DSP-Programmierung mit dem TMS320C25

Workshop zum 11. Internationalen PR-Treff am 20./21. Mai 1995

Jürgen Hasch, DG1SCR@DB0RBS

Einleitung

Der Begriff „Digitale Signalverarbeitung“ (DSV, englisch DSP) steht für die Berechnung von digitalen Zahlenwerten nach einer bestimmten Rechenvorschrift, ausgehend von gegebenen Eingangsgrößen. Die ersten Anwendungen dafür entstanden in den 50er und 60er Jahren mit der Simulation analoger Schaltungen auf Digitalrechnern. Für ein gegebenes Eingangssignal wurde (in langwieriger Rechnung) das Ausgangssignal der eingegebenen Schaltung berechnet.

Durch die enormen Fortschritte in der Mikroprozessortechnik können heutzutage viele analoge Systeme durch in Echtzeit arbeitende digitale Systeme ersetzt werden. Anwendungen wie exakt linearphasige Filter, Sprachcodierung oder Datenübertragung mit komplexen Modulationsarten, sind erst mit Hilfe digitaler Systeme effizient realisierbar. So ist beispielsweise bei modernen GSM-Telefonen die gesamte Signalverarbeitung bis zur ZF-Schnittstelle mit Hilfe Digitaler Signalverarbietung realisiert.

Einige Vorteile der DSV sind:

- Flexibilität des Anwendungsbereichs durch Umprogrammierung
- Tiefer Frequenzbereich erfaßbar
- Ersetzen eines zusätzlichen Mikroprozessors für Steuerungszwecke
- Kein Abgleich notwendig
- Langzeit- und Temperaturstabilität
- Reproduzierbarkeit der erzielten Ergebnisse
- Hohe Genauigkeit erreichbar
- Hohe Zuverlässigkeit
- Geringe Störempfindlichkeit

Natürlich kann man auch einige Nachteile aufzählen:

- Zusätzlicher Schaltungsaufwand für Analogwandler, Prozessor und Speicher
- Nur eingeschränkter Frequenzbereich (kHz-Bereich)
- Störung der analogen Signale durch den Digitalteil

Als Anwendung der Digitalen Signalverarbeitung im Amaterfunk-Bereich sind hier ein paar Beispiele aufgeführt:

- Schmalbandiges CW-Filter, das absolut frequenzstabil und in der Bandbreite umschaltbar ist
- Adaptives Filter zur Störtrager-Unterdrückung, auch von Störträgern mit sich ändernder Frequenz
- Schnelle Fouriertransformation zur Anzeige des Frequenzspektrums eines Signals
- Signalgenerator mit fast beliebigen Kurvenformen
- Modems für RTTY, PACTOR, AMTOR, FAX, Packet Radio usw.

1. Grundlagen der DSV

Der folgende kurze Abriß über die wichtigsten Grundlagen kann natürlich keine Einführung in die Digitale Signalverarbeitung bieten, soll aber zumindest ein paar der wichtigsten Begriffe vorstellen.
Ausgangspunkt ist ein sogenanntes „Echzeitsystem zur Digitalen Signalverarbeitung“ wie in Abbildung 1 gezeigt:

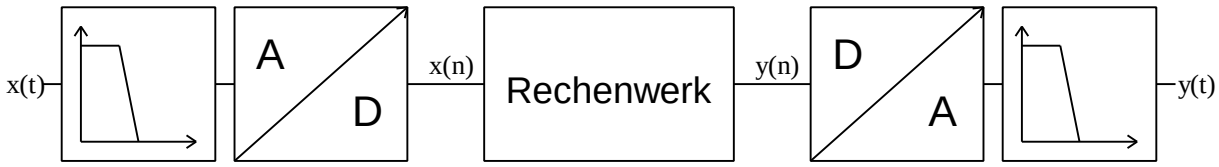


Abbildung 1

Am Eingang liegt das analoge Signal x(t) an. Es wird über einen Tiefpaß dem Analog-Digital-Wandler zugeführt, dort abgetastet und quantisiert. Am Eingang des digitalen Rechenwerks liegt dann die Wertefolge x(n) an. Das Rechenwerk (z.B. ein Signalprozessor) berechnet daraus die Ausgangsfolge y(n), die dann über einen Digital-Analog-Wandler, mit anschließendem Tiefpaß zur Glättung, in das Ausgangssignal y(t) umgesetzt wird.
Die ganze Schwierigkeit der DSV besteht nun darin, dem Rechenwerk die Umsetzung der Eingangsfolge x(n) in die Ausgangsfolge y(n) beizubringen.

Das Rechenwerk selbst kann aus festverdrahteter Logik, einem PC oder eben wie hier besprochen aus einem Signalprozessor bestehen.Voraussetzung selbst einen Signalprozessor programmieren zu können ist neben Grundkentnissen bei der Programmierung von Mikroprozessoren auch das Verständnis, wie aus dem analogen Eingangssignal digitale Zahlenwerte und ungekehrt werden und welche Konsequenzen daraus folgen.

Zuerst soll deshalb der Analog-Digital-Wandler etwas genauer betrachtet werden.
Ein A/D-Wandler besteht im Prinzip aus drei Stufen, wie in Abbildung 2 gezeigt:

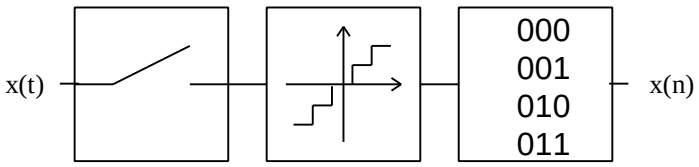


Abbildung 2

Direkt am Eingang liegt der Abtaster, der zu festen Zeitpunkten n*T_A (T_A ist der Kehrwert der Abtastfrequenz f_A) das Eingangssignal durchläßt und zum Quantisierer weitergibt. Dort wird dem analogen Wert mit unendlich vielen Stufen, ein diskreter Wert aus einem Bereich mit endlich vielen Stufen zugeordnet. Die Auflösung des Wandlers bestimmt die Anzahl der möglichen Werte. Anschließend wandelt der Codierer den Wert in das gewünschte Zahlenformat (z.B. in das Zweierkomplement) um. Aus dem analogen Eingangssignal wurde somit eine zeit- und wertdiskrete Folge.

Abtasttheorem

Der Abtastvorgang läßt sich mit Hilfe von Abbildung 3 veranschaulichen. Das links gezeigte zeitkontinuierliche Eingangssignal x(t) wird zu festen Zeitpunkten abgetastet, das entspricht einer Multiplikation mit den rechts angedeuteten Abtastimpulsen $\delta_T(t)$

