir ein paar Testsignale, um beispielsweise das Filterverhalten im Zeit- und Frequenzbereich zu analysieren. Dies kann am einfachsten durch die

PULSGENERATOREN

geschehen, die zum Werkzeugkasten auf der CD-ROM gehören. Der erste und einfachste mit der Bezeichnung `PULSE1.EXE` erzeugt einen sehr kurzen Impuls, der nur aus einem einzigen Abtastwert ungleich Null besteht, während alle anderen Abtastwerte Null sind. Die Position und Höhe des Pulses kann man über Parameter einstellen. Dieses elementare Signal ist sehr wichtig und wird bei späteren Experimenten oft benötigt, wie wir noch sehen werden.

Ein weiterer Impulsgenerator ist das Programm `STEP1.EXE`, das eine einfache Stufe mit vorgegebener Höhe und Position erzeugt. Als Beispiel führen wir den Aufruf `DO XLP1.SPP` aus. Die damit erzeugten Signale sind in Bild 6 dargestellt.

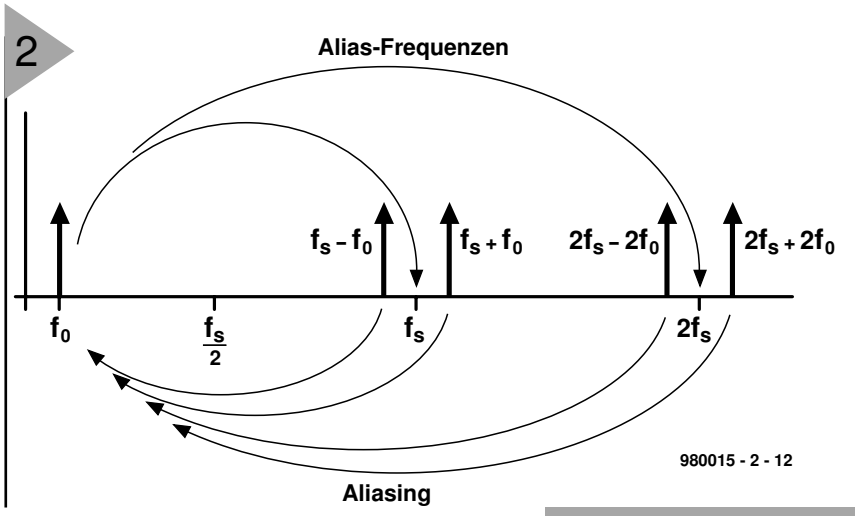
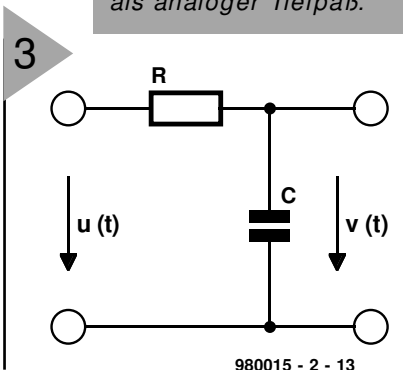


Bild 2. Die äquivalenten Frequenzen entstehen durch den Aliasing-Effekt.

Bild 3. Ein RC-Glied als analoger Tiefpaß.

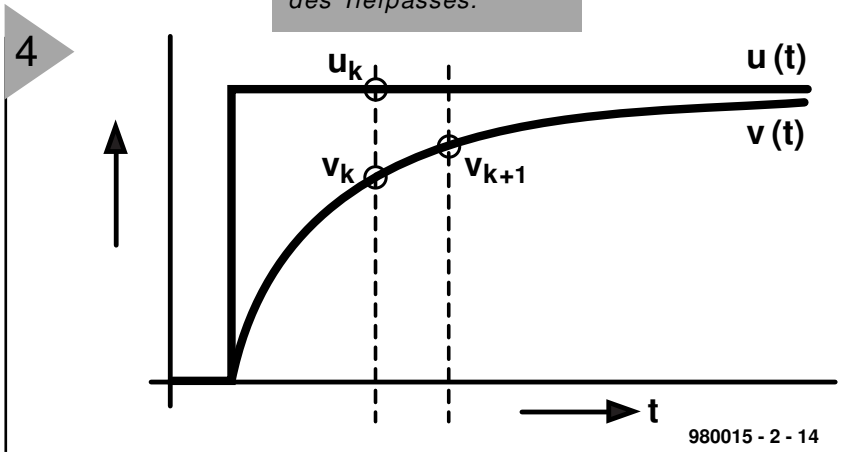


Oben wird die Reaktion des Filters `tmp1.wav` auf den Impuls `pulse1.wav` dargestellt, unten `tmp.wav`, die langsam ansteigende Antwort auf den Sprung `step1.wav`.

Dieses Experiment kann man wieder mit verschiedenen `r`-Werten durchführen. Was passiert zum Beispiel für `r = -0.9`?

Mit diesen Kurvenverläufen kann man das Verhalten eines Tiefpaßfilters aber nur ungenau charakterisieren. Eine weitere Möglichkeit ist ein Sweep-Generator,

Bild 4. Sprungantwort des Tiefpasses.



```
001 program lp1 ;
002 uses dos,crt,graph ;
003
004 {$I SIGLIB.PAS }
005
006 var k:int ;
007     y,scale,r:float ;
008
009 begin
010 start('simple lowpass') ;
011 scale:=1.0            ; set_par_real('\scale=',scale) ;
012 r:=0.95               ; set_par_real('\r=',r) ;
013 inp_fn:='pulse1.wav'  ; set_par_string('\inp=',inp fn) ;
014 out_fn:='tmp.wav'     ; set_par_string('\out=',out fn) ;
015
016 open_inp(inp_fn) ;
017 open_out(out_fn) ;
018
019 y:=0 ;
020 for k:=1 to nsamples do
021  begin
022  output(scale*y) ;
023  y:=r*y+(1-r)*input ;
024  if (k mod 2000)=0 then write('.') ;
025  end ;
026 stop ;
026 end.
```

Bild 5. Pascal-Quellkode des digitalen Filters LP1.

Was wäre ein Kurs über digitale Signalverarbeitung wert ohne die Möglichkeit, auch Spektren von Signalen anschauen zu können. Zu diesem Zweck gibt es auf der CD-ROM das Programm `SPEC1.EXE`, (Quellkode `SPEC1.PAS`), das Spektren von WAV-Dateien berechnen und anzeigen kann.

Zur Berechnung eines Spektrums benutzt `SPEC1.EXE` die sogenannte Diskrete Fouriertransformation (DFT), die aus `N` Werten `N/2` Amplitudenwerte ableitet, die zu den Frequenzen 0 bis $f_s/2$ gehören. In unserem Programm ist `N` = 4096. Diese Amplitudenwerte werden anschließend logarithmisch dargestellt, um einen großen Amplitudenbereich zu erhalten.

Das Experiment `XSPEC1.SPP` erzeugt zwei Signale mit den Frequenzen 193,7988 Hz (`tmp1.wav`) und 196,4905 Hz (`tmp2.wav`) mit jeweils 4096 Samples bei der Abtastrate von 44.100 Samples/s. Führt man die DFT an diesen Sinussignalen durch, erhält man die beiden in Bild 8 dargestellten Spektren.

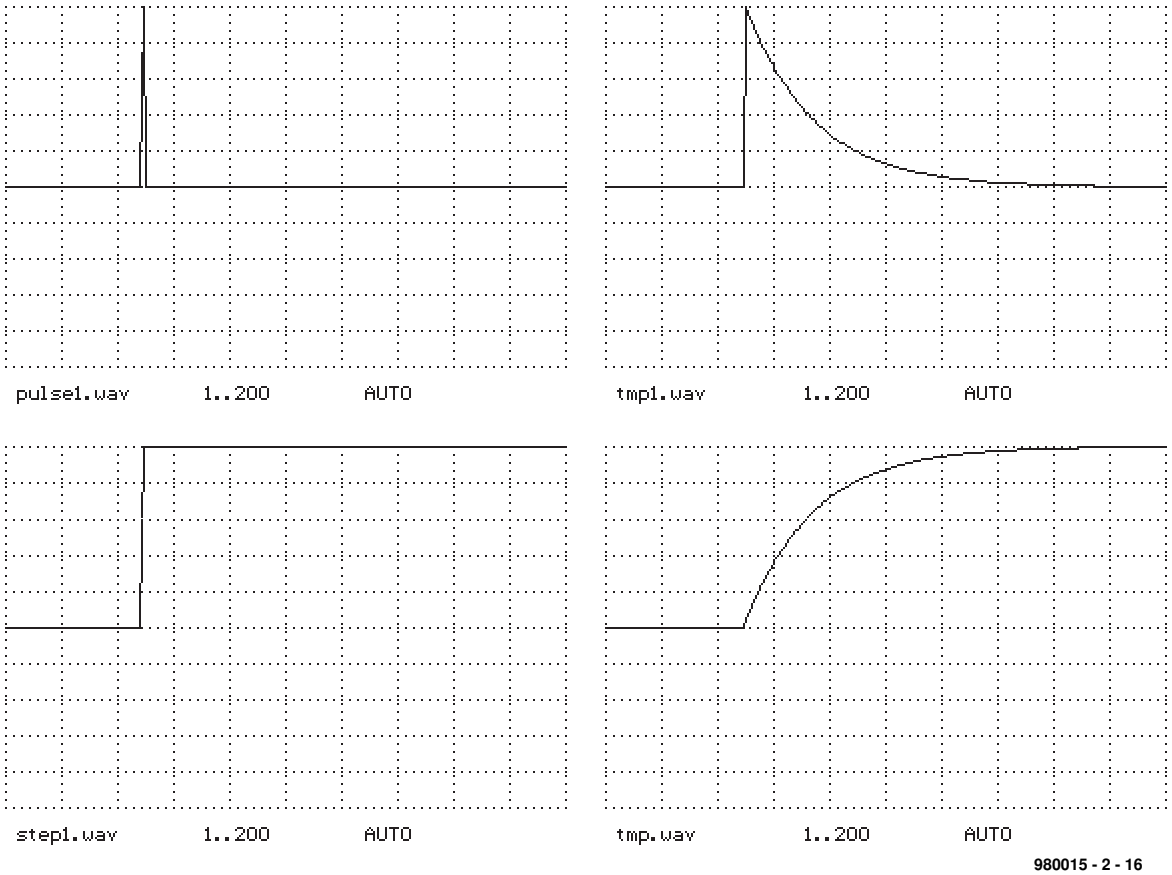
Wie man sieht, scheint in dem ersten Signal (`tmp1.wav`, links oben) nur eine Frequenz vorhanden zu sein. Das Spektrum ist wie erwartet schön linienförmig. Dagegen zeigt die Auswertung des zweiten Signals (`tmp2.wav` rechts oben) nur einen Peak von etwa 20 dB, das Spektrum fällt links und rechts von der Spitze

ebenfalls auf der CD-ROM vorhanden. Mit dem Aufruf von

```
DO XLP2.SPP <return>
```

wird ein Sweep-Signal von 1...1000 Hz erzeugt (oben in Bild 7) und auf das Tiefpaßfilter gelegt. Die Amplitude des Ausgangssignals (unten) wird mit steigender Frequenz kleiner. Eine weitere Möglichkeit ist es, Weißes Rauschen auf das Tiefpaßfilter zu geben und sich das Spektrum des Ausgangssignals anzusehen. Dazu muß man aber erst einmal die Möglichkeit kennenlernen, Spektren von Signalen zu berechnen und zu betrachten.

Bild 6. Impuls- und Sprungantwort des Tiefpasses.



7

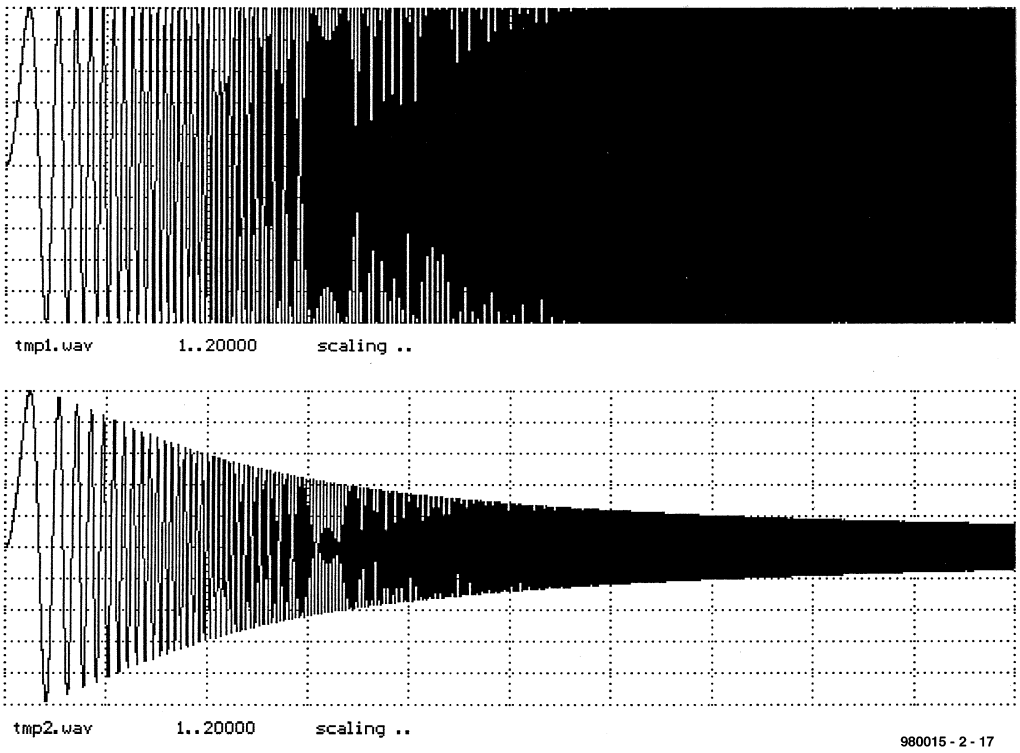
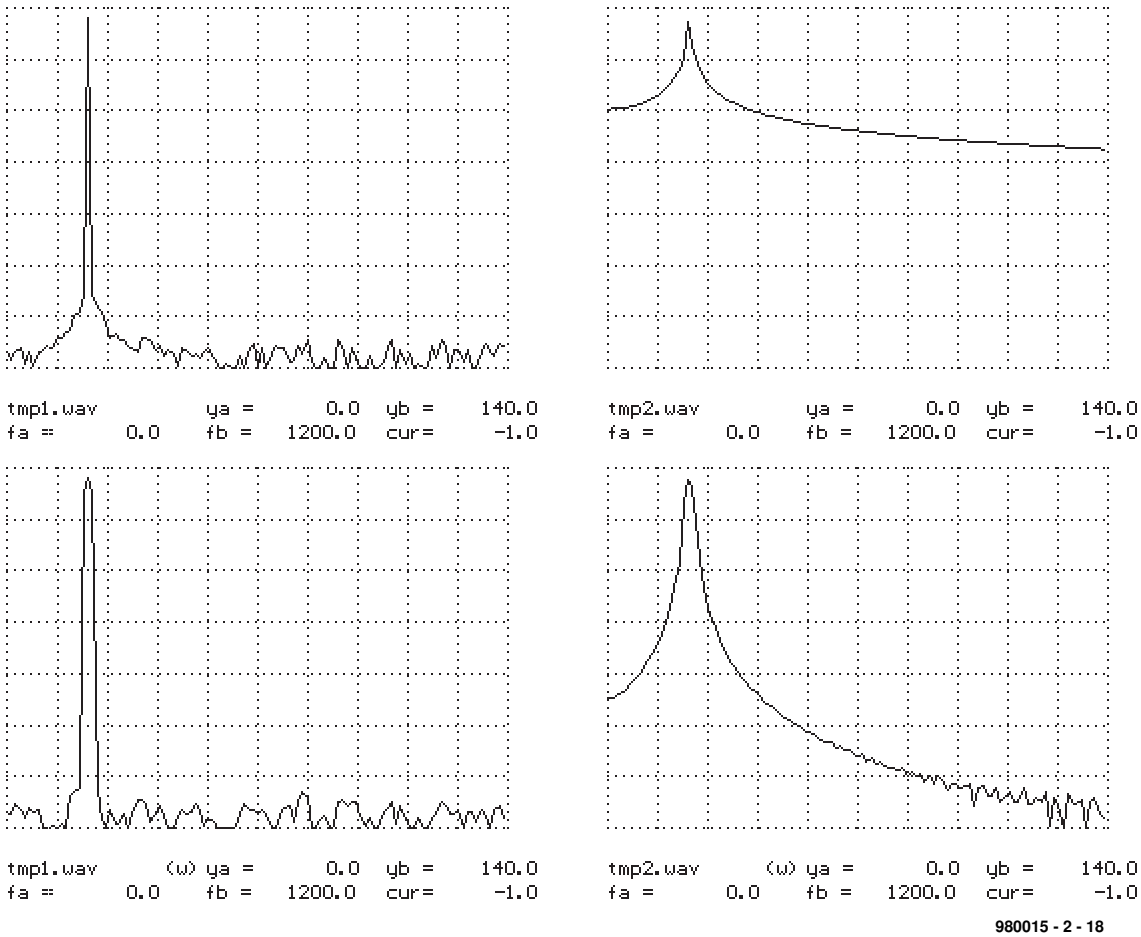


Bild 7. Ein Sweep-Signal wird durch den Tiefpaß geleitet.

Bild 8. Spektren von Sinussignalen, oben ohne und unten mit Fensterung.

aber nicht stark ab.
Wie kommt die DFT zu diesem Ergebnis, daß *alle* Frequenzen im Signal *ein wenig* vorkommen? Bild 9 zeigt, daß beim Signal `tmp1.wav` eine Anzahl vollständiger Perioden genau in den Ausschnitt von 4096 Punkten paßt, wenn es zur DFT herangezogen werden. Auf das Signal `tmp2.wav` trifft das nicht zu. Der Wert des Signals am linken Ende ist Null, am rechten Ende aber nicht, da 18,5 Perioden in den Ausschnitt von 4096 Punkten passen. Aus diesem Grund werden zur Dar-

8



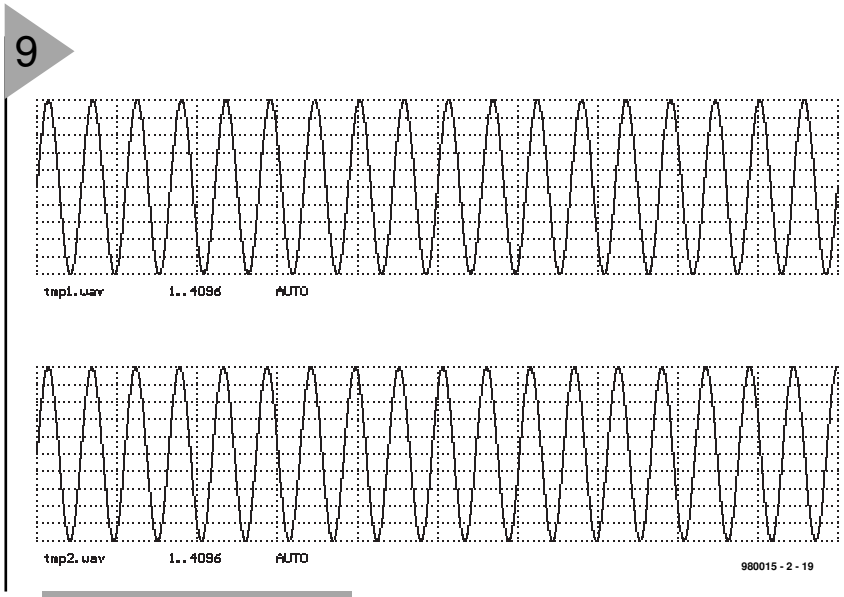


Bild 9. Die Signale passen mehr oder weniger gut in das Fenster des Spektrumanalysers.

stellung des Signals alle möglichen Schwingungen benötigt und entsprechend von der DFT angezeigt.