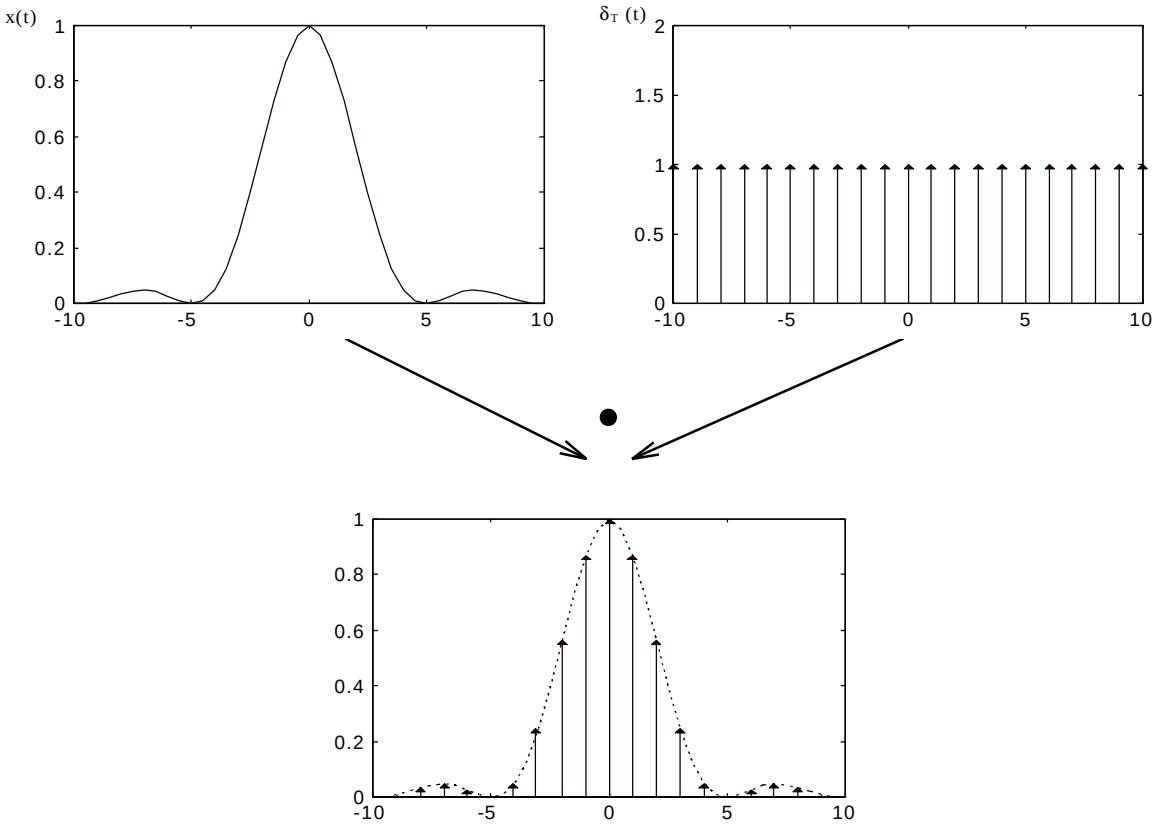


Abbildung 3

Das Ergebnis dieser „idealen Abtastung“ ist ein zeitdiskretes, aber noch wertekontinuierliches Signal.
Man kann diese Betrachtung auch im Frequenzbereich vornehmen. Dem Eingangssignal x(t) im Zeitbereich entspricht X(f) im Frequenzbereich und den Abtastimpulsen $\delta_T(t)$ im Zeitbereich entspricht $\Delta_T(f)$ im Frequenzbereich. Die Multiplikation der Signale im Zeitbereich wird dabei zur Faltung im Frequenzbereich.

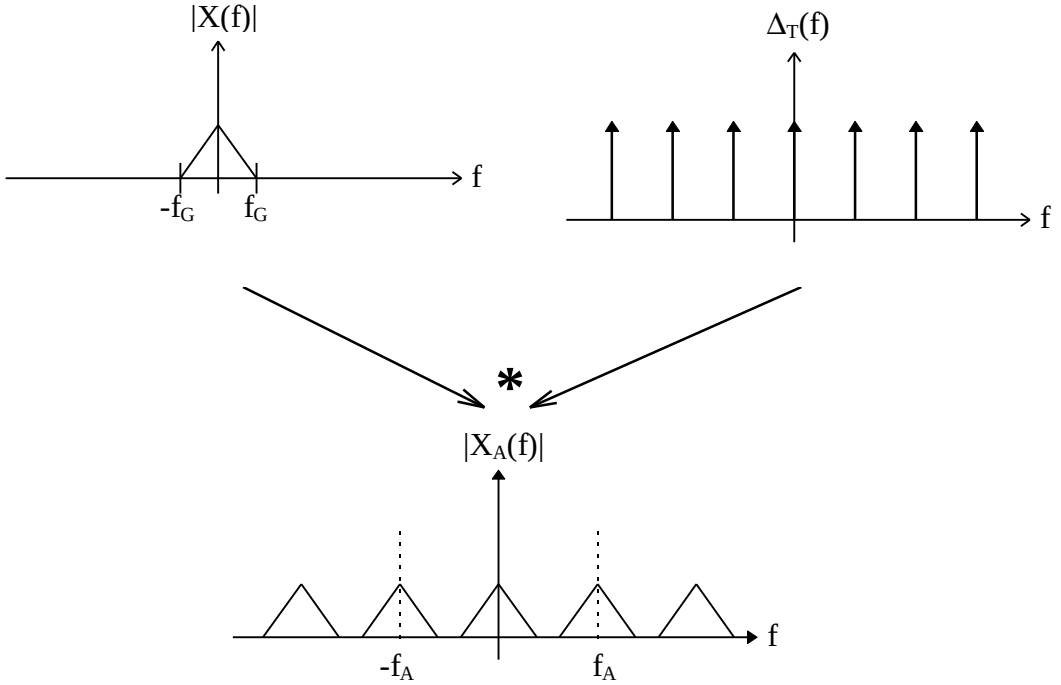


Abbildung 4

Man sieht, daß das Eingangssignal periodisch im Frequenzbereich geworden ist. Die Periode wird durch die Abtastfrequenz f_A festgelegt. Ist diese Frequenz zu klein gewählt, überlappen sich die einzelnen Spektren und das abgetaste Signal wird verfälscht.
Um das Eingangssignal eindeutig erfassen zu können, muß die also die Abtastbedinung eingehalten werden:

$$f_A = 2 \cdot f_G$$

f_G höchster zulässiger Signalanteil des Eingangssignals (gleichzeitig auch Grenzfrequenz des Eingangstiefpasses)
f_A Abtastfrequenz des A/D-Wandlers

Die Abtastfrequenz muß also mindestens zweimal so hoch sein, als die höchte im Eingangssignal vorkommende Frequenz, sonst kommt es zu Überlappungen. Aus diesem Grund ist auch ein Tiefpaß am Eingang notwendig, er sperrt für alle Signale größer f_G.

Für den Digital-Analog-Wandler gilt analog dazu, daß die Wandlungsfrequenz natürlich ebenfalls doppelt so hoch wie die maximal gewünschte Ausgangsfrequenz sein muß. Der Tiefpaß am Ausgang dient zur Glättung des Ausgangssignals (entfernen aller Frequenzen oberhalb der halben Abtastfrequenz).