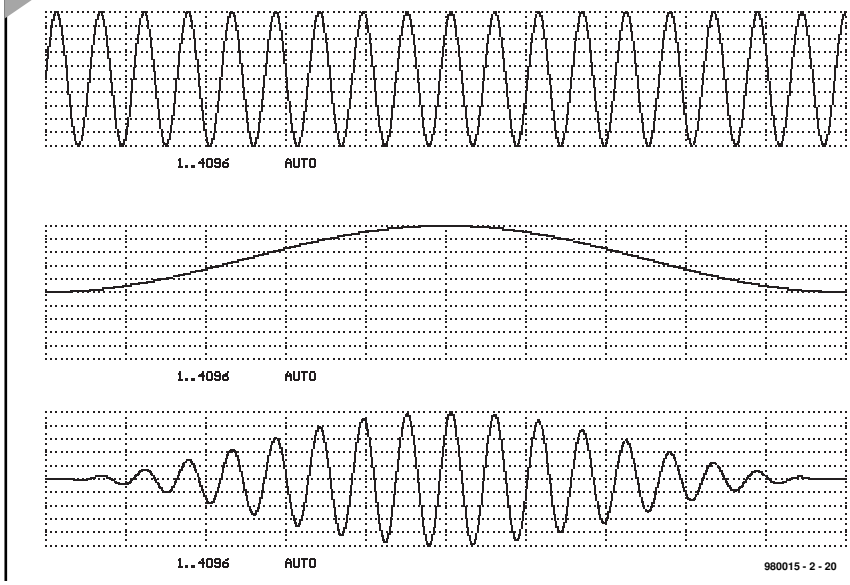
FENSTERLN

Natürlich ist dies kein befriedigender Zustand, denn die DFT soll auch beim Signal `tmp2.wav` ein annähernd linienförmiges Spektrum erzeugen. Das erreicht man durch eine Multiplikation des Eingangssignals (oben in Bild 10) mit einer Fensterfunktion (in der Mitte). Es entsteht das unten abgebildete Signal, das nun einer DFT unterzogen wird. Diese Fensterfunktion erzwingt sozusagen, daß das Signal für die DFT an den Rändern "harmloser", glatter und das resultierende Spektrum linienförmiger wird. In Bild 8 sind unten die Spektren der Signale mit Fensterung wiedergegeben. Wie versprochen, ist das Spektrum von `tmp2.wav` nun deutlich linienförmiger, die scharfe Linie bei `tmp1.wav` allerdings etwas breiter geworden, der Preis, den man bei der Fensterung bezahlen muß. Der Spektrumanalyser auf der CD-ROM läßt die Wahl, mit oder ohne Fensterung zu arbeiten. Das Thema Fensterung wird uns übrigens noch einmal begegnen, wenn es um Filtersynthese geht.

(980015-2)rg

In der nächsten Folge werden wir dann ein paar echte Signale mit dem Spektrumanalyser analysieren und mit dem Thema Filterung fortfahren.

Bild 10. Fensterung heißt Multiplikation des Signals mit einer Fensterfunktion.



Digital Signal Processing

Kursfolge 3: Digitale Bandfilter

In der letzten Folge haben wir uns mit der Berechnung von Spektren beschäftigt. Dieses Grundwissen ist nötig, um mit digitalen Bandfiltern umgehen zu können.

Spektren langer Signale

Bisher wurde die DFT eines Signals fester Länge (in unserem Fall N = 4096) benutzt, um das Spektrum zu berechnen. Wie aber kommt man zu einem Spektrum eines viel längeren Signals ? Dazu gibt es eine Reihe von Verfahren, die in der Literatur [1] beschrieben sind. Unser Analyser macht es sich besonders einfach (Bild 1). Das Eingangssignal wird in Stücke von der Länge N= 4096 unterteilt und diese einzelnen Segmente einzeln mit der Fensterfunktion multipliziert. Dann wird von jedem Segment die DFT berechnet, die Amplitudenwerte (RMS) addiert und nach der Analyse des gesamten Signals das so entstehende Spektrum angezeigt.

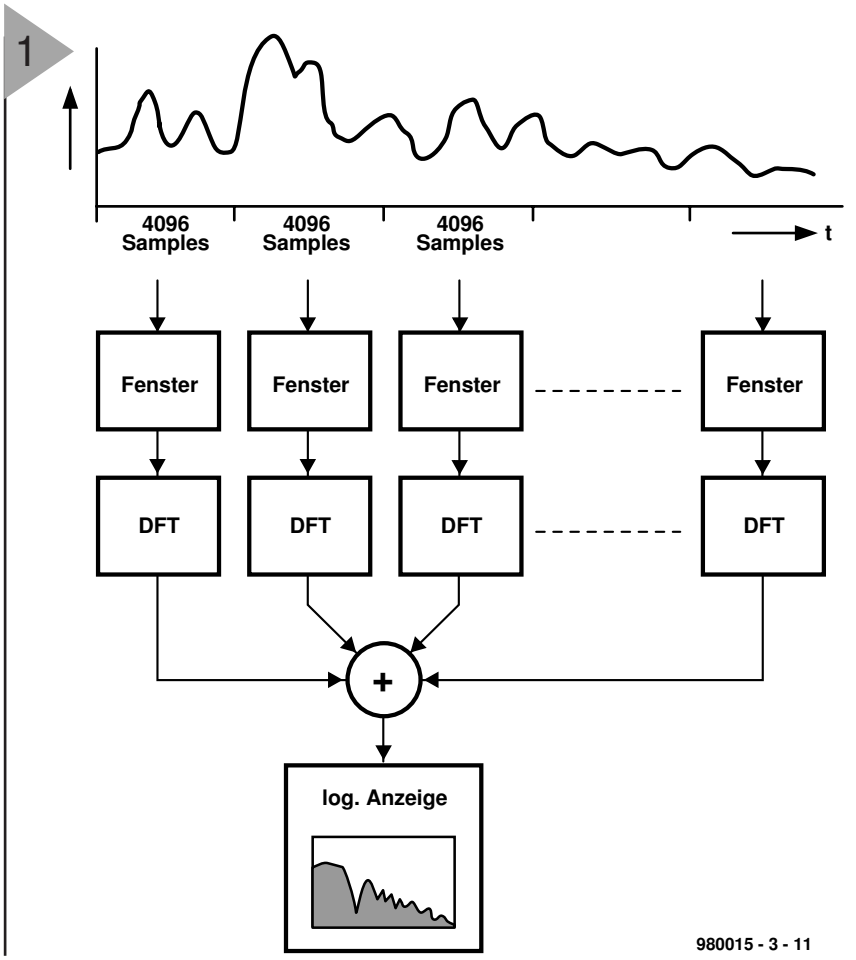


Bild 1. Berechnung des Spektrums langer Signale.

Praktische Anwendung

Zunächst wollen wir aber noch das Spektrum eines echten Signals anschauen, und zwar die Datei `morse2.wav`. Sie enthält zwei Morsesignale mit unterschiedlichen Tonhöhen. Das Experiment `XMORSE1.SPP` berechnet nun das Spektrum (Bild 2). Das Spektrum der Signals weist deutlich zwei Peaks auf, deren Frequenz sich durch Ausmessen bestimmen lassen, auch wenn die Signale sehr schwach sind und von vielen anderen Signalanteilen überlagert sind. Wer es selbst ausprobieren will, analysiere im Ausgangssignal eines FM Empfängers (als Datei auf der CD-ROM) den oft noch vorhandenen Anteil des Stereo-Pilottons auf 19 kHz.

Rauschanalyse des Tiefpassfilters

Nun aber endlich zum Experiment `XLP6.SPP`, das den in Bild 3 skizzierten Aufbau simuliert. Der Signalgenerator `NOISE1.EXE` liefert ein weißes Rauschen, das dem einfachen Tiefpaß zugeführt wird. Wir analysieren das Eingangs- und das Ausgangssignal mit dem Spektrumanalyser `SPEC1`. Das Ergebnis ist in Bild 4 dargestellt.

DFT und FFT

Den Algorithmus, der aus den Abtastwerten das Spektrum berechnet, bezeichnet man als *Diskrete Fourier Transformation (DFT)*. Die schnelle *Fast Fourier Transformation (FFT)* ist ein Verfahren, das die DFT-Berechnungen sehr schnell und effektiv ausführt. Ist `N` die Anzahl der Datenpunkte, so ist der Rechenaufwand für DFT und FFT unten angegeben (Anzahl der Multiplikationen)

Dabei ist `c` der Faktor, um welchen eine FFT-Berechnung schneller ist als

eine normal ausgeführte DFT-Berechnung.

Anzahl der Multiplikationen bei N Datenpunkten für DFT und FFT			
N	DFT	FFT	c
16	256	64	4
128	16 384	896	18
4096	16 777 216	49 152	341

Damit genug zur Berechnung von Spektren, wenden wir uns dem Thema der Filterung zu.

Vom Sinusgenerator zum Bandpass

Den meisten dürften aus der Schule noch die Additionstheoreme der Sinus- und Cosinusfunktion in Erinnerung sein:

$$\cos(a+ß) = \cos a \cdot \cos ß - \sin a \cdot \sin ß$$

$$\sin(a+ß) = \sin a \cdot \cos ß + \cos a \cdot \sin ß$$

Geht man von diesen Theoremen aus, kann man einen digitalen Sinusgenerator realisieren, und anschließend Modifikationen durchführen, um einen digitalen Bandpaß zu erhalten. Wem die Mathematik zu schwer erscheint, der versuche zumindest die Vorgehensweise nachzuvollziehen.
Zuerst setzen wir in den obigen Zeilen

$$P = \cos \varphi, \quad q = \sin \varphi,$$
$$c_k = \cos k\varphi, \quad s_k = \sin k\varphi$$

ein. Ist nun $\varphi = 2\pi f/fs$, so sind c_k und s_k gerade die abgetasteten Werte einer Cosinus- beziehungsweise Sinusschwingung der Frequenz f bei einer Abtatstfrequenz f_s. Um digital eine Cosinus- und Sinusschwingung zu erzeugen, muß man die Werte c_k und s_k, beginnend beispielsweise bei $k = 0$ schnell berechnen. Durch Einsetzen der obigen Abkürzungen in die Additionstheoreme erhält man:

$$c_{k+1} = pc_k - qs_k$$
$$s_{k+1} = qc_k + ps_k$$

Das heißt: Wenn am Anfang die Parameter p und q einmal bestimmt sind, lassen sich aus den Werten c_k und s_k, also den Werten zum Abtastzeitpunkt k, die neuen Werte c_{k+1} und s_{k+1} durch vier Multiplikationen, eine Addition und eine Subtraktion berechnen. Das geht sehr schnell und ist auch auf einem Signalprozessor gut zu programmieren. So ist die Programmierung eines einfachen Oszillators wie im Programm SIN0.PAS (beziehungsweise .EXE) fast ein Kinderspiel. Das Listing in Bild 5 dargestellt.
Das Programm erzeugt ein Sinussignal, wovon man sich durch XSIN1.SPP leicht überzeugen kann. Das Resultat ist in Bild 6 oben dargestellt. Damit der Oszillator startet, müssen die Anfangswerte c_0 und s_0 ent-

2

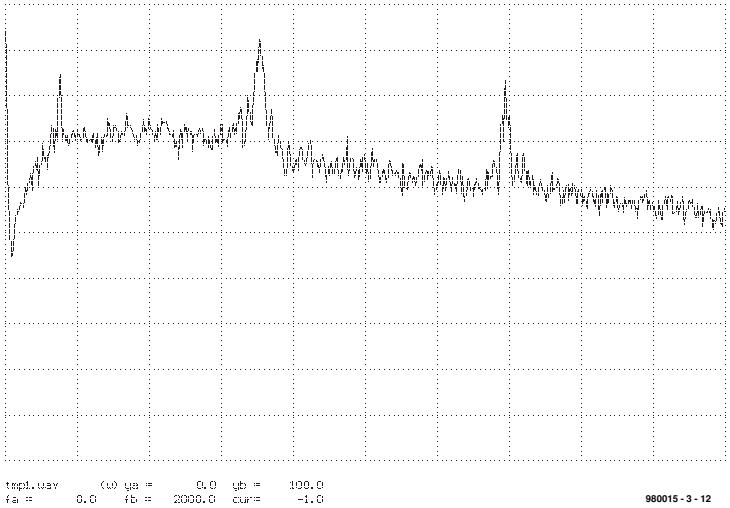


Bild 2. Zwei Morsesignale (700 Hz und 1400 Hz) im Rauschen.

Bild 3. Weißes Rauschen wird tiefpaßgefiltert.

3

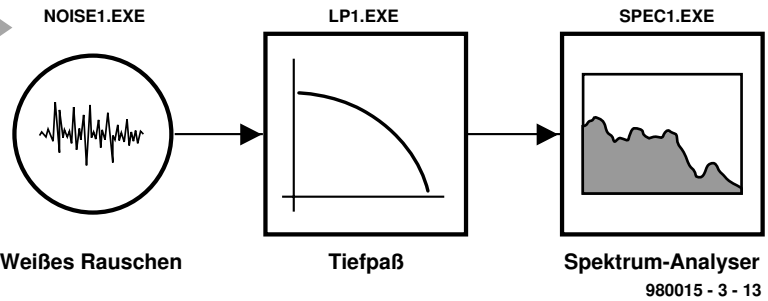
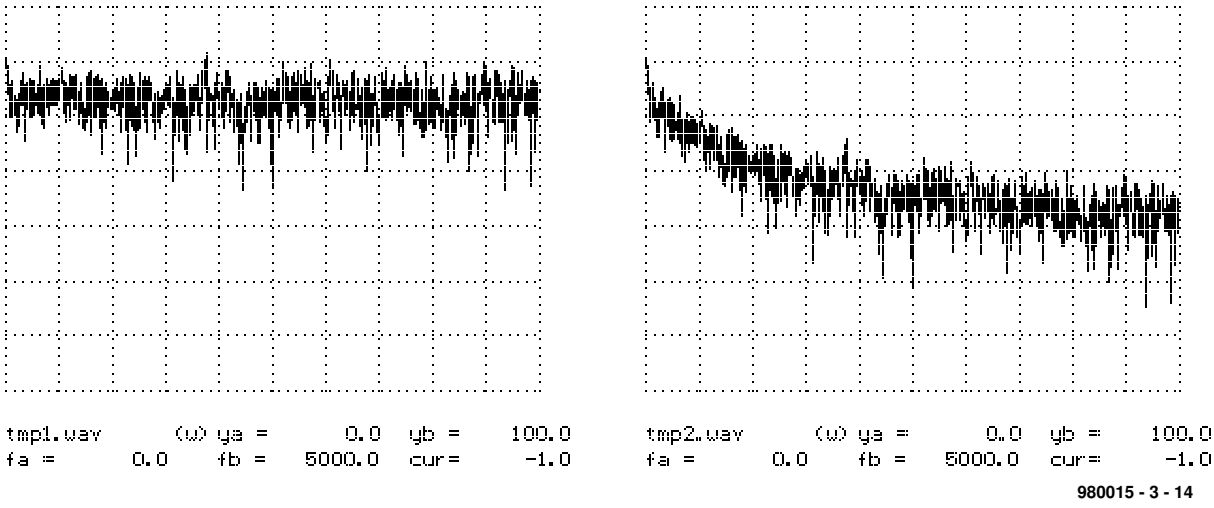


Bild 4. Spektrum von Weißem und gefilterten Rauschen.

4



5

```
program sin0 ;
uses dos,graph,crt ;

{$I SIGLIB.PAS}

var      k:int ;
f0,scale:float ;
p,q,ck,sk,ck_new,sk_new:float ;

begin
start('sin-wave-generator') ;
nsamples:=10000          ; set_par_long('\n=',nsamples) ;
fs:=22050                ; set_par_long('\fs=',fs) ;
f0:=100  ; set_par_real('\f0=',f0) ;
scale:=500               ; set_par_real('\scale=',scale) ;
out_fn:='sin1.wav'       ; set_par_string('\out=',out_fn) ;

open_out(out_fn) ;

p:=cos(2*pi*f0/fs) ;
q:=sin(2*pi*f0/fs) ;
ck:=1 ;
sk:=0 ;

for k:=1 to nsamples do
  begin
  output(scale*ck) ;
  ck_new:=p*ck-q*sk ;
  sk_new:=q*ck+p*sk ;
  ck:=ck_new ;
  sk:=sk_new ;
  end ;
stop ;
end.
```

Bild 5. Programm zur digitalen Sinuserzeugung.