Eine weiter Kenngröße neben der Abtastfrequenz ist die Auflösung der Analogndler (Anzahl der Bits). Diese Größe legt die erfassbare Dynamik des Ein- bzw. Ausgangssignals fest, also das Verhältnis des kleinsten zum größten meßbaren Wert (was dem Signal-zu-Rauschen-Verhältnis SNR entspricht). Formelmäßig ausgedrückt:

$$\text{SNR} = 20 \cdot \log\left(\frac{2^N}{2^0}\right) \approx 6 \cdot N$$

N Anzahl der Bits des A/D-Wandlers (Auflösung)

Für gebräuchliche Auflösungen ergibt sich also:

Auflösung	Dynamik
8 Bit	48dB
12 Bit	72dB
14 Bit	82dB
16 Bit	96dB

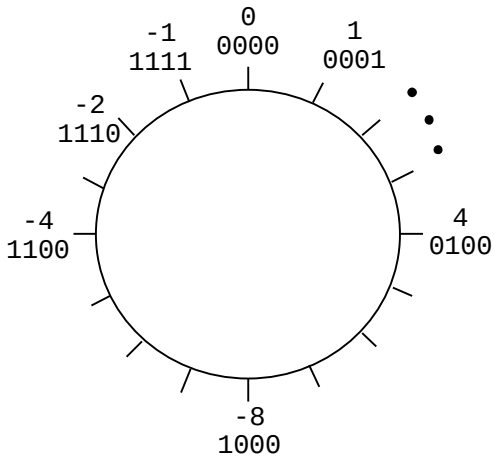
Festkomma-Arithmetik

Viele Signalprozessoren - auch der TMS320C25 - verwenden zur Verarbeitung von Zahlenwerten die sog. Festkomma-Darstellung zur Basis 2, um einen Zahlenbereich von -1.0 bis +0.99 abzudecken. Am Beispiel eines 4-Bit Zahlensystems soll die Festkomma-Darstellung erläutert werden.
Benötigt man nur positive ganze Zahlen, dann lassen sich mit 4 Bit die Zahlen 0-15 codieren, wie in der Tabelle gezeigt:

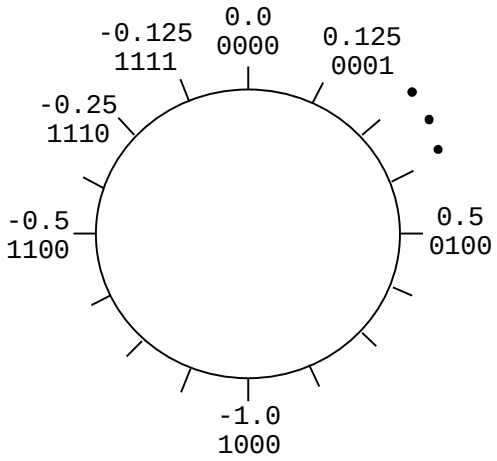
Dezimal	Binär
0	0000
1	0001
2	0010
3	0011
...	...

14	1110
15	1111

Zur Darstellung auch negativer Zahlen führt man das Zweierkomplement ein, d.h. man verwendet das höchstwertige Bit als Vorzeichenbit und zählt von 0 ausgehend rückwärts. Die Abbildung am Zahlenrad veranschaulicht das:



Um den Zahlenbereich von -1.0 bis 0.99 darstellen zu können, sieht man noch einen Skalierungsfaktor vor und erhält so die Festkomma-Darstellung. Die Abbildung zeigt die Festkomma-Darstellung für das 4-Bit System:



LTI-Operatoren

Nachdem die Analogschnittstelle besprochen wurde, sollen noch kurz auf die sog. LTI-Systeme mit ihren Grundkomponenten zur Sprache kommen, da sie in der Literatur eine wichtige Rolle spielen (und auch wichtig sind...). Ein Beispiel eines solchen Systems ist in Abbildung 5 gezeigt:

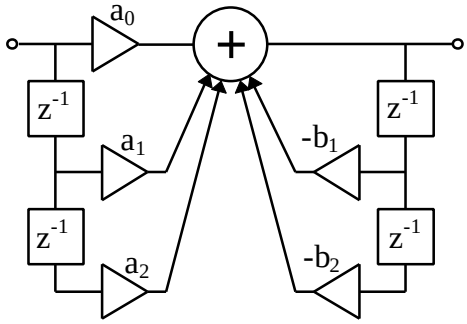


Abbildung 5

Unter LTI versteht man „linear and time invariant“, also linear und nicht zeitabhängig. Nicht zeitabhängig ist schnell erklärt, für ein Eingangssignal darf es keinen Einfluß haben ob man es jetzt oder eine halbe Stunde später auf das System gibt, das Ergebnis muß immer gleich bleiben.
Linear bedeutet:

- Bei der Überlagerung (Addition) zweier Eingangssignale, läßt sich das Ausgangssignal ebenfalls aus der Überlagerung der getrennt ermittelten Ausgangssignale ermitteln (Superpositionierbarkeit).
- Ein fester Faktor bleibt erhalten (Skalierbarkeit)

Ein solches System läßt sich durch die lineare Differenzengleichung

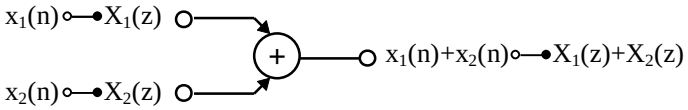
$$y(n) = \sum_{k=0}^{N} a_k x(n-k) - \sum_{k=0}^{M} b_k y(n-k)$$

beschreiben. Führt man die Z-Transformation ein, kann man auch schreiben:

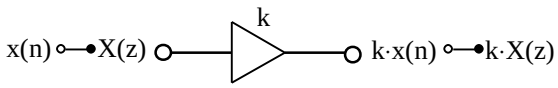
$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} a_k z^{-k}}{\sum_{k=0}^{M} b_k z^{-k}}$$

Daraus lassen sich folgende erlaubte Grundoperationen ableiten:

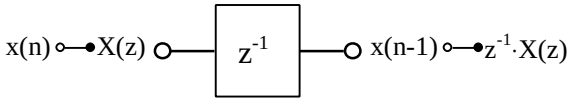
- Addition



- Multiplikation mit einer Konstante



- Verzögerung um einen Abtasttakt T_A



Diese Rechenstrukturen spielen beispielsweise bei den IIR- und FIR-Filtern eine große Rolle. Die Filterstrukturen werden dabei oft als Blockschaltbilder, bestehend aus den vorgestellten Operatoren, dargestellt.

2. Hardware

In dem Abschnitt über die Grundlagen wurde ein „Echtzeitsystem zur DSV vorgestellt“, Abbildung 6 zeigt die Realisierung eines solchen Systems im Blockschaltbild:

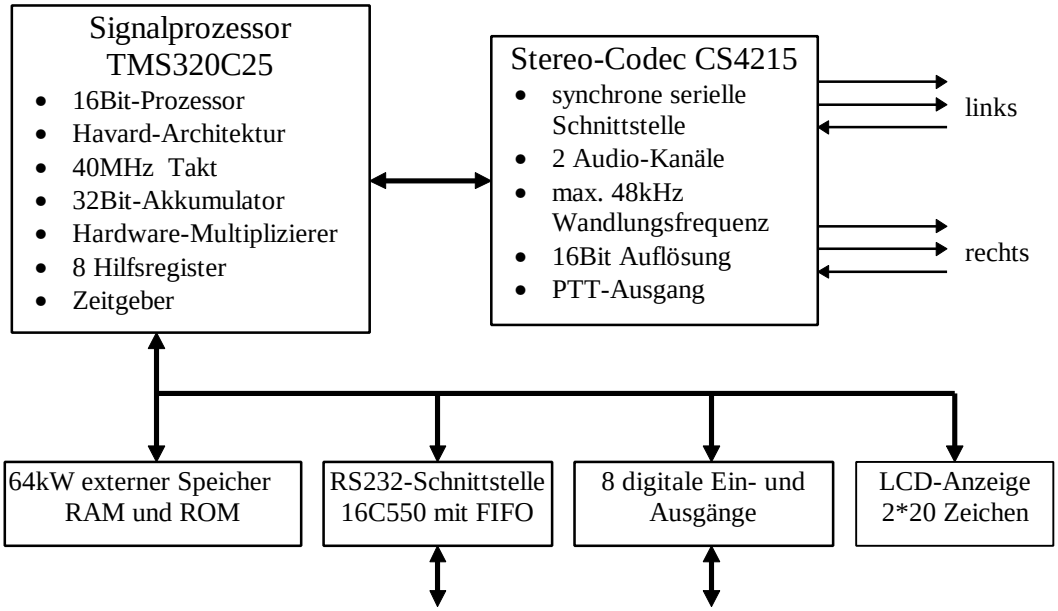


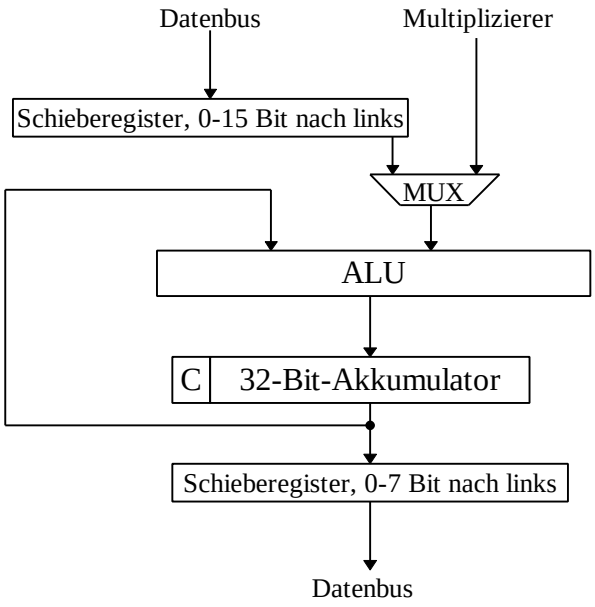
Abbildung 6

Das Herzstück der Karte ist das Rechenwerk in Form eines Signalprozessors. Man könnte hier natürlich jeden beliebigen Mikroprozessor einsetzen, jedoch sind Signalprozessoren für häufig benötigte Rechenoperationen wie Addition und Multiplikation optimiert. Es gibt eine ganze Reihe von Signalprozessoren verschiedener Hersteller, z.B. von Texas Instruments (TMS320-Serie), Motorola (56000er-Reihe) oder Analog Devices (ADSP21xx-Reihe).

Der in der DSP-Karte TMS320C25 von Texas Instruments ist zwar nicht mehr der aktuellste Vertreter aus der TMS320-Familie, aber dafür sehr preiswert und recht weit verbreitet. Die Rechenleistung ist für die relativ einfachen Anwendungen im Amateur-Bereich auch erst einmal ausreichend.
Der TMS320C25 ist ein 16-Bit Mikroprozessor und Vertreter der zweiten Generation von TI-Signalprozessoren. Er ist in der sog. Havard-Architektur aufbebaut, bei der getrennte Daten- und Programmspeicher existieren, es lassen sich somit maximal 64KW (ein Wort hat 16 Bit) adressieren. Der Prozessor erreicht bei einem Takt von 40MHz bei den meisten Befehlen eine Befehlszykluszeit von 100ns, was eine maximale Rechenleistung von 10 Millionen Befehlen in der Sekunde ergibt. Spezielle Befehle für die gleichzeitige Multiplikation und Addition oder ‘bit-reversal’-Adressierungsarten sind bei DSV-Programmen ebenfalls vorteilhaft.

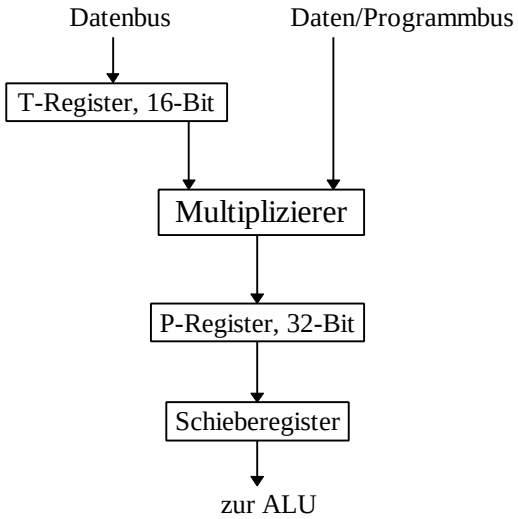
Die aus Sicht des Programmierers sind die wichtigsten Komponenten sind der 32-Bit-Akkumulator, der Hardware-Multiplizierer und die Zusatzregister AR0-AR7. Sie sollen etwas näher vorgestellt werden.

Akkumulator



Im 32-Bit Akkumulator werden die Ergebnisse des Rechenwerks (ALU, Arithmetic Logic Unit) gespeichert. Die ALU kann addieren, subtrahieren und weitere einfache Operationen mit einem oder zwei Operanden ausführen.Die Operanden können aus dem Akkumulator, einem Wert auf dem Datenbus oder vom Multiplizierer kommen.
Die höher- und niederwertigen 16 Bit des Akkumulators können über eigene Befehle angesprochen werden, so kann beispielsweise ein Wert explizit zum höherwertigen Teil addiert werden. Beim Speichern kann der Akkumulatorwert noch zusätzlich um bis zu 7 Bit nach links geschoben werden.

Hardware-Multiplizierer



Mit Hilfe des Hardware-Multiplizierers können zwei 16-Bit Werte innerhalb eines Befehlszykluses multipliziert und über das P-Register der ALU zugeführt werden.
Vor der eigentlichen Multiplikation muß das T-Register explizit mit einem der beiden Multiplikanden geladen werden.

Zusatzregister

Die Zusatzregister AR0 bis AR7 sind vorzeichenlose 16-Bit Register, die vorallem für die indirekte Adressierung benötigt werden. Außerdem lassen sich einfach Zähler und Schleifen programmieren.

Adressierungsarten

Der C25 stellt drei verschiedene Adressierungsarten zur Verfügung, auf die kurz eingegangen werden soll.

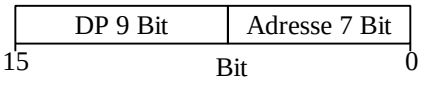
unmittelbare Adressierung

Hierbei wird der entsprechende Wert direkt im Befehl mit angegeben.
Beispiel:

```
ADDK 5          ; zum Akkumulator 5 addieren
LRLK AR0,1234h ; AR0-Register mit Hex 1234 laden
```

direkte Adressierung

Ermöglicht die direkte Adressierung einer Speicherzelle. Damit ein Befehl in nur einem 16-Bit Wort kodiert werden kann, sind nur die 7 niederwertigsten Adressbits im jeweiligen Befehl angegeben. Die höherwertigen 9 Adressbits (‘**D**ata **P**age’) werden mit einem eigenen Befehl extra gesetzt. Die komplette Adresse setztz sich dann aus den beiden Anteilen zusammen:



Beispiel:

```
LDPK 0          ; DP auf 0 setzen
ADD  5          ; zum Akkumulator Inhalt der Speicherstelle
                ; DP+5 addieren
LAR  AR0,5      ; AR0-Register mit Inhalt der
                ; Speicherstelle DP+5 laden
```

indirekte Adressierung

Bei der indirekten Adressierung wird auf die Speicherstelle zugegriffen, auf die Eines der Zusatzregister zeigt. Die Auswahl des Registers geschieht mit dem **LARP**-Befehl.
Beispiel:

```
LARP AR0        ; ARP zeigt auf AR0
ADD  *          ; Inhalt der Speicherstelle auf die AR0
                ; zeigt zum Akkumulator addieren
LAR  AR1,*      ; Inhalt der Speicherstelle auf die AR0
                ; zeigt in AR1 laden
```

Zur indirekten Adressierung gibt es noch Erweiterungen, so kann mit ‘***+**‘ nach Ausführung des Befehls, das Zusatzregister um eins erhöht werden (Postinkrement).