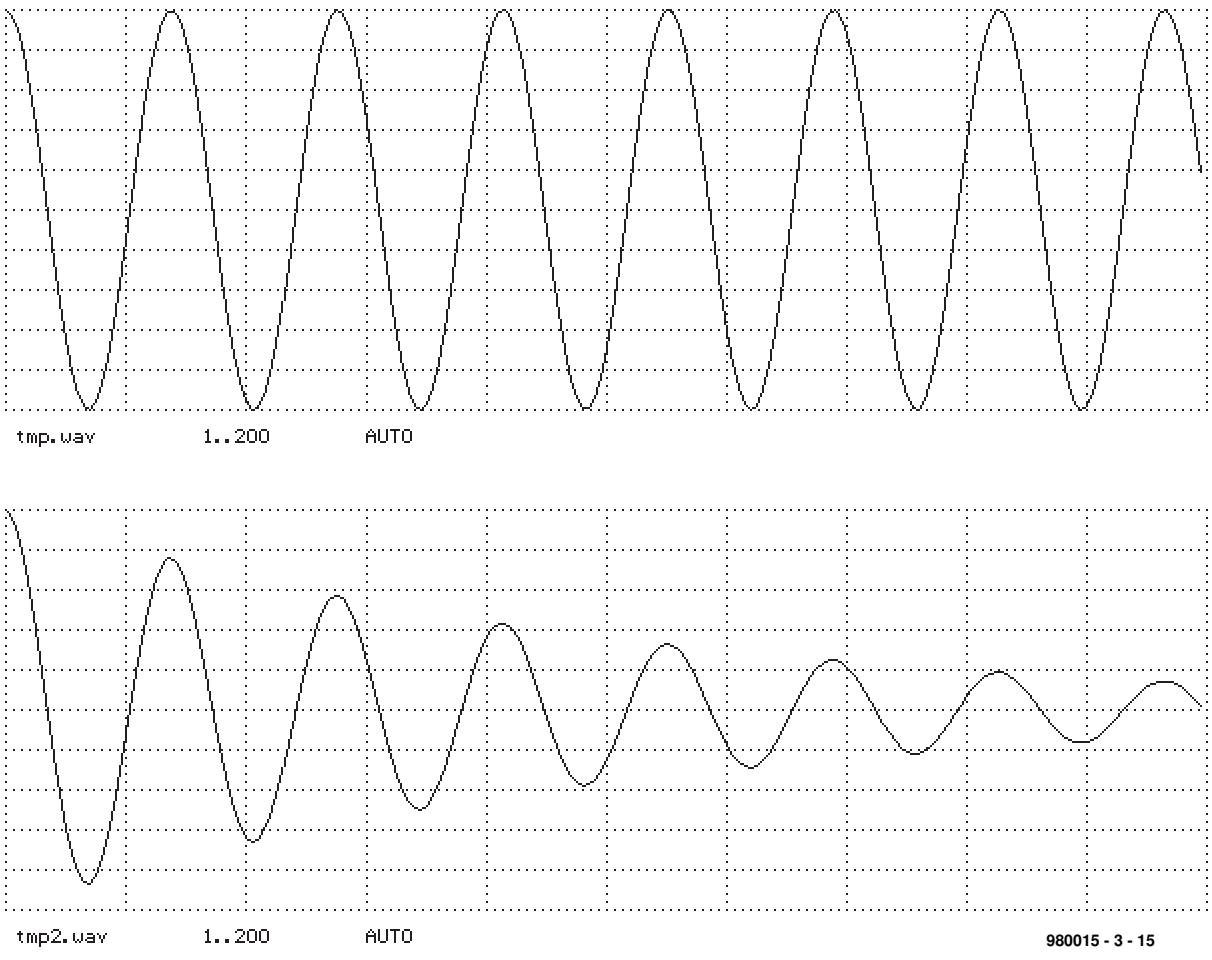
sprechend gesetzt sein, da dies Phase und Amplitude des Oszillators bestimmt.
Bisher war das noch nicht besonders kompliziert. Wenn man es bedenkt, haben wir gerade einen ungedämpften Schwingkreis simuliert. Die Werte c_k sind dann zum Beispiel die Werte des Spulenstromes, die Werte s_k stellen die Kondensatorspannung dar.
Dies ist nicht mehr weit von einem gedämpften Schwingkreis entfernt, wie er in Bild 7 zu sehen ist. Nehmen bei der gedämpften Schwingung die Werte innerhalb jedes Abtastzeitschrittes noch um einen konstanten Faktor r (Dämpfungsfaktor kleiner 1) ab. In die Rechenvorschrift muß also jeweils eine Multiplikation mit $r < 1$ erfolgen. Um den Schwingkreis nun anzuregen, addiert man in der Berechnungsvorschrift noch das Eingangssignal u_k. Damit sieht die Berechnung wie folgt aus:

$$C_{k+1} = r(pc_k - qs_k) + u_k S_{k+1}$$
$$= r(qc_k + ps_k)$$

Das ist bereits ein einfacher Bandpaß,

Bild 6. Oben: Sinussignal, unten: Ausschwingender Schwingkreis.

6



der auch den Kern des zugehörigen Programms SINFIL1.PAS bzw. .EXE bildet. Ein Test beweist, daß es sich wirklich um einen Bandpaß handelt. Als erstes verwenden wir als Eingangssignal den Einheitsimpuls und betrachten die Antwort des Filters. Es sollte eine gedämpfte Schwingung ergeben, und dies macht XSIN1.SPP auch, wie man in Bild 6 unten sieht: Eine schön exponentiell gedämpfte Schwingung, die vom Bandpaß (alias Schwingkreis) erzeugt wird.
Mit dem Experiment XBANDP1.SPP kann man ein Sweep-Signal durch das Filter schicken. Hier kann man wieder mit verschiedenen Werten von f_0 und r spielen, besonders interessant sind die Werte
r = 0.5, r = 0.9, r = 0.99, r = 0.999, r = 0.9999
Etwas seltsam an unserem Filter ist, daß es neben der Cosinus- auch die Sinusschwingung berechnet. Das läßt sich aber umgehen. Man erhält dann ein Filter mit der Berechnungsvorschrift:

$$c_{k+2} = b_1 c_{k+1} + b_2 c_k + u_k$$

und mit den sogenannten Filterkoeffizienten

$$b_1 = -r^2 \quad \text{sowie}$$
$$b_2 = 2 \cdot r \cdot \cos(2\pi f_0 / f_s)$$

Dieses Filter ist im Programm BANDP1.PAS (bzw. .EXE) realisiert.

REKURSIVE ODER IIR-FILTER

Vergleicht man die jetzt abgeleiteten Filter mit dem ersten Tiefpaß, stellt man eine Gemeinsamkeit fest. Beide Filter berechnen den Ausgangswert, indem sie auf vorherliegende Ausgangswerte zurückgreifen. Solche Filter nennt man *rekursive Filter*. Ihre Impulsantwort ist im Normalfall unendlich lang, klingt aber immer mehr ab. Daher hat sich auch der Begriff *Infinite-Impulse-Response Filter (IIR Filter)*. Der erste Tiefpaß griff nur auf einen vorhergehenden Wert zurück, stellt also ein IIR-Filter erster Ordnung dar. Beim Bandpaß, der auf zwei Werte zurückgreift, handelt es sich um ein Filter zweiter Ordnung. Es gibt auch noch Filter höherer Ordnung, die aber meist durch Kaskadierung mehrerer Filtersektionen zweiter Ordnung realisiert werden. Dem Entwurf solcher Filter widmen sich ganze Bücher, ein nicht ganz einfaches Thema [1, 2, 3].

BANDPASS-ANWENDUNGEN

Mit einem solchen Bandpaß kann man natürlich etwas unternehmen, zum Beispiel, bestimmte Frequenzen selektieren. Die Datei MORSE2.WAV enthält das Signal eines Amateurfunkempfängers, der zwei in der Frequenz nah beieinander liegende Morsesignale empfängt. Mit dem Bandpaß kann man eins dieser Signale hervorheben und deutlicher machen, eventuell ist auch eine automatisch Detektion möglich. Dazu eignet sich das Experiment XMORSE3.SPP, das das Signal MORSE3.WAV durch zwei Bandfilter mit den Mittenfrequenzen 700 Hz und 1400 Hz schickt. Das Ergebnis in Bild 8 zeigt klar erkennbar die beiden voneinander getrennten Morsesignale.
Probieren Sie die beteiligten Dateien unbedingt aus und experimentieren ein wenig mit den Filterparametern. Dann merkt man leicht die Wirkungsweise eines einfachen Bandpasses. Schicken Sie auch einmal ein Musiksignal (MUS1.WAV) durch ein Bandfilter und hören Sie sich das Resultat an.

(980015-III)

7

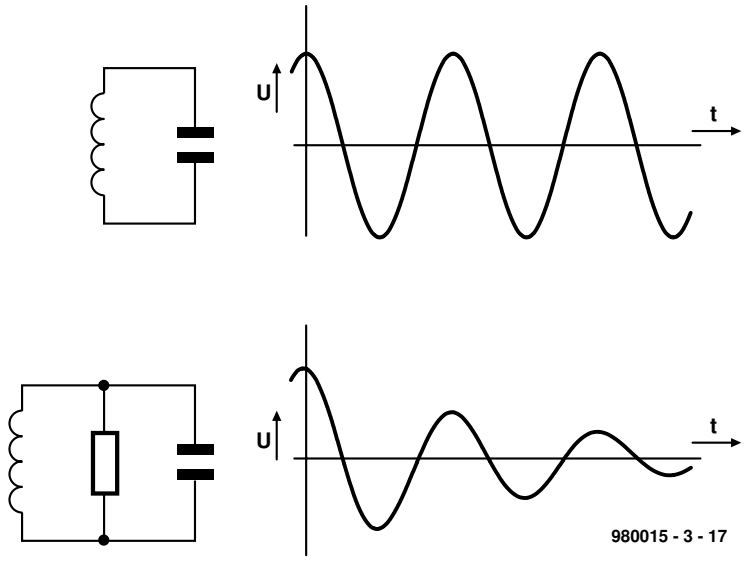


Bild 7. Ungedämpfter und gedämpfter Schwingkreis.

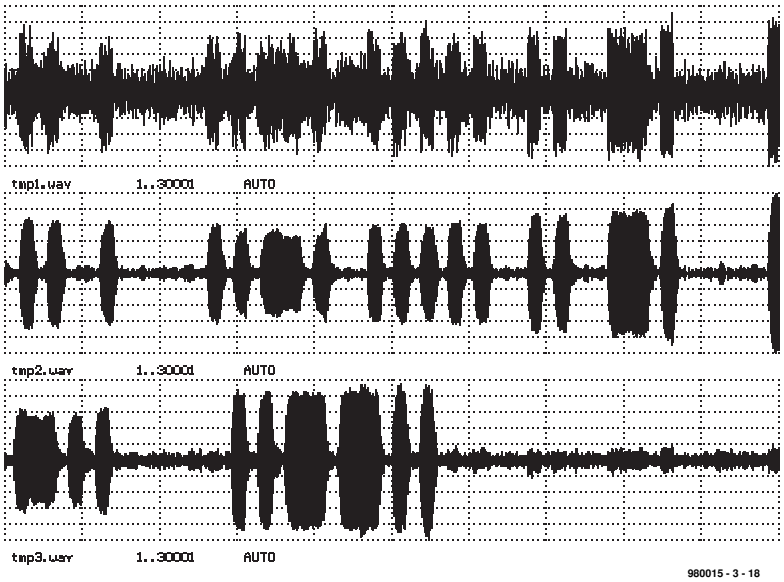
In der nächsten Folge werden wir einen Echogenerator kennenlernen und so modifizieren, daß er zur digitalen Filterung taugt.

Literatur

[1] Oppenheim, Schafer: Digital Signal Processing, Prentice-Hall ,1975
[2] Lücker,R., Grundlagen digitaler Filter, Springer 1980
[3] Digital Processing of Signals, Rader C.M. Gold,B.,McGraw-Hill,New York

Bild 8. Das Morsesignal vor (oben) und nach Filterung (700Hz und 1400Hz). Erkennbar sind die zwei getrennten Signale.

8



Digital Signal Processing

Kurs Teil 4: Vom Echo zum FIR-Filter

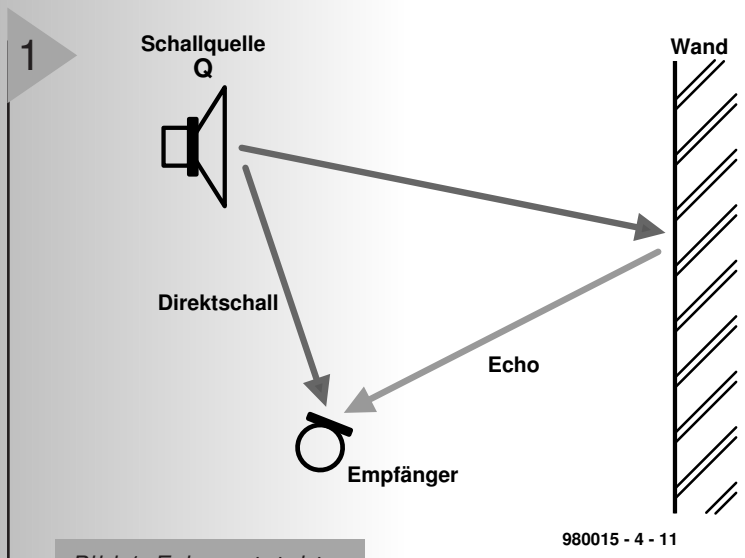


Bild 1. Echo entsteht durch Reflektion = Dämpfung und Verzögerung.

Eine wichtige Aufgabe DSPs im Audiobereich ist die Erzeugung von Nachhall und Echo. Zuerst wollen wir uns deshalb überlegen, wie man auf einfache Weise Echo erzeugen kann und anschließend die dabei gewonnenen Tricks verallgemeinern. Auf diese Weise gelangen wir zu einer ganzen Klasse von digitalen Filtern.

Echo

Echo entsteht, wenn man nicht nur den Direktschall einer Schallquelle Q wahrnimmt, sondern gleichzeitig auch das von einer schallreflektierenden Wand reflektierte Signal. Das reflektierte Signal entspricht (idealisiert) bis auf Dämpfung und Zeitverzögerung dem Originalsignal s_k. Ist die Verzögerung des Signals `M` Abtastschritte groß, so empfängt man das Signal y_k:

$$y_k = s_k + a \cdot s_{k-M}$$

s_k entspricht dem Direktschall, s_{k-M} ist das verzögerte Signal und `a` die Dämpfungskonstante. Wie führt man nun so etwas als Programm aus? Wie man sieht, muß man, um zum Zeitpunkt `k` das Ausgangssignal y_k zu bestimmen, das Eingangssignal kennen, das `M` Abtastschritte zurückliegt. Dazu braucht man einen Zwischenspeicher, der (mindestens) `M` Werte fassen kann. Zweckmäßigerweise organisiert man diesen als Ringpuffer, auf den man mit einem Zeiger zugreift, der sich bei jedem Abtastschritt um eine Position fortbewegt. Das ganze erinnert sehr an die alten Echogeneratoren, die ein umlaufendes Magnetband zur Speicherung benutzten, mit dem einzigen Unterschied, daß sich in der Ringpuffertechnik sozusagen die Schreib/Leseköpfe bewegen, nicht aber die Daten wie beim Magnetband.
Ein Programm (`ECHO1.PAS`) für M = 8192 Abtastschritte ist in Bild 2 zu sehen.
Die Verzögerungszeit hängt dabei von der Abtastrate ab. Wir benutzen in diesem Beispiel Daten mit 22050 Samples/s und erhalten 8192/22050 = 0,37 s. Zuerst sei ein Ohr auf die Originaldatei `WD1R.WAV` geworfen, anschließend versehen wir sie mit dem Aufruf

```
ECHO1 \inp=wd1r.wav out=tmp.wav
```

und statten sie mit einem Echo aus, das sehr schön in der Datei `TMP.WAV` zu hören ist.

Mehrere Echos

Nun muß man sich nicht mit einem Echo zufriedengeben. Um zwei oder mehrere Echos zu realisieren, reicht eine Addition des direkten mit zwei reflektierten Signalen, die jeweils unterschiedliche Verzögerungen `L` und `M` und Dämpfungen `a` und `b` aufweisen:

$$y_k = s_k + a \cdot s_{k-L} + b \cdot s_{k-M}$$

Für noch mehr Echos setzt man entsprechend mehr Summanden ein. Auch für diese Aufgabe gibt es ein Programm auf der CD-ROM mit dem Namen `FIRFIL1`. Die tiefere Bedeutung dieses Namens lernen wir später kennen. Es ist in der Lage, ganz allgemeine Echos berechnen. Um das Gesamtecho zu spezifizieren, muß man dabei in einer Datei die Zahl der Echos sowie ihre jeweilige Zeitverzögerung und Intensität angeben.
Mit der Datei `XECHO2.SPP`

```
\\file 'exec.bat'
firfil1 \inp= speech1.wav \out= tmp.wav
    \filter= fir1.fir
\\eof

\\file 'fir1.fir'a simple test filter50      0.5
 3000  0.4
 6000  0.3
12000  0.2
16000  0.1
\\eof

\\end
```

erhält man fünf Echos mit den Verzögerungszeiten von 0, 3000, 6000, 12000 und 16000 Samples und den Stärken 0,5, 0,4, 0,3 und so weiter. Die maximale Verzögerungszeit beträgt 16383

Samples, die Maximalzahl der Echos 1000. Wir probieren nun dieses Echo mit XECHO2.SPP aus. In der Datei TMP.WAV sind schon ganz schöne Echos zu hören. Wenn man genügend Echos addiert (besonders frühe), kann man auch einen Nachhall simulieren.

FIR-FILTER

Das Programm FIRFIL1 kann viel mehr als Echos zu erzeugen. Zuerst ein Experiment: Wir erzeugen ein Sweep-Signal und schicken es durch das Filter, das in XECHO3.SPP definiert ist. Es weist bereits 256 Echos mit ziemlich seltsamen Werten auf. Das Ergebnis ist auf dem Scope zu sehen (Bild 3). Es handelt sich eindeutig um einen Bandpaß, das Sweep-Signal wird nur in einem bestimmten Bereich durchgelassen.

Nun kann man durch Probieren bestimmt nicht so einfach die Stärke von 256 Echos so ermitteln, daß sich eine Bandpaßfunktion ergibt. Dies erfordert eine konstruktive Vorgehensweise, wie sie gleich vorgestellt wird. Vorher jedoch wollen wir verstehen, wie die Filter funktionieren.

IMPULSANTWORT VON FIR-FILTERN

Ein FIR-Filter, das Echos mit der Zeitverzögerung 0 und der Stärke a_0, mit der Zeitverzögerung 1 und Stärke a_1 und so weiter erzeugt, wird durch die Rechenvorschrift

$y_k = a_0 \cdot x_k + a_1 \cdot x_{k-1} + a_2 \cdot x_{k-2} \ldots$

beschrieben. Dabei sind x_k die Eingangs- und y_k die Ausgangswerte.
Wir wollen nun überlegen, welche Impulsantwort dieses Filter besitzt., oder, mathematisch ausgedrückt, wie die Werte y_k aussehen, wenn nur $x_0 = 1$ ist, alle anderen x_j-Werte jedoch Null. Zunächst ist

$y_0 = a_0 \cdot x_0 + a_1 \cdot x_{-1} + a_2 \cdot x_{-2} \ldots$

zu berechnen. Nur der erste Term der Summe bleibt übrig, also $y_0 = a_0$.
Für $k = 1$ ergibt sich

$y_1 = a_0 \cdot x_1 + a_1 \cdot x_0 + a_2 \cdot x_{-1} \ldots$

Diesmal bleibt der zweite Term übrig ($y_1 = a_1$), und so geht das immer weiter, daß man für alle $k \geq 0$ behaupten kann: $y_k = a_k$. Als Impulsantwort erhalten wir also gerade die Filterkoeffizienten.

Schickt man also einen Impuls durch ein solches Filter und sieht sich das Resultat an, sieht man die Filterkoeffizienten. Dafür ist der zweite Teil von XECHO3.SPP verantwortlich. Die Impulsantwort des obigen Bandpasses ist in Bild 4 dargestellt. Die Impulsantwort sieht aus wie ein ein- und ausschwingender Schwingkreis. So etwas war zu erwarten, da auch "echte" Bandpässe sich so ähnlich verhalten wie ein paar gekoppelte Schwingkreise. Bevor wir daran gehen können, ein solches Filter zu konstruieren, muß Zeit für einen kleinen Exkurs sein.

2

```
program echo1 ;
uses dos,crt,graph ;
{I SIGLIB.PAS }

var k:int;
    x,y:float ;
    buffer:array[0..8191] of float ;
    pointer:int ;

begin
start('simple echo') ;
inp_fn:='tmp1.wav'   ; set_par_string('\inp=',inp_fn) ;
out_fn:='tmp.wav'    ; set_par_string('\out=',out_fn) ;

open_inp(inp_fn) ;
open_out(out_fn) ;

pointer:=0 ;
for k:=0 to 8191 do buffer[k]:=0 ;

for k:=1 to nsamples do
  begin
  x:=input ;
  y:=buffer[pointer] ;
  buffer[pointer]:=x ;
  pointer:=pointer+1 ;
  if pointer>=8192 then pointer:=0 ;
  output(x+0.5*y) ;
  end ;

stop ;
end.
```

Bild 2. Programm zur Echoerzeugung

3

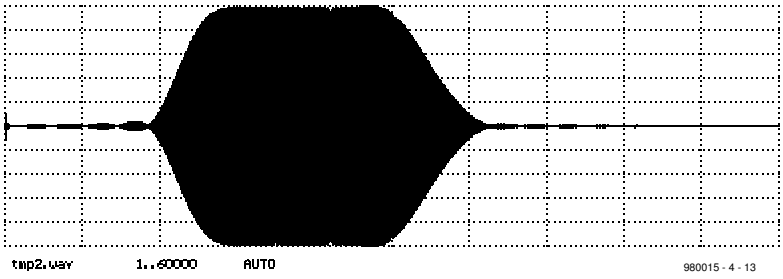


Bild 3. Sweep nach Durchlaufen durch ein Echo-Filter zeigt eine Bandpaß-Charakteristik.

4

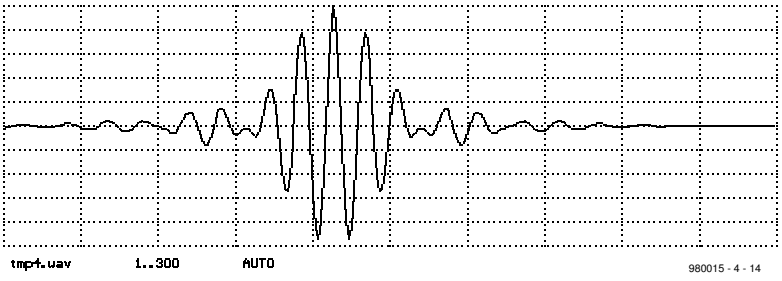


Bild 4. Impulsantwort des Bandpasses

SIGNALE MIT ALLEN FREQUENZEN

Schon im letzten Kursteil wurde ein Rauschsignal verwendet, um den Frequenzgang eines Filters zu bestimmen. Dabei macht man sich die Eigenschaft zu nutze, daß weißes Rauschen alle Frequenzen gleichmäßig enthält. Nach einem Filterdurchlauf (Bild 5 oben) sind die einzelnen Frequenzen gerade noch so stark im Ausgangssignal enthalten, wie das Filter diese Frequenz

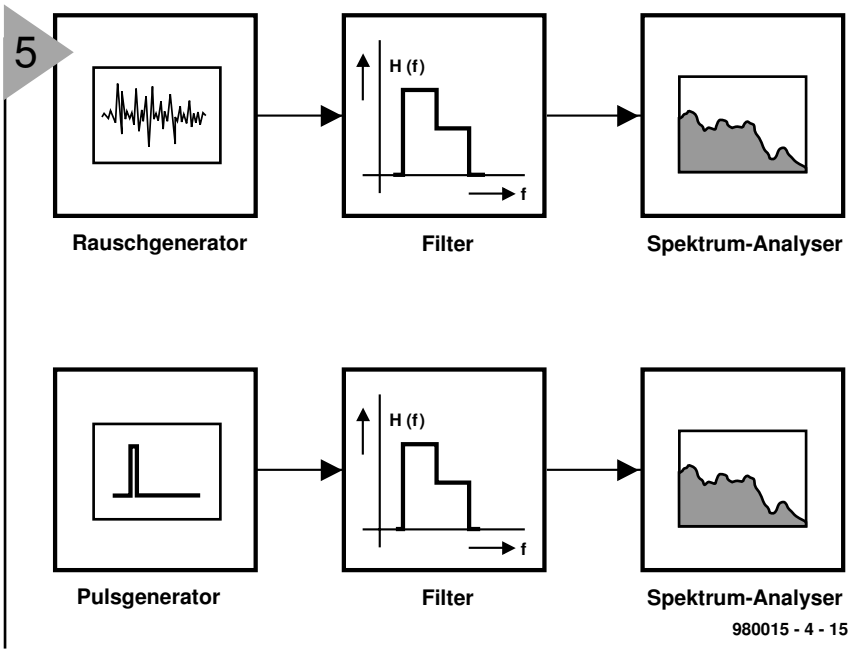


Bild 5. Rauschen und Impulse enthalten alle Frequenzen.

passieren läßt. Dies ist links in Bild 6 deutlich zu erkennen (Experiment `XPULSE1.SPP`).
Nun muß man mit Rauschen Messungen relativ lange ausdehnen, um mittels Mittelwertbildung genaue Resultate zu erhalten. Es stellt sich daher die Frage, ob es nicht Signale gibt, die kürzer sind und dennoch alle Frequenzen enthalten. Das einfachste vorstellbare Signal ist der Impuls, wie ihn der Signalgenerator `PULSE1.EXE` erzeugt. Das Spektrum dieses Impulses ist im idealen Sinne konstant über den Frequenzbereich (Bild 6 rechts oben). Damit kann das Rauschen aus der Frequenzgangmessung durch einen Impuls ersetzt werden. Ein Filter reagiert auf diesen Puls mit der sogenannten Impulsantwort. Diese enthält alle Frequenzen so stark, wie sie vom Filter durchgelassen werden, oder, anders ausgedrückt: *Das Spektrum der Impulsantwort ist genau der Frequenzgang des Filters*, wie der Vergleich von `tmp3.wav` und `tmp4.wav` in Bild 6

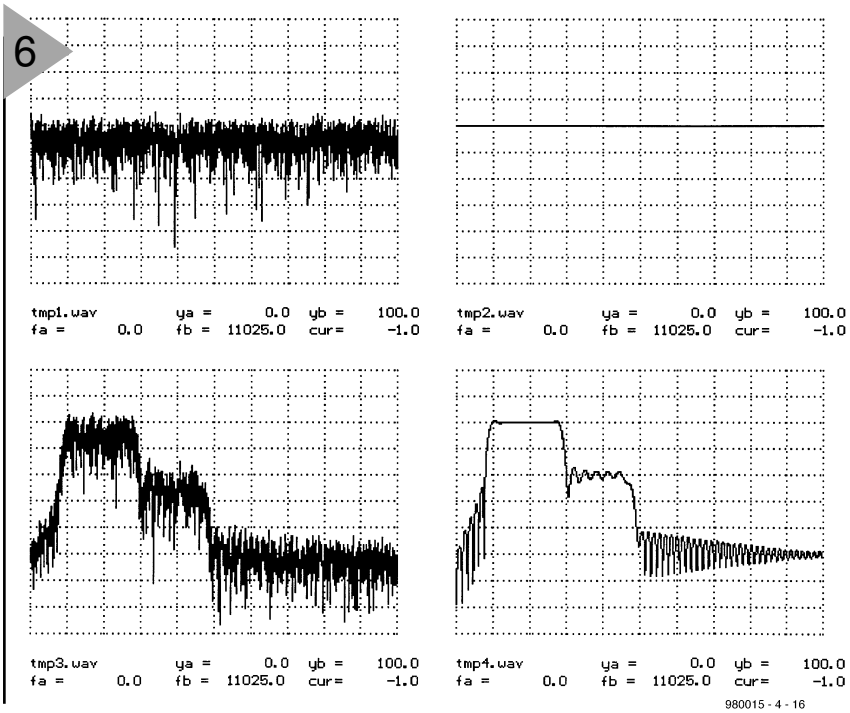


Bild 6. Frequenzgangbestimmung mit Rauschen (links) oder Impuls (rechts), oben jeweils das Spektrum des Eingangssignals, unten das des Ausgangssignals.

deutlich beweist. Das ist eine wichtige Grundlage der digitalen Signalverarbeitung, die man auch mathematisch auf solide Füße stellen kann. Damit man mit dem Spektrumanalyser die Spektren von Impuls und Impulsantwort korrekt auswerten kann, muß übrigens die Fensterung abgeschaltet werden, ansonsten erhält man falsche Resultate. Wichtig ist auch, daß die Impulsantwort kurz genug sein (wie hier 4096 Samples), damit sie ganz in den DFT-Bereich des Spektrumanalysers paßt.
Die Frequenzgangbestimmung mittels DFT der Impulsantwort, wie sie Bild 7 oben beschreibt, ist eine einfache, schnelle und sehr genaue Methode, um Frequenzgänge von Filtern zu

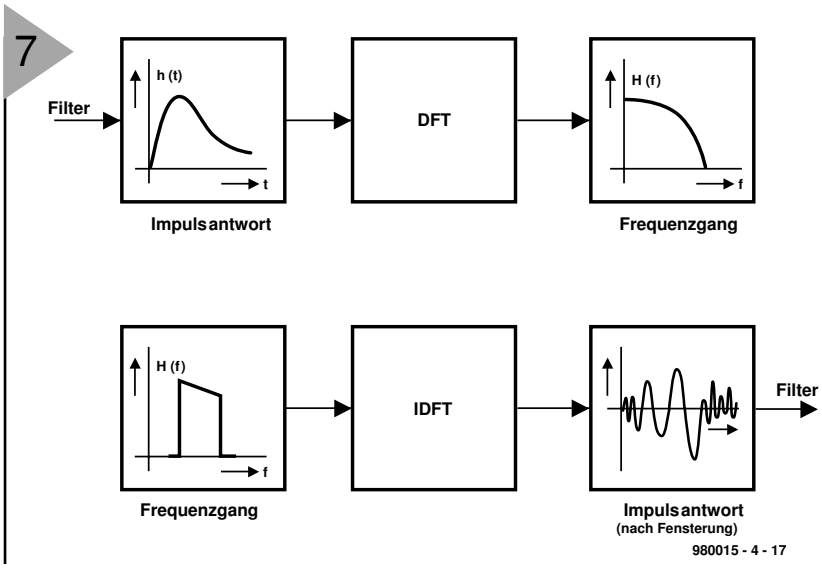


Bild 7. Mit der DFT kann man aus der Impulsantwort den Frequenzgang bestimmen. Mit der Inversen DFT bestimmt man aus dem Frequenzgang die Impulsantwort.

bestimmen. Ist man dagegen Frequenzgängen von Verstärkern auf der Spur, kann es passieren, daß die kurzen und starken Impulse den Verstärker übersteuern beziehungsweise man keine starken Impulse als Eingangssignal verwenden kann. Die noch erlaubten Impulse enthalten alle Frequenzen nur noch relativ schwach, so daß die Meßergebnisse an einem sehr schlechten Signal/Rauschverhältnis leiden. Man weicht deshalb oft auf sogenannte Pseudo-Rauschsignale aus, die eine begrenzte Amplitude und eine gleichmäßige Stärke aller Frequenzen optimal verbinden.

FILTERDESIGN IM FREQUENZBEREICH

Wie wir gesehen haben, kann man mit der *Diskreten Fourier Transformation (DFT)* Spektren von Signalen, insbesondere mit der Impulsantwort eines Filters seinen Frequenzgang bestimmen. Da erhebt sich natürlich auch die Frage, ob man mit Hilfe der *Inversen Diskreten Fourier Transformation (IDFT)* aus einem vorgegebenen Frequenzgang auch die Impulsantwort eines Filters herleiten kann. Dies ist tatsächlich möglich: Ist die Impulsantwort bekannt, läßt sich ja schließlich das Filter mit dem FIR-Filterprogramm `FIRFIL1.EXE` simulieren. Auch in Digitalen Signalprozessoren werden Filter oft so realisiert.

Mit Hilfe des Filterdesign-Programms `SPECFIL1.EXE` ist auf relativ einfache Weise der Entwurf von FIR-Filtern möglich. Es arbeitet wie in Bild 7 unten dargestellt. Aus dem vorgegebenen Verlauf der Filterfunktion im Frequenzbereich wird durch inverse DFT eine Impulsantwort berechnet, die anschließend mit einem *Kaiser-Fenster* modifiziert wird. Dadurch kann man die Eigenschaften des Filters noch tunen. Dann wird die Impulsantwort des Filters in eine Datei ausgegeben, die das FIR-Programm direkt als Eingabe weiterverarbeitet. Bild 8 zeigt eine SPP-Datei (`XFILDES1.SPP`), die ein einfaches Filter berechnet.

Die Filterspezifikation, neben der Sample-rate und der Anzahl der Samples der Impulsantwort die Parameter des Kaiser-Fensters (a) und die Werte der Filterverstärkung an gewünschten Stellen, liegt in der Datei `TMP.FIL`. Zwischen den angegebenen Stellen wird der Verstärkungsverlauf linear interpoliert (Bild 9).

Die Ausgabe des Programs ist in Bild 10 dargestellt. Oben links ist der vorgegebene und der realisierte Frequenzverlauf zu sehen (logarithmische Skalierung), unten links ist die Impulsantwort des Filters abgebildet, daneben der Frequenzgang mit linearer Skalierung.

8

```
\\file 'exec.bat'
\\ generate filter
specfil1 \filter=tmp.fil \fir=tmpa.fir \bitmap=pic.ps
\\eof

\\ filter specification

\\file 'tmp.fil'
a simple filter as example
normal
44100    ; sample frequency
128      ; number of taps
7        ; alpha for window
7        ; number of data points in frequency domain
0      0
1990   0
2000  100
5000  100
5200  10
8000  10
8200  0
\\eof
```

9

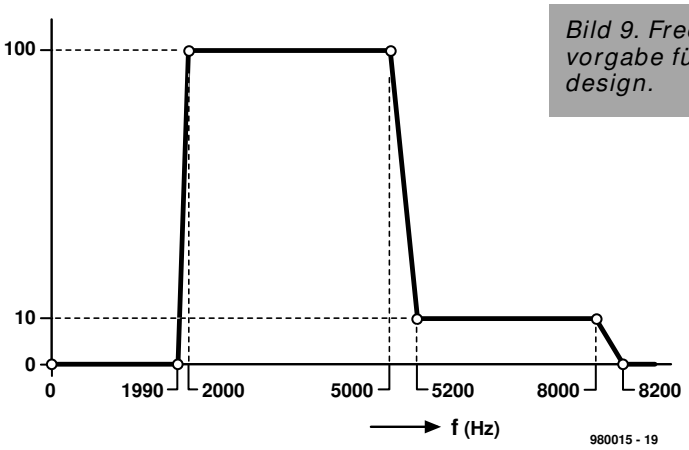


Bild 9. Frequenzgangvorgabe für das Filterdesign.

10

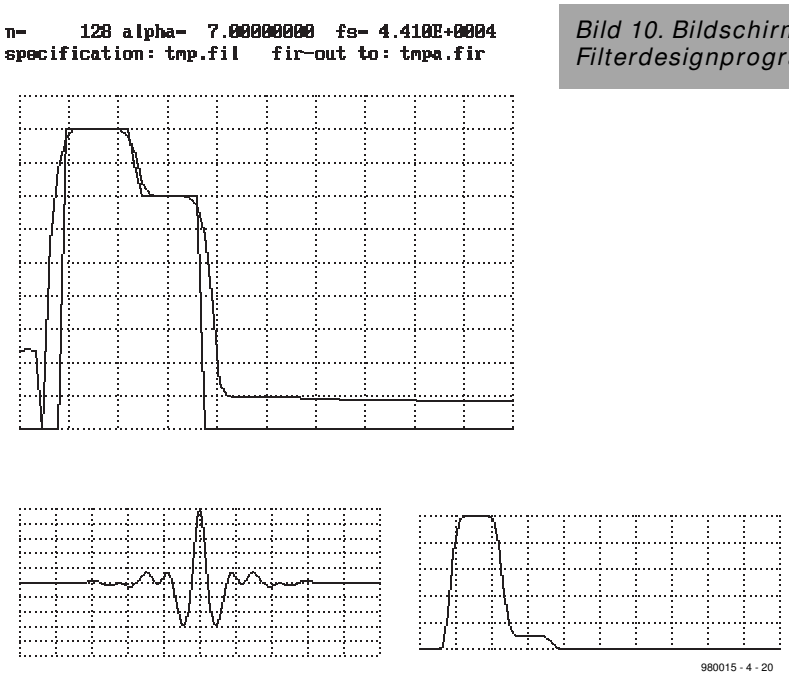


Bild 10. Bildschirm des Filterdesignprogramms.

In der nächsten Folge schließen wir das Kapitel Filterung ab und wenden uns Modulations- und Demodulationstechniken zu.

(980015-4)

Digital Signal Processing

Kurs Folge 5: Noch mehr Filter, De- und Modulation

Zur Synthese von Filtern muß noch etwas mehr gesagt werden, bevor wir uns dem neuen Thema Modulations- und Demodulationstechniken zuwenden.

1

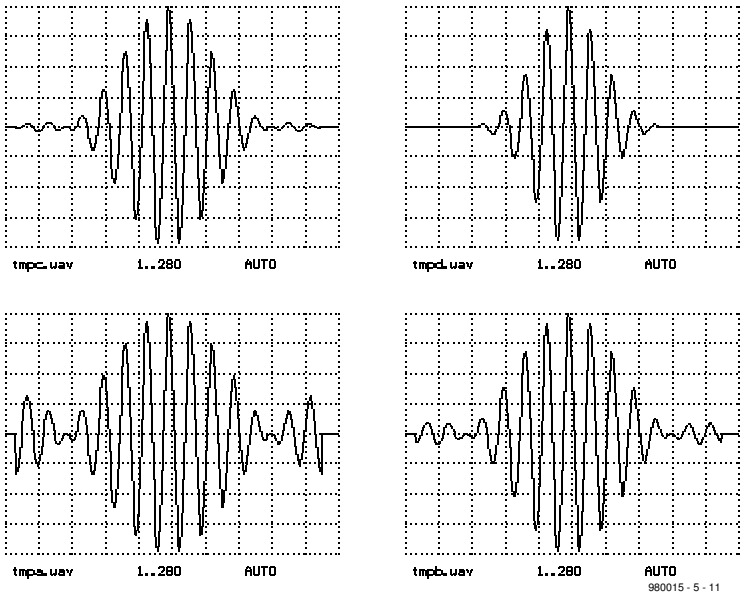


Bild 1. Impulsantwort bei verschiedenen Fenstern.