Analogbaustein: Stereo-Codec

Auf den Analogwandler-Baustein soll nur kurz eingegangen werden. Der hier verwendete Stereo-Codec (Coder-Decoder) CS4215 besitzt jeweils zwei Analogein- und ausgänge mit 16 Bit Auflösung und in mehreren Schritten programmierbarer Wandlungsfrequenz. Die Datenübertragung vom Signalprozessor zum Codec geschieht über eine synchrone serielle Schnittstelle.

Die Analogeingänge besitzen Anti-Aliasing-Filter, das Signalanteile oberhalb der halben Wandlungsfrequenz unterdrücken, die Analogausgänge besitzen entsprechend Glättungstiefpässe für die halbe Wandlungsfrequenz.

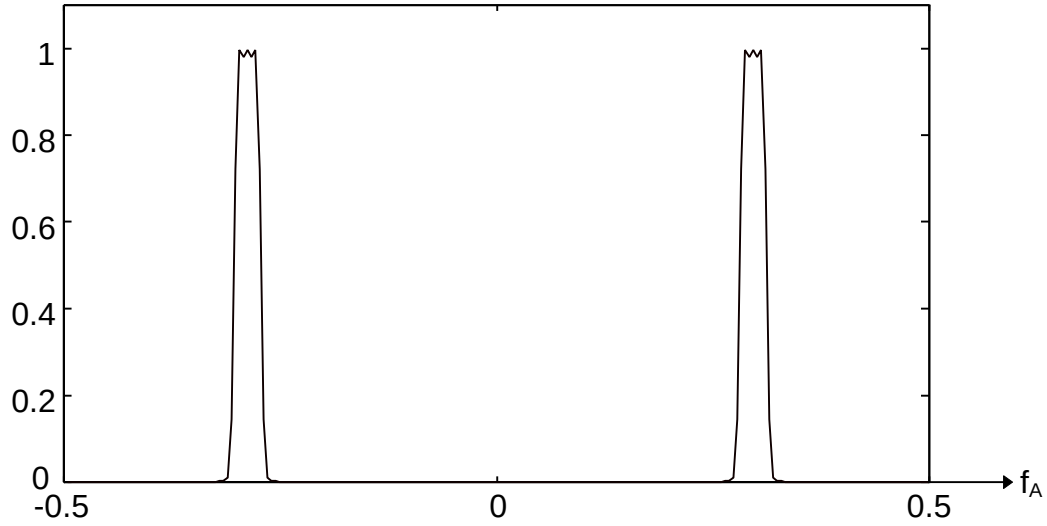
3. Beispiele

Digitale Filter: Das FIR-Filter

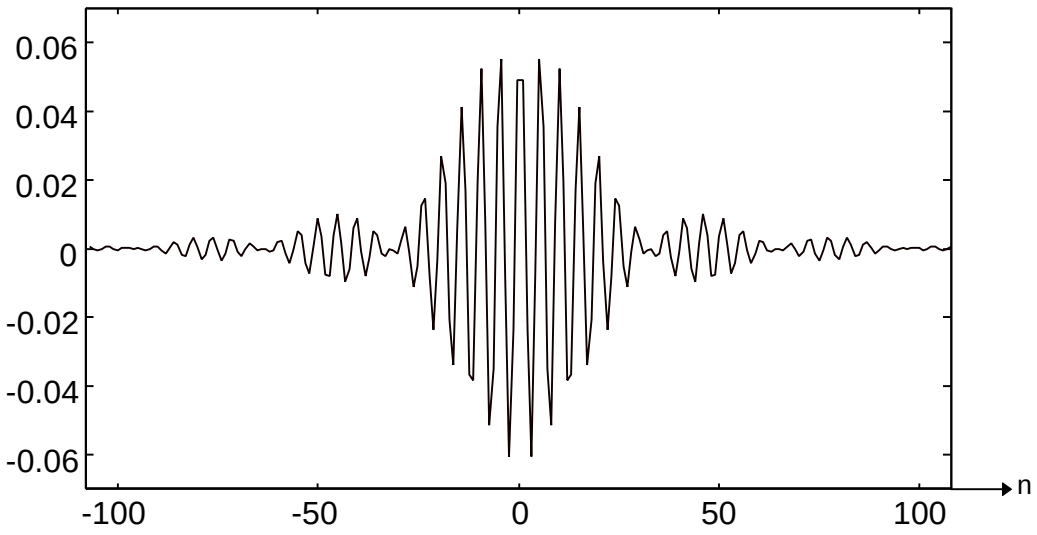
FIR-Filter, auch Transversalfilter genannt, sind die einfachste Form von Digitalfiltern. Da sie keine Rückkoplung besitzen (nichtrekursiv sind), sind sie relativ einfach zu berechnen und in ihren Eigenschaften vorhersagbar. Als Beispiel soll ein schmalbandigen CW-Filters realisiert werden.
Zur Realisierung eines FIR-Filters verwendet man meistens einen der folgenden beiden Möglichkeiten:

- Algorithmus nach Parks/McLellan (Remez exchange algorithm)
- Begrenzung der Impulsantwort des Filters mit der Fenstermethode (windowing method)

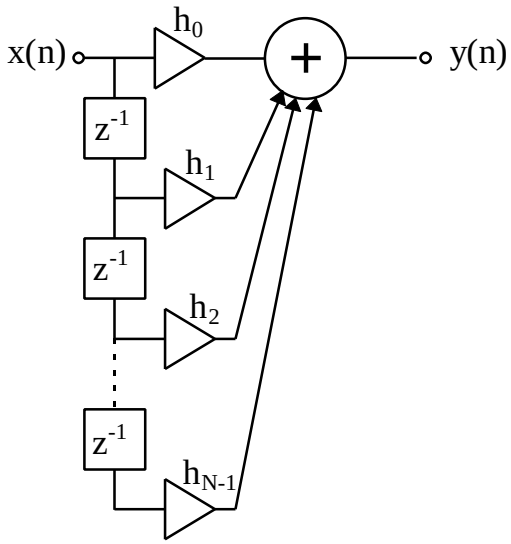
Hier wurde die Fenstermethode mit Kaiserfenster gewählt. Die Abbildung zeigt die erhaltene Übertragungsfunktion |H(f)| des gewünschten Filters (positive und negative Frequenzen sind zu sehen):



Die dazugehörige Zeitfunktion (Impulsantwort) h(n) des Filters, bei einer Filterlänge von N=216:



Die Struktur einens FIR-Filters kann mit den vorgestellten Operatoren (Addition, Multiplikation und Zeitverzögerung) dargestellt werden:

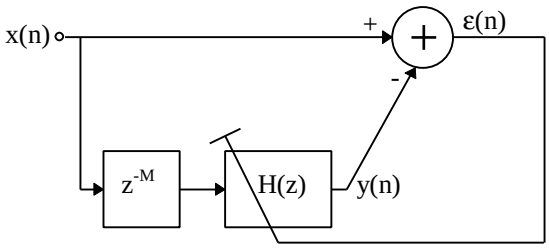


Die Realisierung in Assemblercode für den 320C25 sieht dann folgendermaßen aus:

```
LRLK AR0,X0     ; Tabelle mit Abtastwerten
ZAC             ; Akkumulator löschen
RPTK FIRLEN-1   ; 215+1 Wiederholungen
MACD HN         ; Multiplikation, Addition und kopieren
APAC
SACH YN         ; Ergebnis speichern
```