Schon wieder: Fensterung

Um bei vorgegebenem Frequenzgang ein Filter zu entwerfen, hat man viele Freiheitsgrade. Zuerst wäre die Anzahl der Samples der Sprungantwort zu nennen, also die Ordnung des Filters oder Zahl der Taps. Zu hohe Ordnungen führen zu immensen Rechenzeiten. Im allgemeinen versucht man daher, die Ordnung `N` so klein wie möglich zu halten. Dann kommt bei unserem Programm noch die Wahl des Fenster-Parameters dazu. Sein Einfluß ist in **Bild 1** und **Bild 2** dargestellt. Die Oszillogramme wurden mit `XFILDES2.SPP` erzeugt, die Werte von α sind 0.1, 3, 5 und 14. Eine Wahl eines kleinen Wertes von α führt dazu, daß praktisch keine Fensterung vorgenommen wird. Die Sprungantwort weist am linken und rechten Ende "Zacken" auf, die zu Nebenzipfeln im Frequenzgang führen. Ein steigender Wert von α geht mit einer Reduzierung der Nebenzipfel einher, aber der Frequenzgang folgt den Vorgaben nicht mehr so scharf, sondern verrundet mehr und mehr. Zwischen diesen beiden Übeln kann man wählen, will man nicht die Ordnung des Filters erhöhen.

Phasengang

Das Filterdesignprogramm erzeugt Filter mit sogenanntem linearen Phasengang. Alle Frequenzkomponenten werden quasi nur um eine konstante Zeit durch das Filter verzögert. Diese Zeit ist genau die Hälfte der Filterordnung. In der zweiten Zeile der Filterspezifikation kann man anstatt "normal" auch noch "hilbert" eintragen, dann wird ein Filter erzeugt, daß den gleichen Frequenzgang hat, aber alle Signalanteile um 90 Grad dreht. Ein solches Filter nutzt man zur Erzeugung bestimmter Signale, wie noch gezeigt wird.

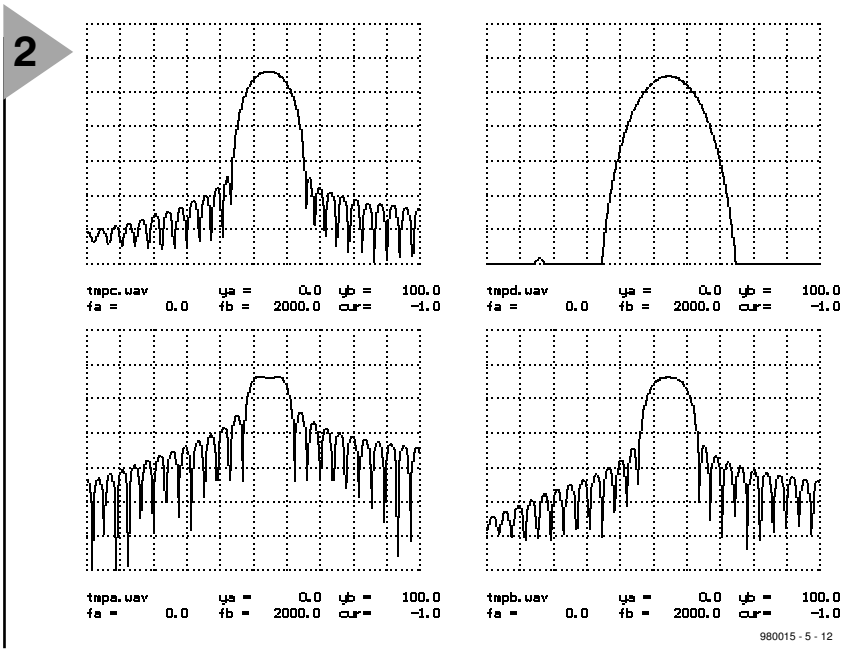


Bild 2. Frequenzgang bei verschiedenen Fenstern.

BESSERE FIR-FILTER

Das einfache Design-Programm erzeugt normalerweise nicht die optimalen Filter, so daß man häufig mit geringerer Filterordnung den vorgegebenen Frequenzgang noch besser treffen könnte. Das Design solcher Filter ist aber nicht ganz einfach. Es sei dazu auf die Literatur [Lit. 1,2] verwiesen. Es gibt dazu auch eine Reihen von kommerziellen Programmen, die der professionelle Programmierer von DSP-Systemen anwenden kann, wenn er sehr gute Filter benötigt. Einen Programmausschnitt für ein FIR-Filter auf einem DSP, dem ADSP2181 von Analog Devices zeigt **Bild 3**. Bei dem üblicherweise verwendeten 16,66-MHz-Quarz wird ein Befehl in 30 ns ausgeführt.

Die Innere Schleife (Zeile 3/4) benötigt pro Durchlauf 30 ns. Bei einer Samplerate von 48.000 Samples pro Sekunde bleiben pro Sample nur 20 μs Zeit, bei Stereosignalen nur 10 μs. Bei Stereosignalen kann ein FIR-Filter also eine Länge von **N = 10 μs / 30 ns = 333** haben. Man sieht, daß selbst moderne DSPs nicht in der Lage sind, viele lange Filter gleichzeitig ablaufen zu lassen. Daher ist man bei DSP-Entwicklungen immer bestrebt,

3

```
1  cntr=taps-1;
2  mr=0,mx0=dm(i2,m2),
   my0=pm(i7,m7);
3  do fir1 until ce;
4  fir1:
   mr=mr+mx0*my0(ss),
   mx0=dm(i2,m2),
   my0=pm(i7,m7);
5  mr=mr+mx0*my0(rnd);
6  if mv sat mr;
```

Bild 3. Listing eines DSP-FIR-Filterprogramms.

mit niedrigen Abtastraten und einfachen Filtern zu arbeiten. Oder man verwendet keine FIR-Filter, sondern die früher erwähnten IIR-Filter, die allerdings oft keine besonderen Anforderungen an den Phasengang erfüllen.

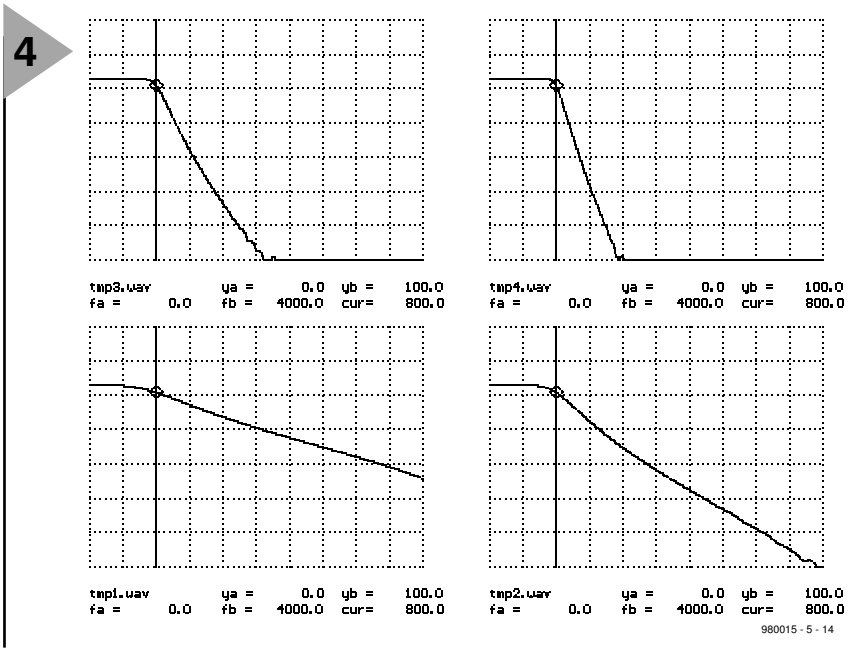
GEFILTERTES RAUSCHEN

Als letztes wollen wir ein einfaches schmalbandiges Bandfilter konstruieren, damit Rauschen filtern und die erzeugte WAV-Datei mit dem Spektrumsanalyser analysieren. Hört man diese Datei an, fällt auf, daß das Signal zwar rauscht, aber nur in einem schmalen Frequenzbereich. Die Datei **XFILDES3.SPP** stellt ein brauchbares Bandfilter mit etwa 300 Hz Bandbreite bei einer Mittenfrequenz von 1150 Hz zur Verfügung.

BUTTERWORTH-FILTER

Wer die oben vorgestellten FIR-Filter verwendet, um sehr steilflankige und schmalbandige Filterfunktionen zu realisieren, bemerkt schnell den hohen Rechenaufwand. Für Tiefpässe stellen wir nun noch das Programm **BUTTER1.PAS**, mit dem man steilflankige rekursive (IIR) Tiefpässe in die Simulationen einbeziehen kann, die wesentlich schneller als ähnlich steilflankige FIR-Filter sind und ihren analogen Verwandten entsprechen. Mit der Filterordnung (hier muß sie geradzahlig sein) stellt man die Flankensteilheit ein. Als zweiter Parameter ist die Grenzfrequenz anzugeben. In **Bild 4** sind die Frequenzgänge für die Grenzfrequenz 800 Hz bei einer Samplerate von 11.025 Samples/s durch die Datei **XBUTTER2.SPP** dargestellt, und zwar für die Ordnungen 2,4,8 und 12. Der Frequenzgang ist bis zur Grenzfrequenz flach und zeigt kein Überschwingen, dann nimmt die Dämpfung immer mehr zu. In **Bild 5**, erzeugt mit **XBUTTER1.SPP**, ist die Filterantwort auf den Sprung für die 2., 4. und 8. Ordnung zu sehen. Die Höhe des Überschwingens nimmt mit steigender Filterordnung genauso zu

Bild 4. Butterworth-Frequenzgang bei verschiedenen Ordnungen.



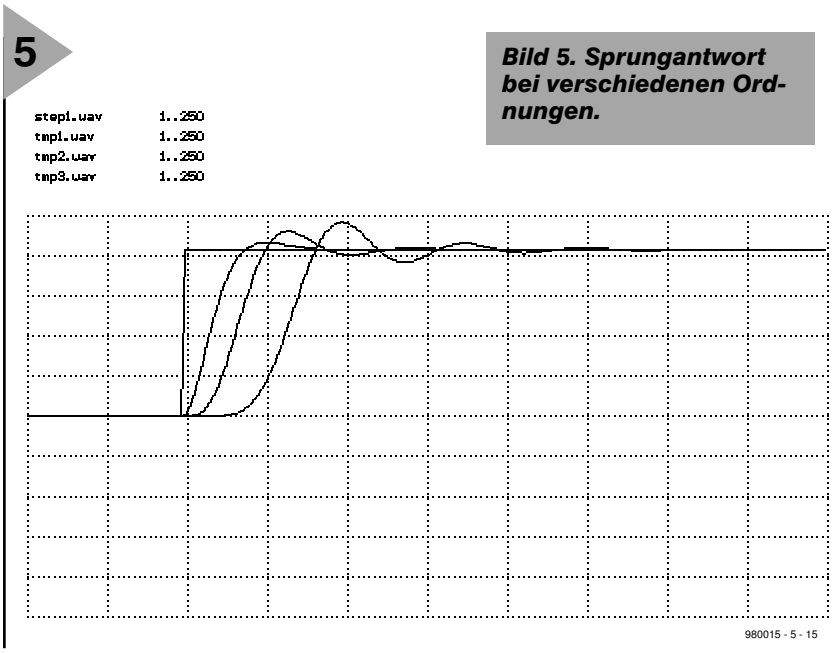


Bild 5. Sprungantwort bei verschiedenen Ordnungen.

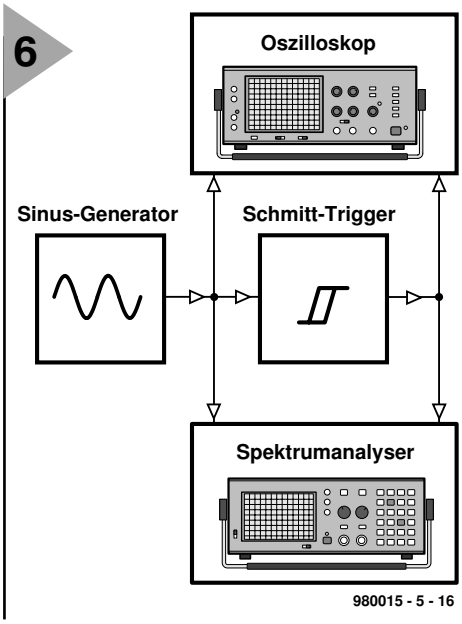
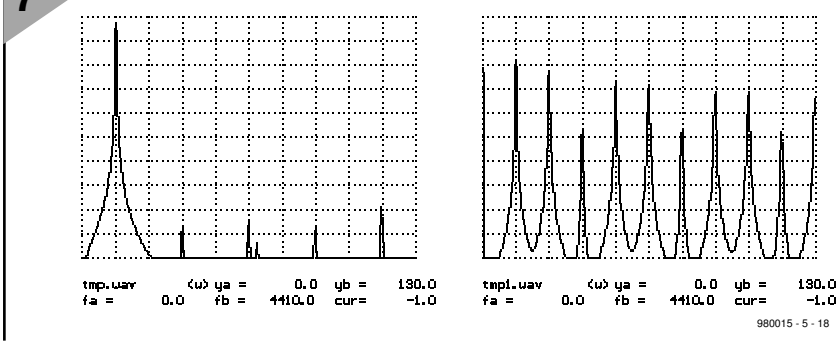


Bild 6. Mit dem Schmitt-Trigger wird ein Rechtecksignal erzeugt.

Bild 7. Spektrum von Sinus und Rechteck.



wie die Reaktionszeit.
Es gibt wie in der analogen Filtertechnik Tschebycheff- und einige andere Filter-Funktionen (auch als Hoch- oder Bandpässe) in der digitalen Signalverarbeitung. Sie hier zu diskutieren, würde aber den hier gegebenen Rahmen sprengen, so daß wieder auf die Literatur verwiesen werden muß.

Periodische Signale

Bei der Betrachtung periodischer Signale mit einem Spektrumanalyser fällt auf, daß sie nur Frequenzanteile bei Frequenzen enthalten, die ganzzahlige Vielfache der Grundfrequenz sind. Das entspricht der mathematischen Tatsache, daß ein periodisches Signal in Grund- und Oberwellen zerfällt. Als Experiment sehen wir uns dazu **Bild 6** an. Ein Schmitt-Trigger (**SCHMITT1.EXE**) bildet aus einem Sinus- ein Rechtecksignal, deren Spektren in **Bild 7** (Experiment **XFOUR1.SPP**) zu sehen sind. Beim Rechteck sind die Frequenzen enthalten, die der vielfachen Frequenz des Sinussignals entsprechen. Um die Frequenzanteile eines periodischen Signals zu bestimmen, benutzt man die Fourieranalyse, die der Spektrumanalyser näherungsweise durchführt.

Fourier-Synthese

In der Programmsammlung ist auch ein Programm enthalten, das aus der Liste der Frequenzanteile (Frequenz mit Amplitude und Phase) das zugehörige Signal berechnet. In der Tabelle sind beispielhaft die Signalanteile eines Rechtecks mit Tastverhältnis 1:1 angegeben.

Rechteckspektrum
(aus XFOUR2.SPP)

Harm.	*Anteil*	
1	*10*	*1·10*
3	*3,3333*	*1/3·10*
5	*2*	*1/5·10*
7	*1,428*	*1/7·10*
9	*1,111*	*1/9·10*
11	*0,9090*	*1/11·10*
13	*0,7692*	*1/13·10*
15	*0,6666*	*1/15·10*

Bild 8 zeigt die Ergebnisse der Fouriersynthese, wobei unterschiedliche Anzahlen von Oberwellen mitberechnet wurden. Das entsprechende Experiment führt die Datei **XFOUR2.SPP** aus.

Eine Hörprobe

Das Experiment **XFMSYN2.SPP** erzeugt die Signale **TMP1.WAV** und **TMP2.WAV**, die gleiche Frequenzanteile enthalten, allerdings unterschiedliche Amplituden aufweisen. Schon die Oszillogramme in **Bild 9** machen bedeutende Rolle der Phasenbeziehungen der Frequenzanteile klar. Das erste Signal besitzt eine fast konstante Amplitude. Bei der Hörprobe kann man eine schwache Frequenzmodulation wahrnehmen. Dem Signal **TMP2.WAV** sieht und hört man hingegen eine kräftige Amplitudenmodulation an. Beim Signal **TMP1.WAV** handelt es sich übrigens tatsächlich um ein FM-Signal, das um schwachen Signalanteile reduziert wurde. Es besitzt ebenfalls ein Linienspektrum, das jedoch symmetrisch zu einer bestimmten Frequenz liegt. Man kann also nicht mehr von einer Grundwelle mit Oberwellen sprechen.

Modulationsverfahren

Die digitale Signalverarbeitung findet auch zunehmend Eingang in die Empfängertechnik. Handys beispielsweise sind ohne komplexe Modulationsverfahren gar nicht denkbar. Daher ist ein Überblick über einige Verfahren zur Modulation und Demodulation sinnvoll. Obwohl es sich dabei hauptsächlich um klassische Verfahren handelt, gibt es doch interessante Einblicke in die Zusammenhänge zwischen Spektren von Signalen.

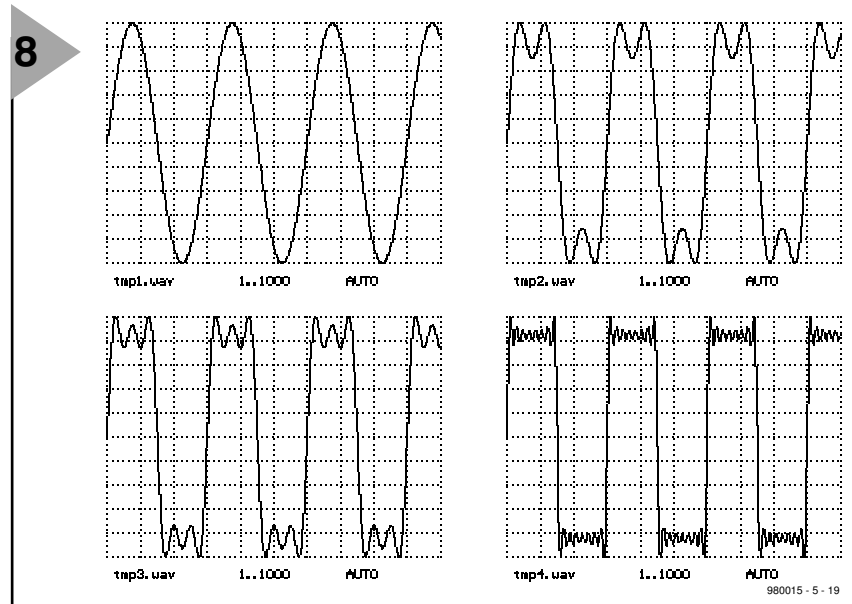
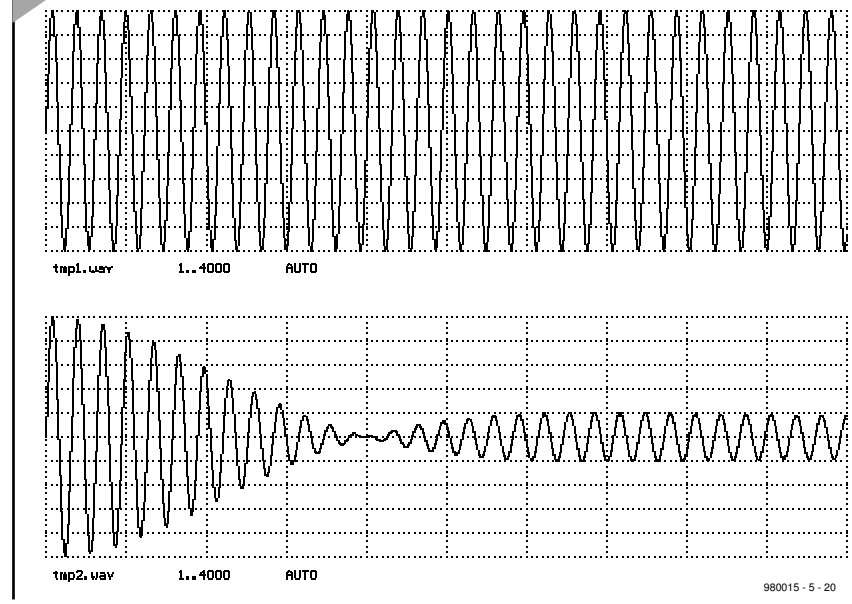


Bild 8. Fouriersynthese eines Rechtecks mit verschieden vielen Oberwellen.

Bild 9. Signale mit gleichen Amplitudenspektren, aber verschiedenen Phasenspektren.



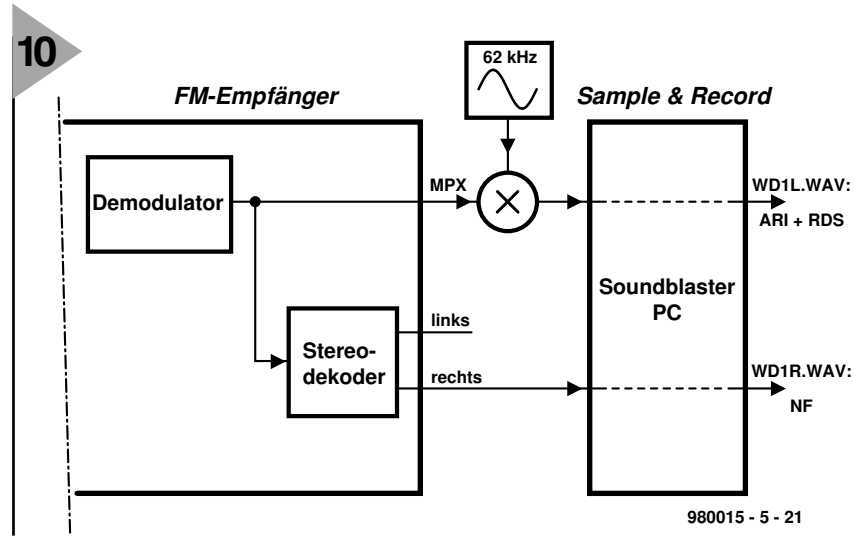


Bild 10. Aufnahme eines Rundfunksignals inklusive ARI- und RDS-Information

QUELLMATERIAL

Um den Kurs so praxisnah wie möglich zu halten, arbeiten wir mit "natürlichem" Datenmaterial. Die beiden Signale **WD1L.WAV** und **WD1R.WAV** wurden wie in **Bild 10** dargestellt gewonnen. In Deutschland senden UKW-Sender einen amplitudenmodulierten 57-kHz-Hilfsträger zur Verkehrsnachrichtenkennung (ARI). Gleichzeitig überträgt der Hilfsträger durch Phasenumtastung digitale RDS-Signale. Um dieses interessante Signal zu samplen, wurde es durch Mischung mit einem 62-kHz-Oszillator auf 5kHz ± 2kHz umgesetzt. Dieses Signal, mit dem wir AM und digitale Phasenmodulation kennenlernen, liegt in der Datei **WD1L.WAV** vor. Gleichzeitig wurde das Audiosignal in der Datei **WD1R.WAV** gesampled, und zwar an einer besonders interessanten Stelle: Um bei einer Verkehrsdurchsage den ARI-Generator vom Sender aus einzuschalten, wird in das NF-Signal der sogenannte Hintz-Triller, ein FM Signal, integriert. Genau zu diesem Zeitpunkt wurde die Datei aufgezeichnet. Man kann deutlich den Hintz-Triller hören und seine FM-Struktur studieren und gleichzeitig feststellen, wie sich die ARI-AM in der Datei **WD1R.WAV** ändert. Die Analyse dieser beiden Dateien hält allerhand an interessanten Entdeckungen bereit.

BBC: AM UND PM

Um auch die Amplitudenmodulation etwas interessanter zu studieren, wurde eine Signalsequenz des BBC-Senders auf 198 kHz aufgezeichnet. Die Trägerfrequenz wurde dabei durch Mischung mit 188 kHz auf 10 kHz herabgesetzt (**Bild 11**), was mit 44,1 kHz gut zu samplen ist. So läßt sich AM-Demodulation an einem echten NF-Beispiel testen. Interessanterweise wird bei der BBC dieser Sender zusätzlich zur Übertragung von Steuerinformationen phasenmoduliert. Auch dies ist in der Datei **BBC188.WAV** aufgezeichnet.

AMPLITUDEN-MODULATION

Bei der Amplitudenmodulation wird die Amplitude eines sinusförmigen Trägersignals durch das Modulationssignal **s(t)** gesteuert. Bei einem cosinusförmigen Träger mit der Trägerfreqenz f_c (das heißt $\omega_c = 2\pi f_c$) ist das gesendete Signal

$x(t) = [c + m \cdot s(t)] \cdot \cos(\omega_c t)$

Dabei ist **c** die Grundamplitude und **m** der Modulationsgrad.
Üblicherweise ist die Trägerfrequenz

groß gegenüber den in **s(t)** vorkommenden Frequenzen. Unsere Simulationen verwenden verhältnismäßig niedrige Trägerfrequenzen im Bereich von 10 kHz. Damit kommen als modulierende Signale nur relativ niederfrequente Signale in Betracht. Im in **Bild 12** dargestellten Experiment **XAM1.SPP** moduliert ein dreieckförmiges Signal mit 150 Hz den sinusförmigen Träger von 2 kHz. **Bild 13** zeigt links das Dreiecksignal **s(t)**, daneben die resultierende AM. Die Spektren des Dreieck- und des AM-Signals sind in **Bild 14** dargestellt. Man erkennt, daß beim AM-Spektrum links und rechts vom Träger die Spektralkomponenten von **s(t)** auftauchen. Wie kommen diese Seitenbänder zustande? Bei einem angenommenen cosinusförmigen Signal **s(t)** läßt sich das Spektrum leicht mit den Additionstheoremen berechnen:

$$[c + m\cdot\cos(\omega_m t)]\cdot\cos(\omega_c t) = c\cdot\cos(\omega_c t) + m/2\cdot\cos[(\omega_c-\omega_m)t] + m/2\cdot\cos[(\omega_c+\omega_m)t]$$

Das Signal laßt sich also in drei einzelne Cosinussignale zerlegen, die Trägerfrequenz und zwei Spektrallinien im Abstand der Modulationsfrequenz von der Trägerfrequenz. Man kann sich modulierende Signal aus vielen Cosinusschwingungen zusammengesetzt vorstellen:

$$s(t) = a_0\cdot\cos(\omega_0 t) + a_1\cdot\cos(\omega_1 t) + a_2\cdot\cos(\omega_2 t) + \ldots$$

Jeder Summand sorgt für je eine Spektrallinie links und rechts vom Träger. So entstehen die Seitenbänder. Um amplitudenmodulierte Signale untersuchen zu können, gibt es in der Programmsammlung ein Programm (**AMGEN1.EXE**) zur Erzeugung von AM-Signalen, mit dem auch das oben dargestellte Experiment durchgeführt wurde.

Im nächsten und letzten Teil dieses Kurses wenden wir uns dann der Demodulation und weiteren Modulationsverfahren zu.

Literatur:

[1] Digital Processing of Signals, Rader C.M. Gold,B.,McGraw-Hill,New York
[2] Lücker,R., Grundlagen digitaler Filter, Springer 1980

11

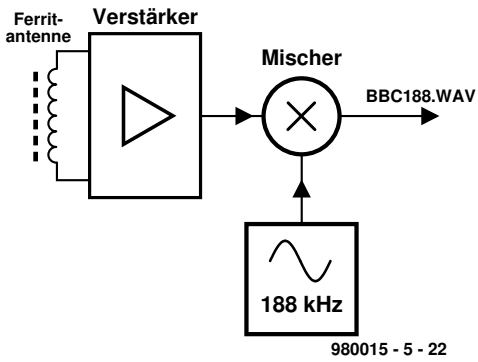


Bild 11. Aufnahme eines AM-Rundfunksignals.

12

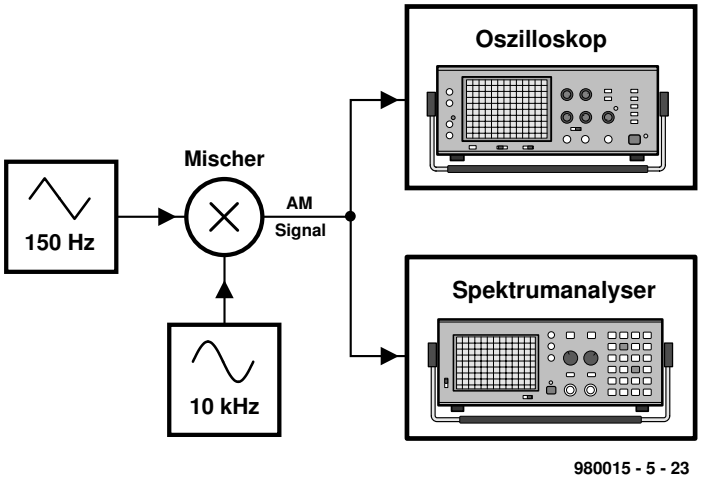


Bild 12. Erzeugung eines AM-Signals

13

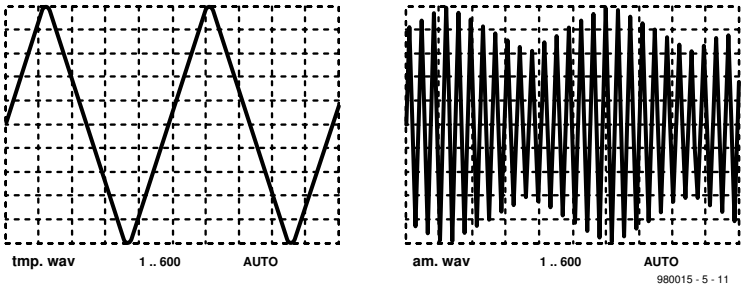
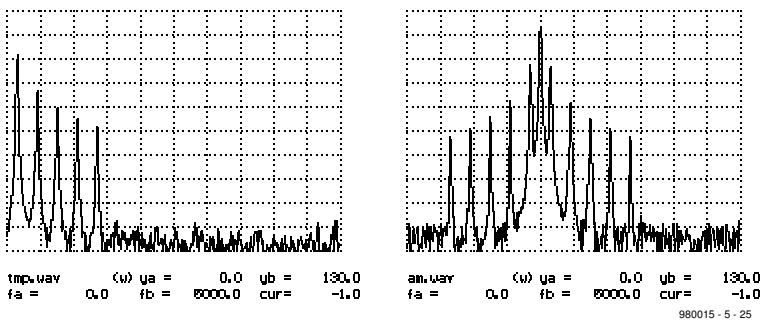


Bild 13. Wellenformen bei AM.

Bild 14. Spektrum bei AM.

14



Digital Signal Processing

Bis zum Ende: Modulation und Demodulation

In diesem sechsten und letzten Teil des DSP-Kurses geht es wiederum um Modulation und Demodulation. Mit dem erworbenen Wissen dürfte es Ihnen auch leichtfallen, weitere auf der Kurs-CD-ROM vorhandene Experimente ohne Anleitung durchzuführen.

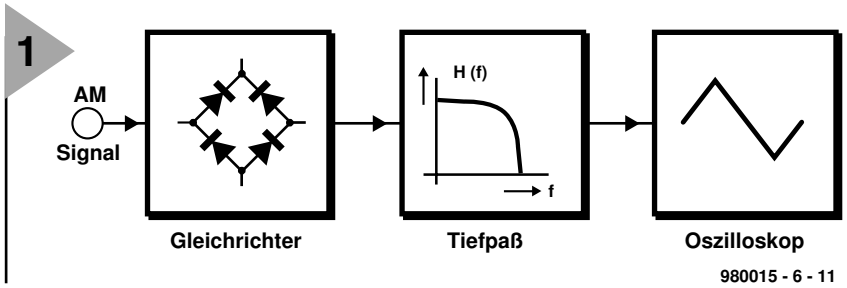


Bild 1. Prinzip der Hüllkurvendemodulation

HÜLLKURVEN-DEMODULATION

AM-Signale lassen sich am einfachsten demodulieren, indem man ihre Hüllkurve bildet. Dies geschieht durch Gleichrichtung und anschließende Tiefpaßfilterung (**Bild 1**). Um das im letzten Kursteil vorgestellte AM-Signal zu demodulieren, verwenden wir die Experiment-Datei **XDEMOD3.SPP**. Das Resultat ist in **Bild 2** dargestellt. Die Tiefpaßfilterung ist notwendig, um die unerwünschten spektralen Anteile nach der Gleichrichtung zu unterdrücken.

BBC UND MEHR

Mit dem Hüllkurvendetektor läßt sich auch das BBC-Signal **BBC188.WAV** mit 10-kHz-ZF demodulieren. Dafür sorgt **XDEMOD2.SPP**, das Resultat können wir uns als Datei **TMP4.WAV** anhören. Bei der Betrachtung des Spektrums der Datei **BBC188.WAV** (Experiment **XDEMOD2B**) fällt neben dem Peak bei 10 kHz, dem Träger der BBC, ein weiterer starker Peak bei 5 kHz auf. Als lokaler Oszillator zur Aufnahme von **BBC188.WAV** wurde 188 kHz verwendet, so daß sich eine ZF von 10 kHz für BBC auf 189 kHz ergibt, aber andererseits eine 5-kHz-ZF für 183 kHz oder 193 kHz. Und auf 183 kHz liegt ein weiterer Sender, nämlich Europe 1. Demoduliert man die Datei **BBC188.WAV** mit einer ZF von 5 kHz (**XDEMOD2A.SPP**), ist tatsächlich dieser Sender in französischer Sprache zu hören. Und bei einer ZF von 19 kHz liegt sogar noch ein Sender (Sendefrequenz 207 kHz, Experiment **XDEMOD2C.SPP**).

SYNCHRON-DEMODULATION

Es gibt noch weitere Verfahren, um ein AM-Signal zu demodulieren. Bei der Diskussion der Amplitudenmodulation war zu sehen, daß bei der Multiplikation eines Signals mit dem Sinussignal eines lokalen Oszillators (LO) das Spektrum um die Frequenz des lokalen Oszillators verschoben wird. Genauso kann man das Spektrum durch eine nochmalige Multiplikation wieder zurückverschieben. Das Experiment **XDEMOD5.SPP** ist in **Bild 3** dargestellt, das Resultat in **Bild 4**. Verwendet man als LO ein Sinussignal, also das gleiche Signal, das als Träger zur Modulation verwendet wurde, so kann man das Signal durch Multiplikation und Tiefpaßfilterung wiedergewinnen (Bild 3 und **Bild 4** oben). Verwendet man als LO ein um 90 Grad phasenverschobenes Signal, so bleibt nach der Tiefpaßfilterung kein Signal mehr übrig. Zur

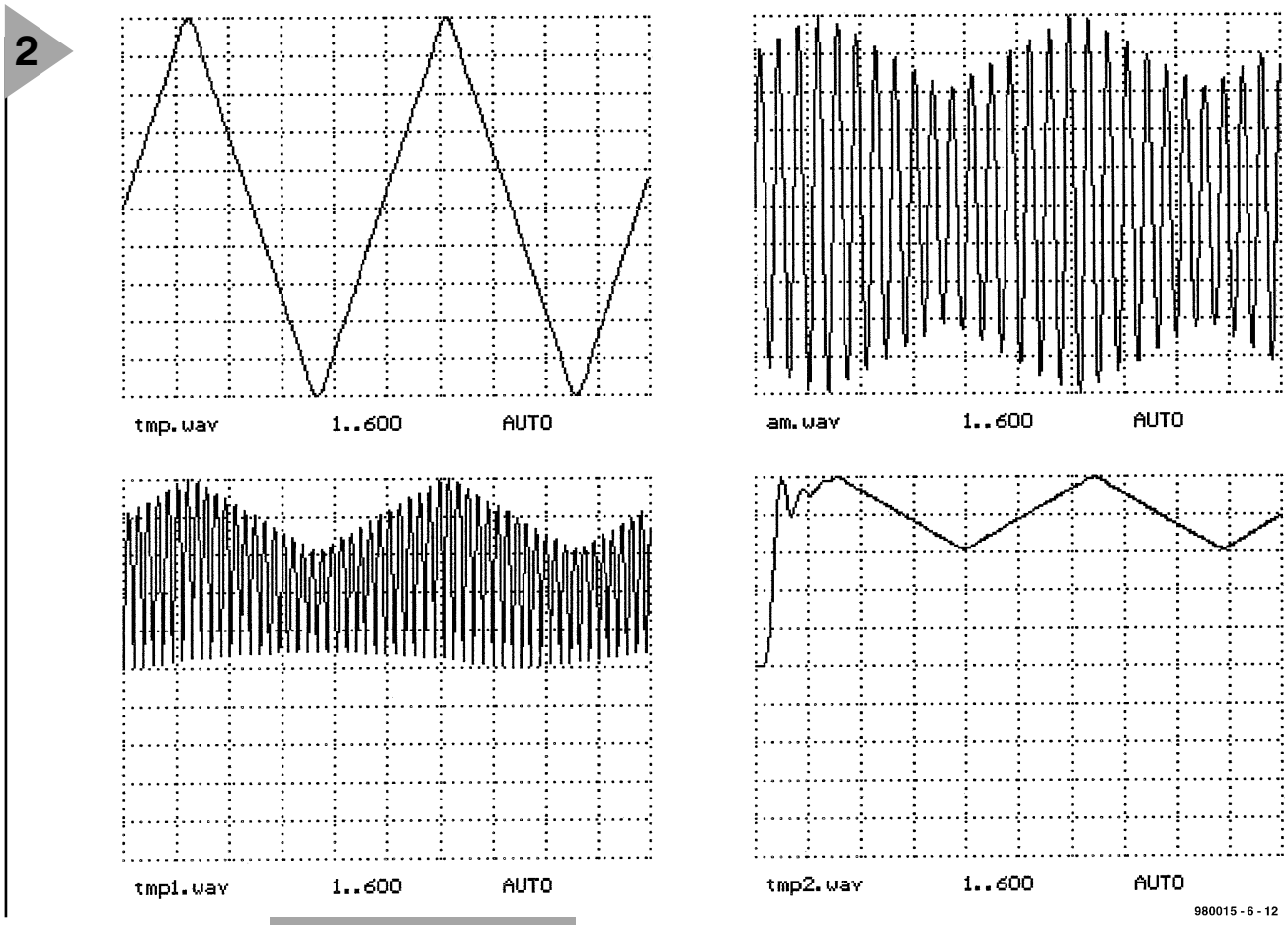


Bild 2. Signale bei AM Modulation und Demodulation

Demodulation muß der LO also in Phase mit dem Träger das AM-Signals sein. Man spricht daher auch von synchroner beziehungsweise kohärenter Demodulation. Wenn der LO nicht genau die gleiche Frequenz hat wie der Träger, wechseln sich die beiden Situationen *In Phase* und *90 Grad Phasenverschiebung* im Rhythmus der Differenzfrequenz ab, die Lautstärke schwankt. Daher muß bei einem Synchrondemodulator ein Schaltkreis, zum Beispiel eine phasengeregelte Schleife im Empfänger dafür sorgen, daß der LO immer die richtige Frequenz und Phase besitzt. Diese Art der Demodulation ist übrigens bei AM zwar aufwendig, liefert aber die optimalen Resultate.