Adaptives FIR-Filter

Das im vorigen Abschnitt gezeigte Filter hat einen Nachteil: Die Filtercharakteristik läßt sich nur durch Umprogrammierung ändern. Hat man aber eine zeitlich veränderliche Störung auszufiltern, hilft das nicht weiter.
Die Abbildung zeigt das Blockschaltbild eines Adaptiven Filters zur Unterdrückung periodischer Störungen, beispielsweise von Sinusschwingungen:

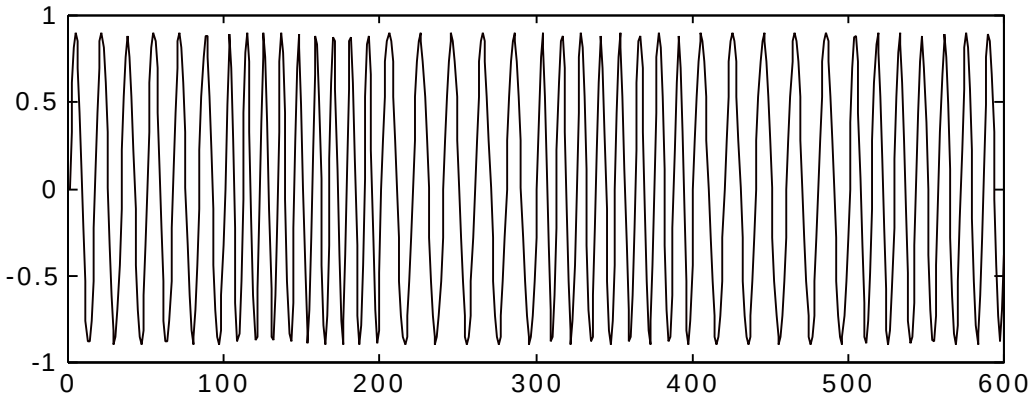


Das Eingangssignal $x(n)$ wird um M Abtasttakte verzögert dem FIR-Filter der Länge N mit der Übertragungsfunktion $H(z)$ zugeführt. Das gefilterte Signal $y(n)$ wird vom Eingangssignal $x(n)$ abgezogen, es ensteht das Fehlersignal $\varepsilon(n)$. Mit Hilfe dieses Fehlersignals werden die Filterkoeffizienten verändert:

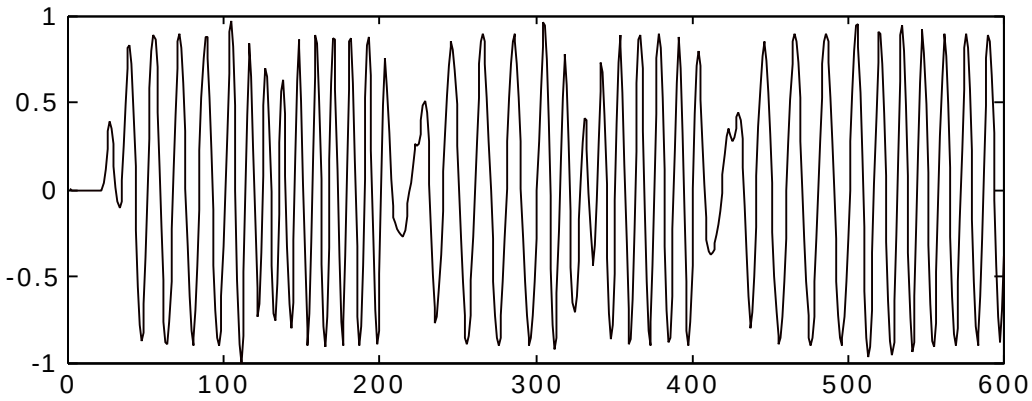
$$h_{neu}(0..N) = h_{alt}(0..N) + 2*\mu*\varepsilon(n)$$

Wobei μ eine Zeitkonstante ist, die die Dauer des Einschwingvorgangs bestimmt.
Die folgenden Abbildungen zeigen was bei einem sinusförmigen Eingangssignal mit veränderlicher Frequenz geschieht:

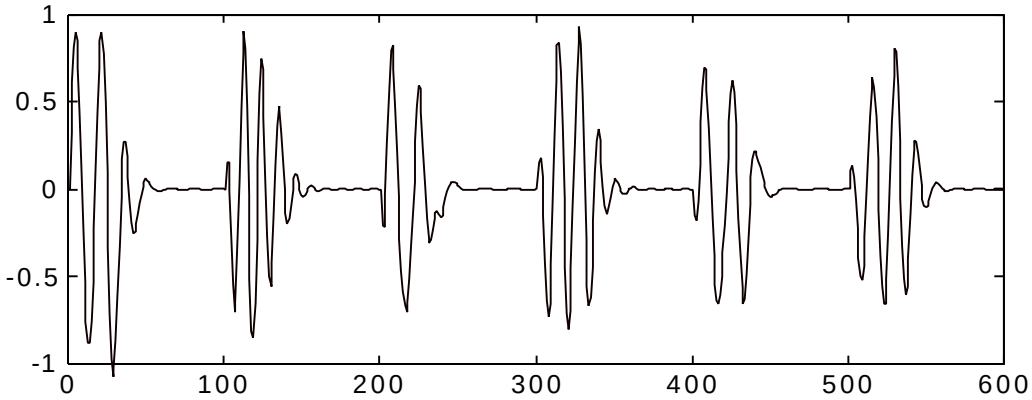
Eingangssignal x(n):



Filterausgang y(n):



Fehlersignal ε(n):



Man erkennt, daß nach einer kurzen Einschwingzeit das Fehlersignal ε(n) abgeklungen ist, die periodische Störung - und nur diese - ist ausgefiltert worden. Verwendet man das Ausgangssignal des Filters y(n) erhält man einen Bandpaß, der nur die periodische Störung durchläßt.
Der Programmteil zum Abgleich der Koeffizienten sieht so aus:

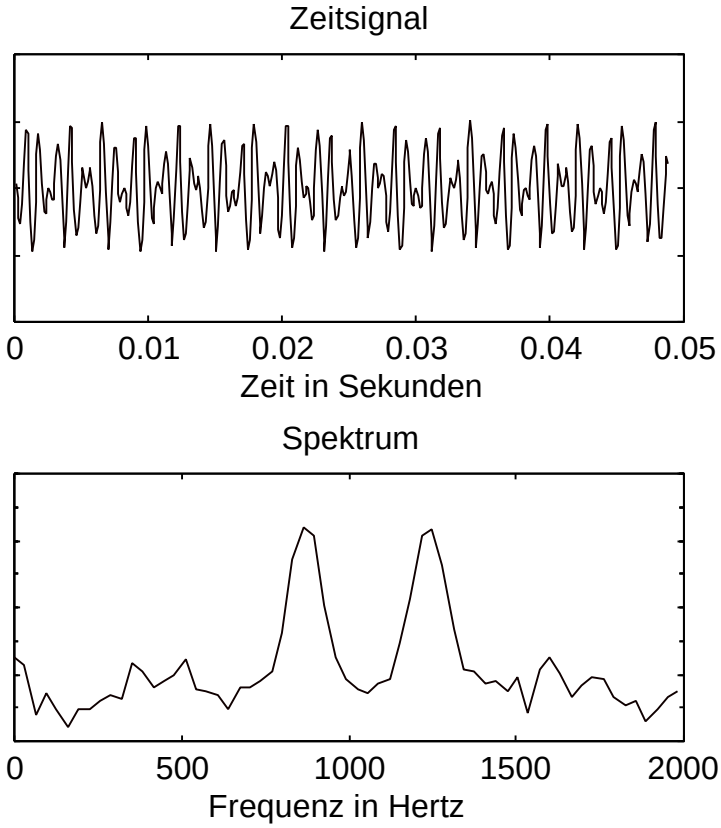
```
REPT FIRLEN
   ; Koeffizientenabgleich
   ; HN_NEU = HN_ALT + (2 * MU * EPSILON) * V_R-i
   ZALR *,AR0          ; HN_ALT[i] gerundet
   MPYA *-,AR2         ; V_R-i
   SACH *+             ; HN_NEU[i]
ENDM
```

FFT

Das vom A/D-Wandler kommende Eingangssignal ist physikalisch gesehen eine Spannungszeitfunktion. Mit Hilfe der Diskreten Fourier-Transformation (DFT) kann man diese Spannungszeitfunktion in Elementarfunktionen zerlegen und somit das Spektrum ermitteln.