HOCHAUFLÖSENDER SPEKTRUMANALYSER

Wir haben gerade gesehen, daß mit einem AM-Generators beziehungsweise durch Multiplikation mit einem Sinussignal das Spektrum eines Signals verschoben werden kann. Dies ist das Grundprinzip jedes Empfängers, wenn er das Eingangssignal in die Zwischenfrequenz umsetzt. Durch den gleichen Trick kann man die Auflösung unseres Spektrumanalyser-Programms **SPEC1** heraufsetzen. Bei einer Abtastung des Originalsignals mit 22.050 Samples/s erhält man bei einer DFT-Länge von 4096 Punkten je Punkt einen Frequenzabstand von 5,3833 Hz. Um einen Bereich des Spektrums genauer anzusehen, können wir diesen Frequenzbereich durch Mischen in den tieffrequenten Bereich verschieben und anschließend eine Tiefpaßfilterung und Downsampling anwenden.

In der Praxis funktioniert das so: Um das Signal **WD1L.WAV** in der Nähe von 5 kHz hoch aufzulösen, bei 4096 DFT-Punkten eine Auflösung von etwa 0,5 Hz zu erzielen, müssen wir von 22.050 Samples/s um den Faktor 10 downsamplen. Mit 2205 Samples/s lassen sich Frequenzen bis 1,1 kHz darstellen. Das 5-kHz-Signal aus **WD1L.WAV** wird in diesen Bereich gebracht, indem wir 5 kHz durch Mischung mit 5,3 kHz auf 300 Hz heruntermischen. Wir starten mit 40.960 Samples, damit danach 4096 Samples übrigbleiben. Dies erledigt die Datei **XSPEC2.SPP**, das Resultat ist in **Bild 5** zu sehen. Man erkennt den Träger und zwei Signale links und rechts vom Träger (ursprünglich 57 kHz). Das bei 125 Hz Abstand ist die ARI-Durch-

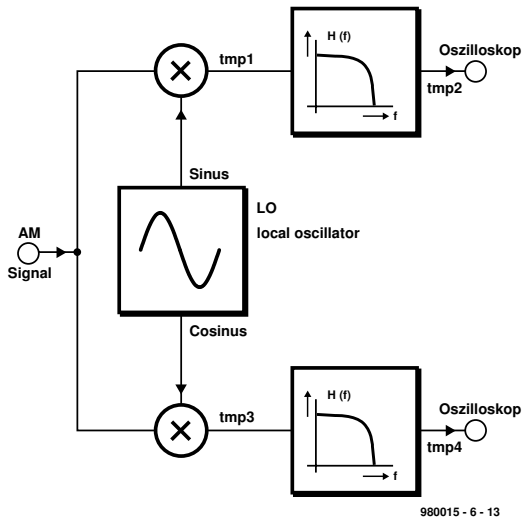


Bild 3. Synchrone Demodulation mit Original- und phasenverschobenem Träger.

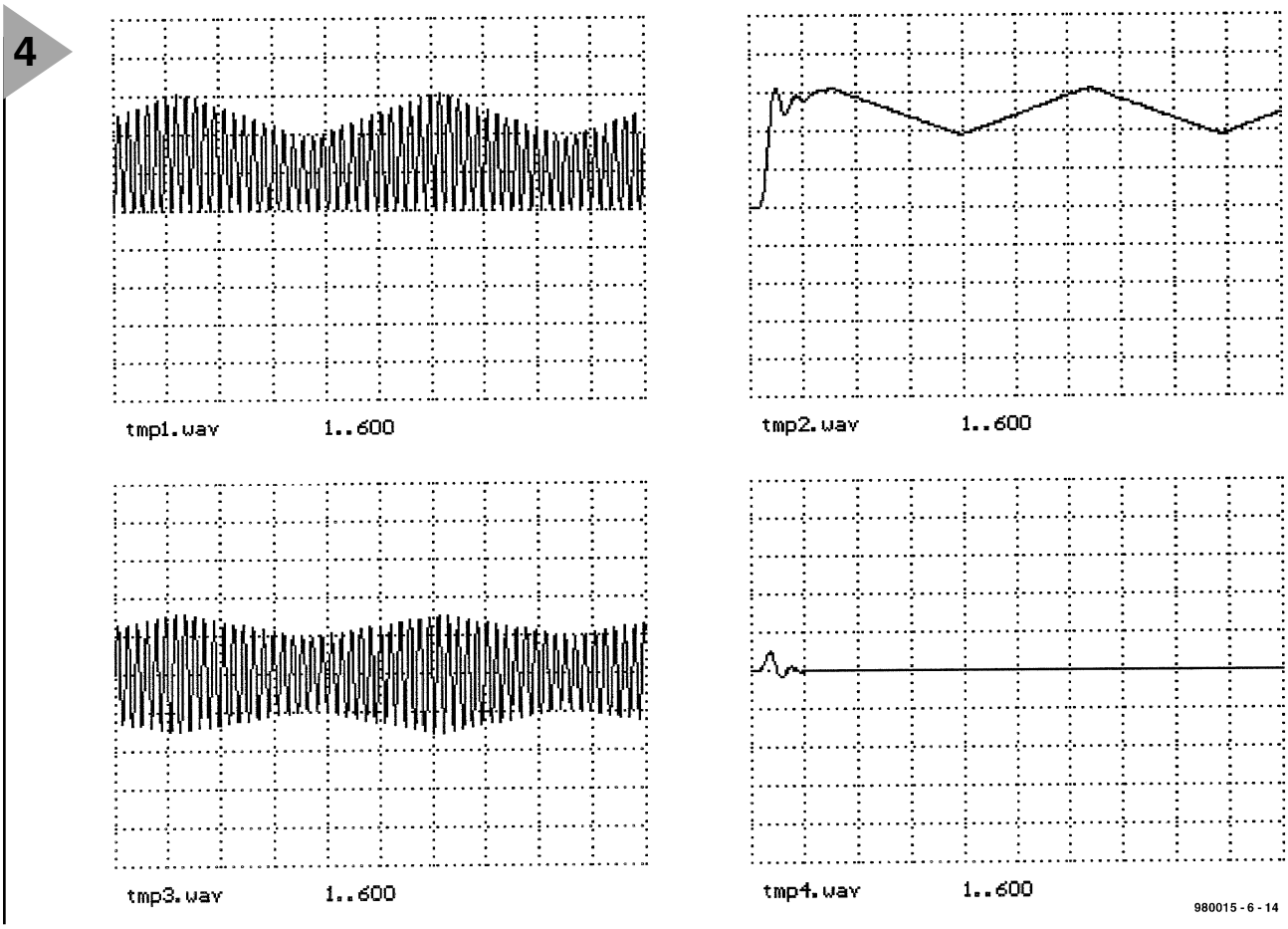
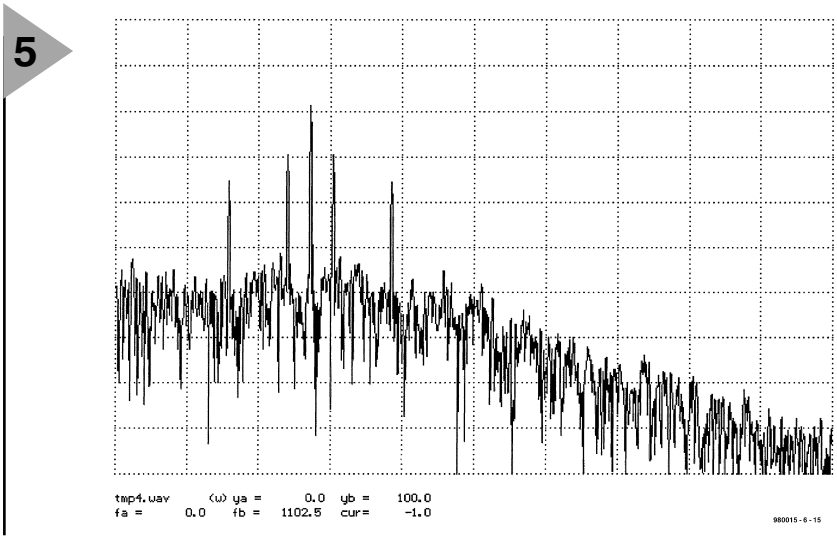


Bild 4. Synchrone Demodulation (oben) und synchrone Demodulation bei um 90 Grad verschobenem LO (unten).

sagekennung, die eingeschaltet wird, wenn gerade eine Verkehrsdurchsage läuft. Das andere Signal weist einen Abstand von 34,93 Hz Abstand zum Träger auf. Es handelt sich um die sogenannte Bereichskennung, die mitteilt, in welcher Gegend der gerade empfangene UKW-Sender liegt. Betrachtet man übrigens ein Stück am Anfang der Datei **WD1L.WAV**, wenn die Durchsage noch nicht läuft, findet man die 125-Hz-Spitzen nicht, da die Durchsage-Kennungs-AM dann noch nicht aktiviert ist.

Bild 5. Hochaufgelöstes Spektrum des 57-kHz-ARI-Hilfsträgers.



QUADRATUR-MODULATION UND DEMODULATION

Die Behandlung der Synchrondemodulation zeigte, daß bei der Demodulation der LO zur Demodulation in Phase mit dem Träger sein muß, wohingegen ein um 90 Grad phasenverschobener LO kein Ausgangssignal hervorruft. Das kann man sich technisch zunutze machen, indem man ein AM-Signal mit einem Sinussignal als Träger und bei gleicher Trägerfrequenz ein Cosinussignal als Träger eines zweiten Signals benutzt. Diese beiden Signale kann man dann über den gleichen Kanal übertragen (**Bild 6**) und dann durch Demodulation mit den entsprechenden LOs wieder getrennt demodulieren, wie es das Experiment **XDEMOD4.SPP** im dargestellten Aufbau simuliert. Als Signal 1 wird ein Dreieck- eingesetzt, als Signal 2 ein Sinussignal. Modulationsverfahren, die gleichzeitig Sinus und Cosinus als Träger verwenden, nennt man wegen des häufigen Vorkommens von 90 Grad in diesen Systemen Quadraturverfahren. Der Hauptträger mit seinem Signalteil wird oft mit dem Buchstaben I für In-Phase bezeichnet, den anderen Teil nennt man Quadratur-Teil Q, weil er *In Quadratur*, das heißt in 90 Grad zum anderen Kanal steht.

RDS DEMODULATION

Eine praktische Anwendung eines solchen Quadraturverfahrens findet man

beim UKW-Rundfunk. Für Ein/Ausschalten des Autoradios bei Verkehrsdurchsagen wurde bereits relativ früh ein 57-kHz-Hilfsträger verwendet, der niederfrequent amplitudenmoduliert wird, zum Beispiel mit 125 Hz, wenn gerade eine Durchsage läuft. Dies bildet den I-Teil. Ab 1980 wollte man dann zusätzlich digitale Daten übertragen (RDS, Radio Data System). Damit sich ARI und RDS nicht gegenseitig beeinflussen, wählte man für RDS einen um 90 Grad versetzten Hilfsträger. Die RDS-Daten werden also im Q-Teil übertragen. Das Experiment **XRDS2.SPP** simuliert den ARI- und RDS-Demodulator nach **Bild 7**. Die Phase des LO wurde von Hand so eingestellt, daß sie im Bezug zum Träger richtig ist. Normalerweise sorgt dafür ein Phasenregelkreis (PLL). In **Bild 8** sind die demodulierten Signale zu erkennen. Das obere Signal repräsentiert die sinusförmige Bereichskennungs-AM (34.93 Hz), während unten das digitale RDS-Signal abgebildet ist.

FREQUENZ-MODULATION

Bei der Amplitudenmodulation wird des Spektrum des modulierenden Signals im Prinzip einfach durch den Träger im Frequenzbereich verschoben. Bei der Frequenzmodulation dagegen beeinflußt das modulierende Signal **s(t)** die Frequen des Trägers. Gesendet wird das Signal **y(t)** nach der Vorschrift

$$y(t) = \cos[2\pi f_c + \mu\, s(t)]$$

Genau genommen betrachten wir damit die FM als Abart der Phasenmodulation. Nun gibt es keinen einfachen Zusammenhang mehr, um das Spektrum des gesendeten Signals zu berechnen. Selbst für ein sinusförmiges Signal wird das Spektrum schon ziemlich komplex. Es gilt dann

$$y(t) = \cos[2\pi f_c\, t + \mu \cos(2\pi f_m\, t)]$$

In **Bild 9** sind die Spektren für f_c = **2000 Hz** und die Werte von μ = **2,4048** beziehungsweise μ = **4,0** und f_m = **100** beziehungsweise f_m = **200** dargestellt (**XFM1.SPP**). Man sieht, daß einzelne Spektrallinien im Abstand von f_m um den Träger herum auftauchen. Der Träger muß dabei selbst nicht einmal im Signal vorhanden sein. Die Stärke der einzelnen Linien kann man mit Hilfe der sogenannten Besselfunktionen ausrechnen. Üblicherweise ist ein FM-Spektrum breiter als das Spektrum des modulierenden Signals, wie dies auch bei klassischen FM-Aussendungen anzutreffen ist. Obwohl die NF-Bandbreite nur 15 kHz beträgt, belegt ein FM-Spektrum ±150 kHz um den Träger herum. Genau diese größere Bandbreite macht ein FM-Signal störsicher, da schmalbandige Störer nur kleine Teile des Spektrums verfälschen. Diese benötigte sogenannte *Carson-Bandbreite* berechnet man nähe-

6

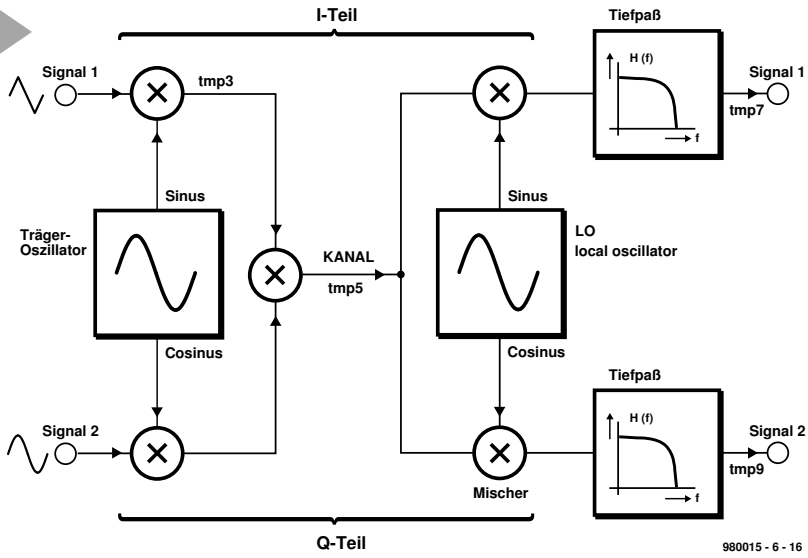


Bild 6. Aufbau eines System bei Quadraturmodulation.

Bild 7. So werden ARI und RDS demoduliert.

7

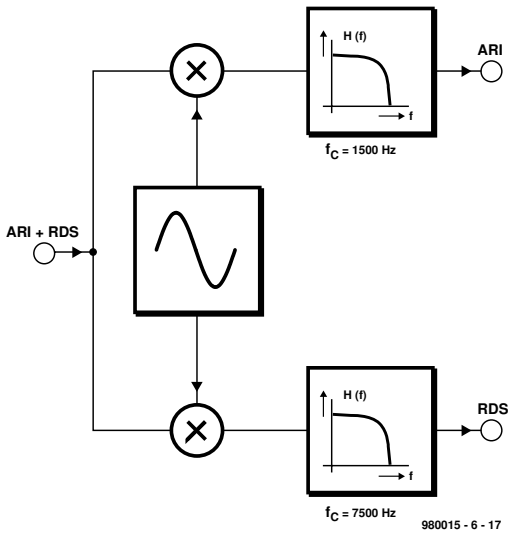
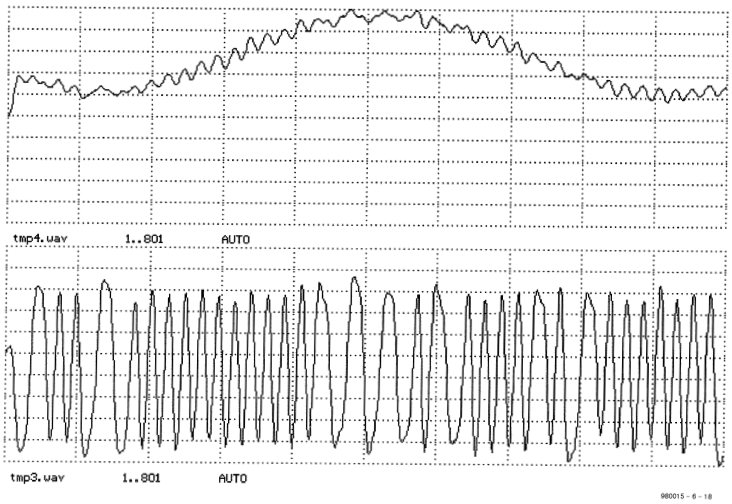


Bild 8. Demodulation von ARI (oben) und von RDS (unten).

8



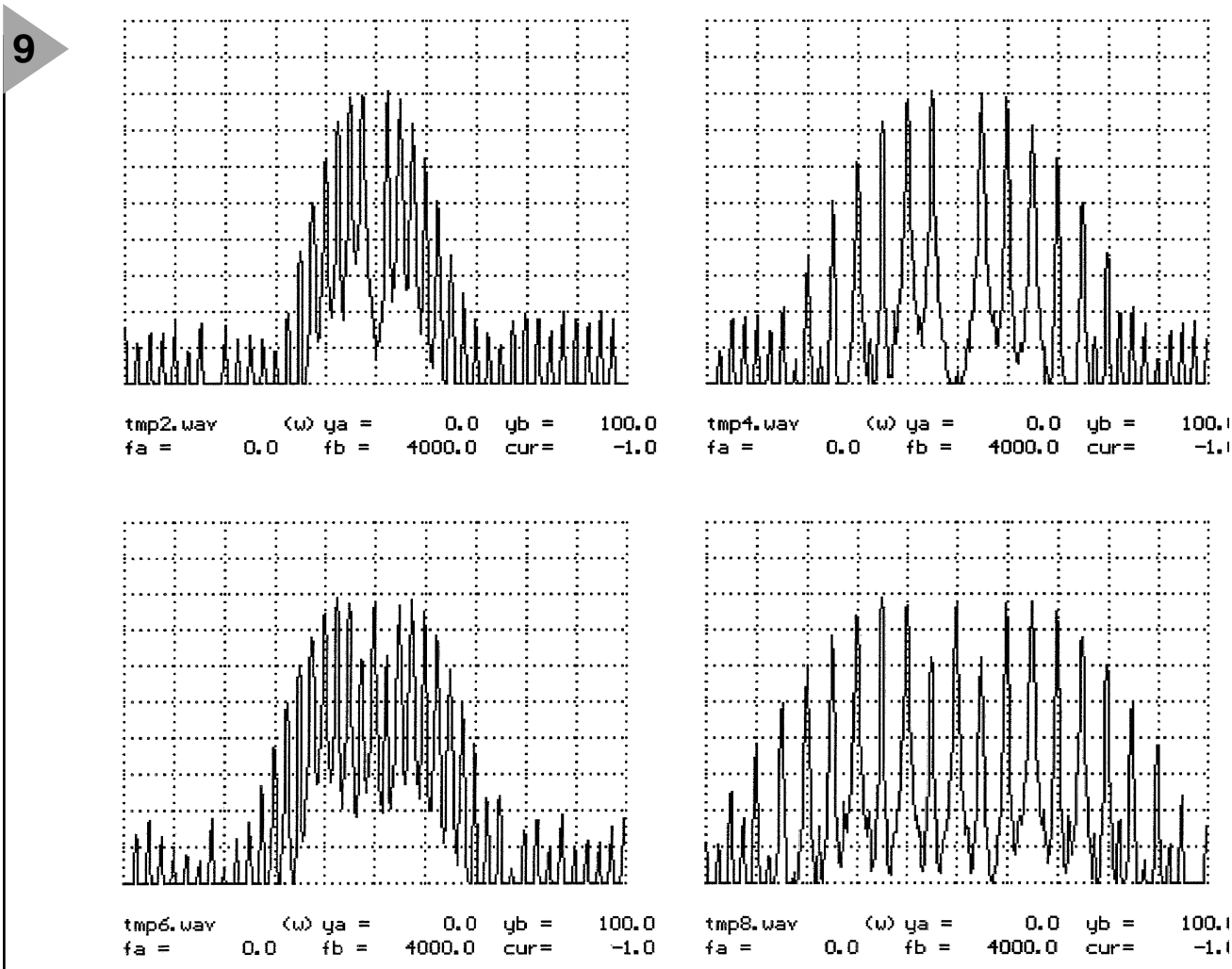


Bild 9. FM-Spektren bei sinusförmigem modulierenden Signal.

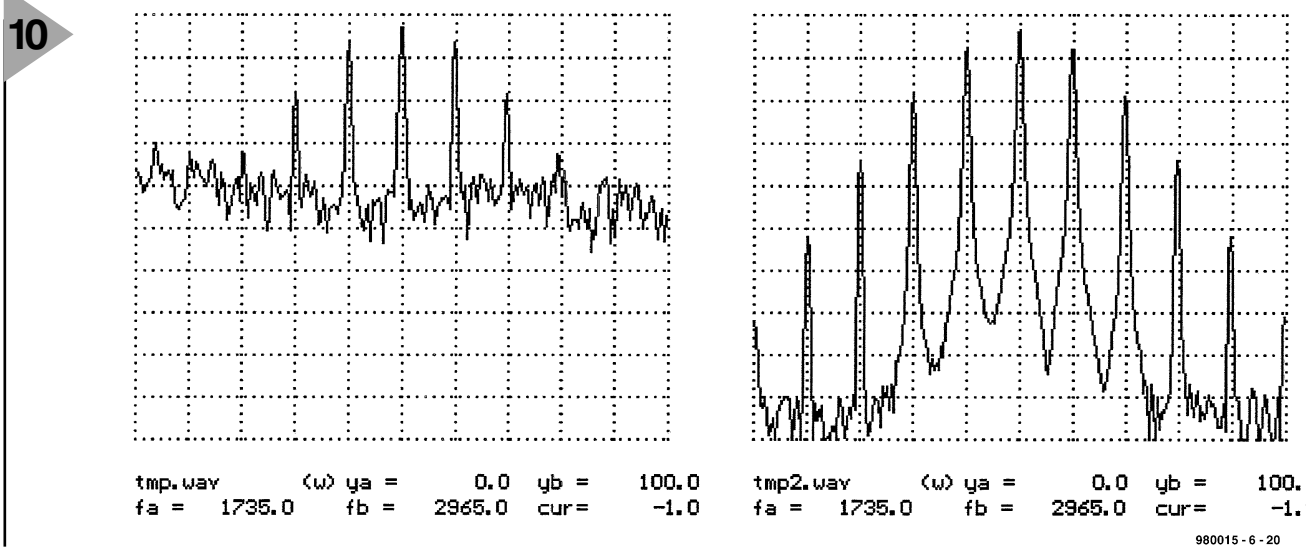
rungsweise aus der Bandbreite und Energie des modulierenden Signals.

HINTZ-TRILLER

Auf der Kurs-CD-ROM ist auch ein Beispiel für ein einfaches FM-Signal mit sinusförmiger Modulation zu finden. Wie schon im letzten Kursteil erwähnt, wird in Deutschland bei der ARD der sogenannte Hintz-Triller verwendet, um den Beginn eines Verkehrshinweises zu anzuzeigen. Der Hintz-Triller besitzt einen Träger von 2350 Hz, der mit 123 Hz frequenzmoduliert wird, und zwar mit einem Modulationsindex von μ = **1**. Die Dauer des Signals beträgt 1,2 Sekunden.

Hören Sie sich die Datei **WD1R.WAV** an und versuchen Sie kurz vor der Durchsage, den Triller wahrzunehmen! Sieht man sich nun die Samples von 31000 bis 39000 mit dem Spektrumsanalyser (**XHINTZ4.SPP**) genauer an, erkennt man in **Bild 10** links die spektralen Linien im Abstand von 123 Hz um 2350 Hz herum. Das gleiche Signal läßt sich auch selbst erzeugen, das Spek-

Bild 10. FM-Spektrum des Hintz-Trillers.



trum mit den gleichen Spektrallinien ist in Bild 10 rechts zu sehen. Dabei sind auch Linien mit höherem Abstand erkennbar, die im Fall des ARD-Signals in der restlichen NF untergehen.

FUNKFERNSCHREIBEN, FSK

Beim Funkfernschreiben wird oft die sogenannte Frequenzumtastung (FSK = Frequency Shift Keying) verwendet, um Fernschreibsignale zu übertragen. Ein Fernschreibsignal (**Bild 11**) besitzt einen Ruhepegel, den man mit **SPACE** bezeichnet. Er liegt an, wenn der Fernschreiber nicht aktiv ist. Ein einzelnes Zeichen startet dann mit dem **STARTBIT**, bei dem das Signal in den Zustand **MARK** übergeht. Das Startbit dauert beispielsweise bei einer Übertragungsrate von 50 Baud 1/50 s = 20 ms. Danach folgen die einzelnen Bits des zu übertragenden Zeichens. Beim alten, teilweise noch verwendeten Baudot-Code sind es fünf Zeichen, wobei jedes Bit genauso lange wie das Startbit dauert. Dann folgt ein **STOPBIT** im Zustand **SPACE**. Es folgt eine beliebig lange Pause (asynchrone Übertragung), dann geht es für das nächste Zeichen wieder von vorne los. Beim FSK-Verfahren ordnet man den beiden Zuständen **MARK** und **SPACE** einfach zwei Frequenzen (etwa 850 Hz für **MARK** und 1350 Hz für **SPACE**) zu und

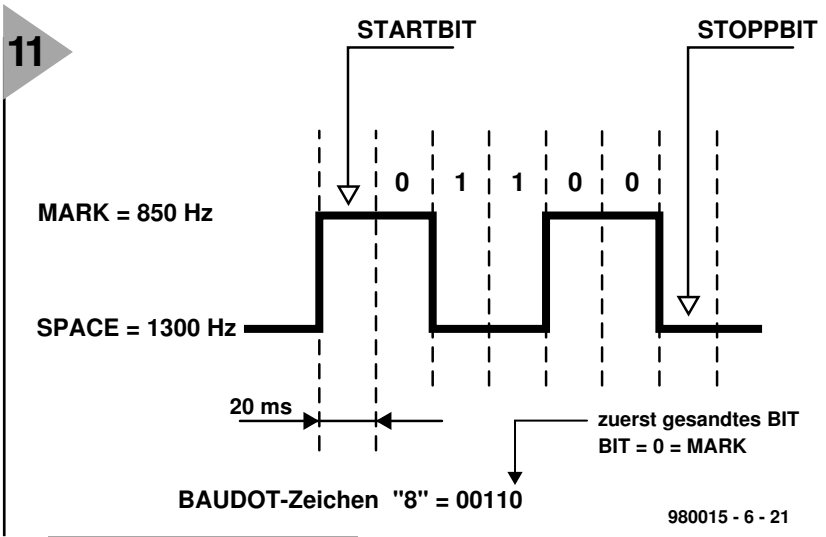


Bild 11. Signal bei Funkfernschreiben.

Bild 12. Aufbau des Filterdekoders für Funkfernschreiben.

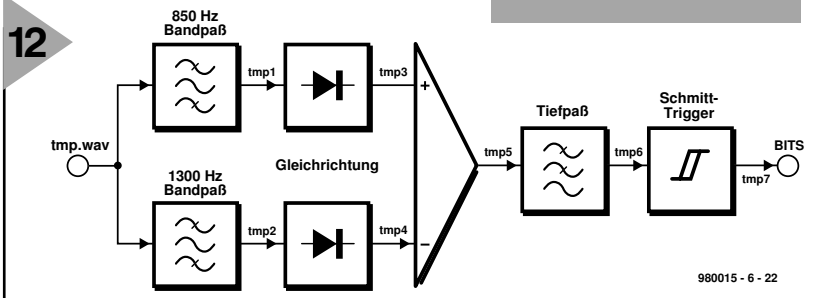
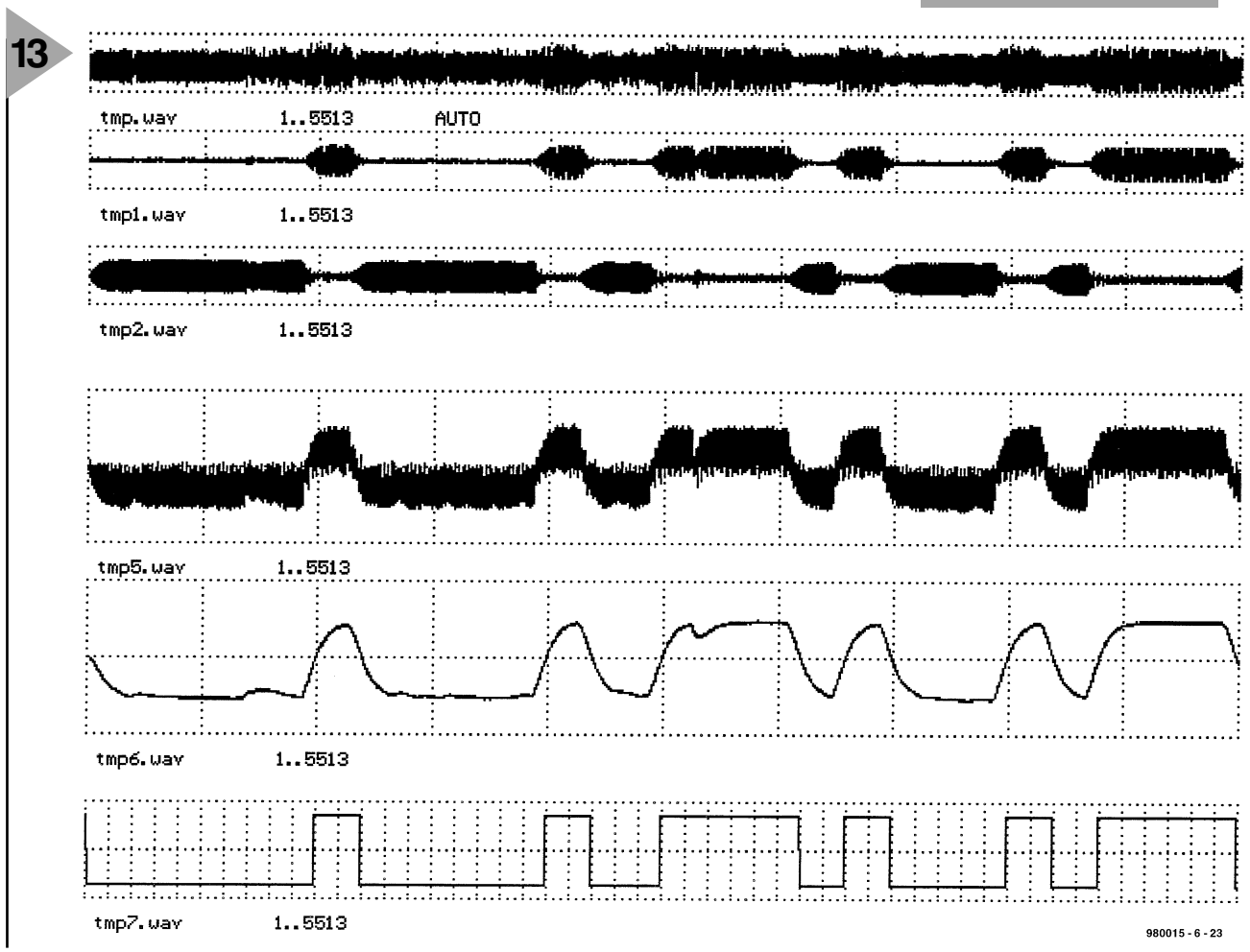


Bild 13. Signale beim Filterdekoder.



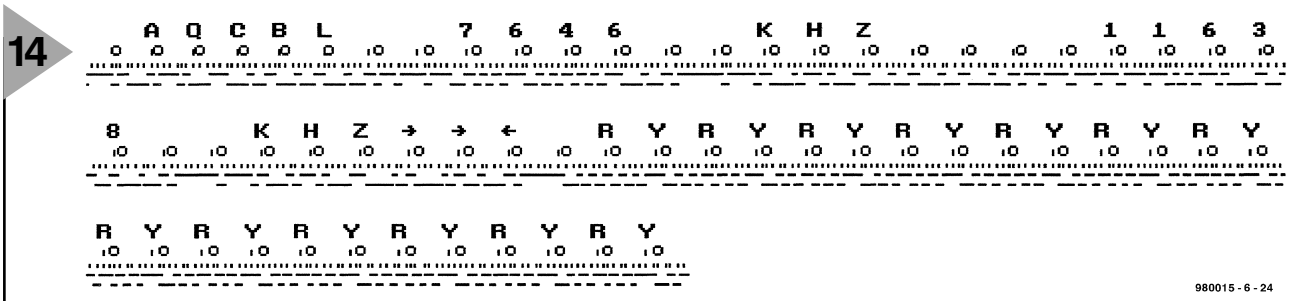


Bild 14. Dekodiertes Funkfernschreibsignal.

sendet diese beim Vorliegen des entsprechenden Zustandes zum Beispiel mit einem spannungsgesteuerten Oszillator (VCO) aus. Den Abstand zwischen den beiden Frequenzen bezeichnet man auch als **SHIFT**. Die Datei **RTTY1.WAV** enthält ein solches Signal mit 50 bit/s, **MARK** = 850 Hz und **SPACE** = 1350 HZ. Mit dem Experiment **XRTTY1.SPP** kann man das Spektrum des Fernschreibsignals ermitteln. Man sieht zwei deutlich herausragende Peaks bei den beiden **SPACE**- und **MARK**-Frequenzen. Da diese Signale ganz im Bereich von 300 Hz bis 3400 Hz liegen, kann man sie gut über einen Telefonie-Funkkanal übertragen.

FUNKFERNSCHREIB-DEKODIERUNG

Um ein solches FSK-moduliertes Signal zu dekodieren, bildet man einfach den klassischen Filterdekoder (**Bild 12**) nach, wie dies im Experiment **XRTTY6.SPP** geschehen ist. Die resultierenden Kurvenformen sind in **Bild 13** dargestellt, **Bild 14** zeigt den empfangenen Klartext. Die Zeichenfolge RYRYRYR wird immer gerne als Test-Text verwendet, weil sie die schnellsten Bitwechsel enthält (**XRTTY5.SPP**).

NOCH MEHR?

Die sechs Folgen des Kurses haben schon mehr als einen kleinen Einblick in den Umgang mit digitalen Signalprozessoren gewährt. Es gibt aber sicherlich noch eine Vielzahl interessanter Themen, die noch gar nicht besprochen wurden. Einige Experimente dazu (siehe Kasten) finden sich noch zusätzlich auf der Kurs-CD-ROM, der der Autor den Ehrentitel *Elektors Signal PRocessing Experimente und Simulations SOftware (ESPRESSO)* hat zukommen lassen. Die Experimente lassen sich mit dem gewonnenen Wissen leicht in Eigenregie durchführen. Eine kleine Hilfestellung gibt die Datei **EXPS.DOC**.

(980015-VI)

Einseitenbanderzeugung(SSB)
Filtermethode und Phasenmethode
Einseitenbanddemodulation-Phasenmodulation
AMDS
BBC
Wetterfunk
Wetter-Satellitenbild
Zeitzeichen-Sendung
DTMF