Auf die mathematische Beschreibung der Diskreten Fourier-Transformation und der Ableitung des FFT-Algorithmus soll hier verzichtet werden, dies ist ausführlich in [1]-[7] zu finden.

Mit entsprechend optimiertem Code ist eine 256-Punkte FFT innerhalb von 10ms auf dem TMS320C25 möglich. Als Beispiel für eine FFT ist das Zeitsignal eines DTMF-Tones und das daraus berechnete Spektrum wiedergegeben:



Signalerzeugung

Mit einem DSP lassen sich natürlich auch Signale erzeugen. Einfache Formen wie Rechteck oder Rampe lassen sich einfach mit Hilfe von Zählern realisieren. Weitaus interessanter ist die Erzeugung einer sinusförmigen Schwingung, die beiden gebräuchlichsten Verfahren sollen kurz vorgestellt werden:

Quasistabiles IIR-Filter

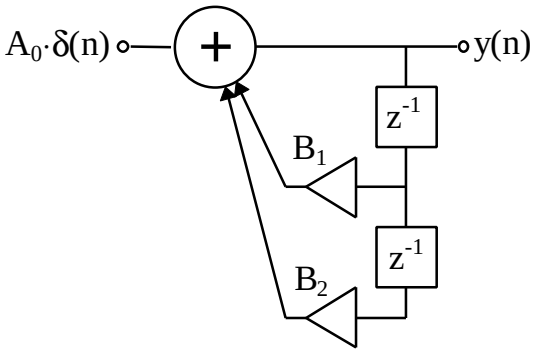
Hierbei wird ein IIR-Filter mit einem konjungiert komplexen Polpaar auf dem Einheitskreis verwendet. Das bedeutet, daß wenn das Filter mit einem Impuls angeregt wird, es am Ausgang sinusförmige Ausgangswerte erzeugt.
Die Übertragungsfunktion eines solchen Filters lautet:

$$H(z) = \frac{\mathrm{A}_0 \cdot z^{-1}}{1 + \mathrm{B}_0 \cdot z^{-1} + \mathrm{B}_1 \cdot z^{-2}}$$

Daraus ergibt sich die zu realisierende Differenzengleichung:

$$y(n) = A_0 \cdot x(n-1) - \mathrm{B}_1 \cdot y(n-1) - \mathrm{B}_2 \cdot y(n-2)$$

In der Abbildung ist die Struktur des Filters schematisch dargestellt:



Die dabei auftretenden Konstanten sind so definiert:

$A_0 = \sin(2\pi f)$

$B_1 = 2\cos(2\pi f)$

$B_2 = -1$

Wobei f die Frequenz der gewünschten Schwingung ist.
Am Anfang wird ein nur ein Startimpuls mit dem Betrag A_0 auf den Eingang des Filters gelegt, danach schwingt es selbstständig weiter.
Der Programmausschnitt zeigt wie wenig Aufwand nötig ist:

```
DMOV Y1          ; Y1 -> Y2
DMOV Y0          ; Y0 -> Y1
LT   Y2          ; erster Rückkopplung
MPY  B2
PAC
SACH P2,1
LT   Y1          ; zweite Rückkopplung
MPY  B1
PAC
SACH P1,1
ZAC              ; Addition der Komponenten
ADD  P1,2
ADD  P2,2
SACL X0
```

Vorteil dieser Art von Schwinungserzeugung ist der geringe Rechenaufwand der benötigt wird. Da der TMS320C25 aber nur 16-Bit Zahlen verarbeiten kann, lassen sich nur bestimmte Frequenzen erzeugen und die Qualität der Sinusschwinung wird durch Rundungsfehler beeinträchtigt.

Tabellenmethode

Eine Alternative zur Erzeugung eines Sinussignals ist die sog. Tabellenmethode, bei der die Sinusfunktion in einer Tabelle vorberechnet abgespeichert wird. Da nicht alle möglichen Werte der Sinusfunktion gespeichert werden können, behilft man sich mit zwei Vereinfachungen.

Zuerst wird eine Fallunterscheidung vorgenommen und die Sinusfunktion nur für einen Quadranten gespeichert:

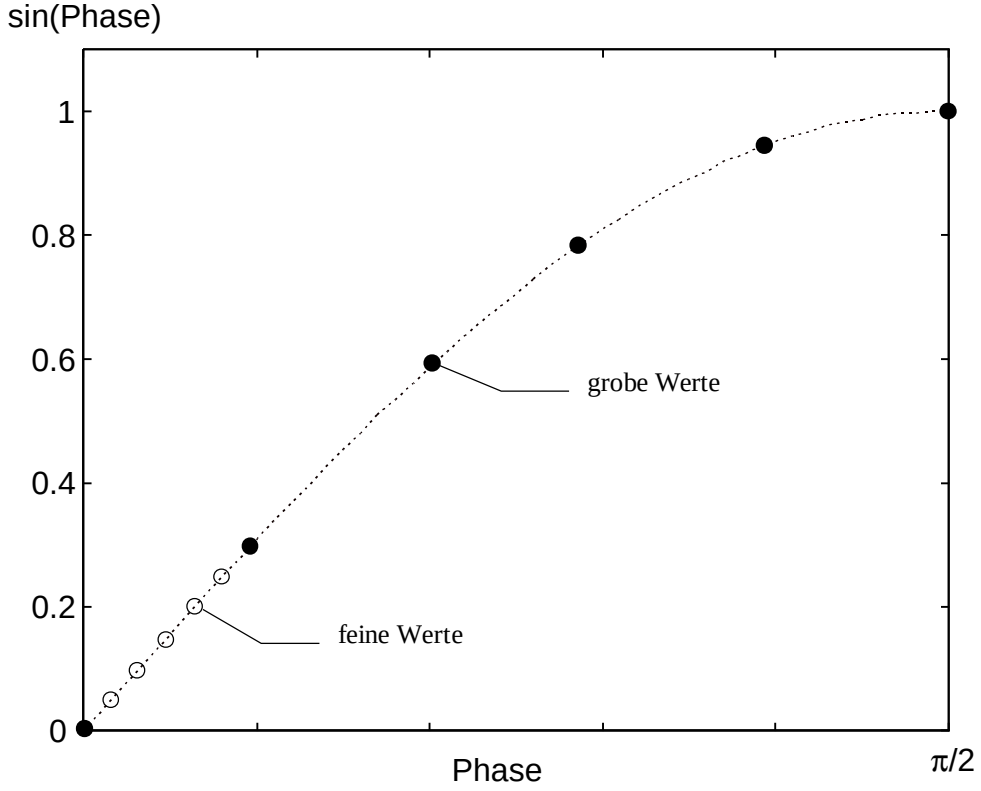
Phase	Sinusfunktion
$-\pi...-\pi/2$	$-\sin(\pi/2-\pi/2...0)$
$-\pi/2...0$	$-\sin(\pi/2...0)$
$0...\pi/2$	$\sin(0...\pi/2)$

π/2...π	sin(π/2-0...π/2)

Zusätzlich teilt man den Phasenwert in einen groben und einen feinen Teil. Für den groben Wert nimmt man die höherwertigen Bits und für den feinen Wert die niederwertigen Bits des Phasenwerts. Daraus ermittelt man getrennt den jeweiligen Sinuswert aus einer vorberechneten Tabelle für die groben und feinen Werte. Mit Hilfe der Additionstheoreme erhält man schließlich die gewünschte Funktion:

$$\sin(grob + fein) = \sin(grob) \cdot \cos(fein) + \cos(grob) \cdot \sin(fein)$$

Die Abbildung verdeutlicht das Prinzip der Aufteilung in einen groben und einen feinen Wert::



Anhang A: Übersicht Lowcost DSP-Karten

Die folgende Liste ist nach Kenntnisstand des Autors erstellt worden und soll keine „Marktübersicht“ darstellen.

DSPCOM

Hersteller:	RBW Elektronik GmbH
Prozessor:	TMS320C25 68HC11 als Protokoll-Prozessor
Analoginterface:	TLC32040 (14 Bit Auflösung, 19200Hz max. Wandlungsfrequenz) AD7569 (8 Bit Auflösung)
Software:	RTTY, Amtor, Pactor, Packet 300, 1200, 9600 mit WA8ED-Hostmode

DSP-93

Hersteller:	TAPR/AMSAT
Prozessor:	TMS320C25
Analoginterface:	TLC32040 (14 Bit Auflösung, 48kHz max. Wandlungsfrequenz)
Software:	AFSK 300, 1200 Baud, FSK 1200 und 9600 Baud, PSK 1200 Baud

DG1SCR-Karte

Hersteller:	DG1SCR, DK5DV
Prozessor:	TMS320C25
Analoginterface:	Stereo-Codec CS4215 (16 Bit Auflösung, 48kHz max. Wandlungsfrequenz)
Software:	Packet 1200, 9600, CW-Filter, Denoiser RTTY, AMTOR, PACTOR, FAX

ALEF NULL

Hersteller:	Kaj Wiik (OE6EH) und Jarkko Vuori (OH2LNS)
Prozessor:	DSP56001
Analoginterface:	Stereo-Codec CS4215 (16 Bit Auflösung, 48kHz max. Wandlungsfrequenz)
Software:	CW-Filter, Denoiser, Packet 1200, 9600

DSP1232/2232

Hersteller:	AEA Inc.
Prozessor:	DSP56001
Analoginterface:	AD7870 (12 Bit Auflösung) AD767 (12 Bit Auflösung)
Software:	Packet 300, 1200, 9600 RTTY, CW, FAX, SSTV, Sat RX 1200/4800 bps

DSK26

Hersteller:	Texas Instruments
Prozessor:	TMS320C26
Analoginterface:	TLC32040 (14 Bit Auflösung, 48kHz max. Wandlungsfrequenz)
Software:	Assembler, Debugger und Beispielprogramme von Texas Instruments RTTY, AMTOR, PACTOR, Packet 1200, 9600 CW-Filter, Denoiser

DSK50

Hersteller:	Texas Instruments
Prozessor:	TMS320C50
Analoginterface:	TLC32040 (14 Bit Auflösung, 48kHz max. Wandlungsfrequenz)
Software:	Assembler, Debugger und Beispielprogramme von Texas Instruments

DSP56002EVM

Hersteller:	Motorola
Prozessor:	DSP56002
Analoginterface:	Stereo-Codec ?
Software:	Assembler, Debbuger

PSA-Soundkarten

Hersteller:	Diverse
Prozessor:	ADSP2115 von Analog Devices
Analoginterface:	Stereo-Codec AD1848 bzw. CS4231
Software:	SDK mit Assembler gibt es kostenlos AMTOR/PACTOR/RTTY, Packet 1200Baud

Anhang B: Literatur

Allgemeine Literatur

[1] Daniel Ch. von Grüningen: Digitale Signalverarbeitung
VDI-Verlag

[2] P. Gerdsen, P. Kröger: Digitale Signalverarbeitung in der Nachrichtenübertragung
Springer-Verlag

[3] Openheimer, Schafer: Digital Signal Processing
Howard Sams

[4] Rabiner: Digital Signal Processing Algorithms in C
John Wiley & Sons

[5] S. Stearns, R. David: Signal Processing Algorithms in Fortran and C
Prentice-Hall

[6] J. Treichler, C. Johnson, M. Larimore: Theory and Design of Adaptive Filters
John Wiley & Sons

[7] S. Mitra, J.F. Kaiser: Handbook for Digital Signal Processing
John Wiley & Sons

Speziell zum TMS320C25

[7] J. Hasch, DG1SCR: Signalprozessorkarte mit dem TMS320C25
Adacom-Heft 7/94

[8] TMS320C2x User's Guide
Texas Instruments

[9] Digital Signal Processing Applications Volume I-III
Texas Instruments

[10] B. Hutchins, T. Parks: A Digital Signal Processing Laboratory using the TMS320C25
Prentice Hall

[11] R. Chassaing, D. W. Horning: Digital Signal Processing with the TMS320C25
John Wiley & Sons

Anhang C: Software

Matheprogramme

1. Mathematica
   Symbolisches Mathematikprogramm mit hervoragenden Grafikfähigkeiten. Spezielle Signalverarbeitungsbibliothek ist über FTP erhältlich.
   Studentenversion für ca. 400,- DM
2. Maple
   Ebenfalls symbolisches Mathematikprogramm, jedoch nicht ganz so komplex in der Bedienung wie Mathematica.
   Studentenversion für ca. 200,- DM
3. Matlab
   Vom Kern her rein auf numerische Berechnungen mit Arrays und Matrizen ausgerichtet, einige Erweiterungen für symbolische Berechnungen (Maple-Kern) und Signalverarbeitung. Viele kommerzielle Erweiterungen erhältlich (z.B. Simulink).
   Studentenversion für ca. 100,- DM

Filterdesign

1. Digifilt
   Programm zum Entwurf von FIR- und IIR-Filtern. Berücksichtigt Wortlängeneffekte, usw.
   Erhältlich bei der Zeitschrift ‘Elektronik’ für 99,- DM.

Internet FTP-Server

Hier eine kleine Auswahl von FTP-Servern, die Infos und Programme zum Themengebiet DSP enthalten. Die Auswahl ist rein wilkürlich.

TI.COM	- Programme für TI-Signalprozessoren
FTP.ANALOG.COM	- Informationen und Programme zu DSP’s von Analog Devices
FTP.UCSD.EDU	- Afu-bezogene DSP-Programme
FTP.UNI-KL.DE	- spiegelt teilw. den Inhalt von TI.COM und FTP.UCSD.EDU
FTP.UNI-STUTTGART.DE	- Public-Domain Assembler ‘AS’, z.B. für TMS320C25
FTP.TAPR.ORG	- Enthält Infos und Programme zum DSP-93
FTP.IX.DE	- Programme zur Zeitschrift Elrad, enthält auch DSP-Software