Hilfe Index für SoundTool 3.2

Informationen für Anwender:


Allgemeine Informationen über SoundTool

Was gibt es neues in dieser Version ?


Das Fenster von SoundTool


Menüpunkte von SoundTool


Kommandozeilen-Parameter von SoundTool


Ziehen und Ablegen (Drag and Drop) mit SoundTool


Ein/Ausgabe-Bibliotheken installieren


Verwendung von dynamischem Datenaustausch

Informationen für Programmierer:


Einträge in SNDTOOL.INI
Dateiformate


Verwendung der Zwischenablage
Beispiele zum Datenaustausch über die Zwischenablage


Eine DLL zur Wiedergabe von Signalen hinzufügen


Eine DLL zur Aufnahme von Signalen hinzufügen

Allgemeine Informationen über SoundTool

SoundTool

für

Microsoft Windows

Version 3

SoundTool ist ein einfaches Werkzeug um digitalisierte 8-bit Tonsignale zu bearbeiten.

Die vorliegende Version stützt sich dabei auf die Dynamische Link Bibliothek DSOUND.DLL (© 1990-1991 Aaron Wallace) um Teile oder das gesamte Tonsignal abzuspielen.

SoundTool kann Teile eines Signals ausschneiden, in die Zwischenablage kopiern, oder daraus einfügen und verschiedene andere Manipulationen des Signals vornehmen. Im Signale aufzuzeichnen benötigen Sie entweder eine A/D Wandler-Karte oder die Sound Blaster Karte von Creative Labs.
Außerdem ist eine DDA-Schnittstelle enthalten mit der es möglich ist, Konzepte wie voice-mail oder in Dokumente eingebettete Geräusche zu realisieren.

Um SoundTool sinnvoll verwenden zu können, ist eine  hilfreich, Sie können aber alle Funktionen auch ohne dieses Tierchen verwenden.

SoundTool ist Shareware.

Sie sollten DM 30.- an folgende Adresse senden, wenn Sie sich ein ruhiges Gewissen sichern und ein ewiges Schmoren in der Hölle ersparen möchten. Außerdem erhalten registrierte Benutzer **zusätzliche Bibliotheken**, die Ein- und Ausgabe mit Hilfe der **Sound Blaster** Karte bzw. über die Multimedia-Schnittstelle von Windows 3.1 ermöglichen.

Martin Hepperle
Robert-Leicht-Straße 175
W-7000 Stuttgart 80
- Adresse ist gültig bis Dezember 1992 -
Postgiroamt Stuttgart 1987 45-701 (BLZ 600 100 70)

Thank you for reading and have a nice day.

Was gibt es Neues in dieser Version von SoundTool ?

Diese Version von SoundTool sieht aus wie die Vorgängerversionen, aber hinter den Kulissen hat sich einiges verändert.:

Neues DLL Format: (2.5)
Die wichtigste Änderung betrifft die Ein- und Ausgabeschnittstelle. Aufnahme und Wiedergabe sind nun komplett in dynamische Bibliotheken ausgelagert, so daß es möglich ist SoundTool mit den meisten Audio-Karten zu verwenden sofern der erforderliche Treiber geschrieben wird.
Manche Hersteller wie z.B. Creative Labs (Entwickler der verbreiteten Sound Blaster Karte) liefern spezielle Treiber für Windows wodurch es relativ einfach ist, die zusätzliche Schnittstellenbibliothek für SoundTool zu erstellen.

SoundBlaster Unterstützung: (2.3)
Registrierte Anwender von SoundTool erhalten die notwendigen Bibliotheken für die Sound Blaster Karte; die hier vorliegende Shareware-Version enthält die Ausgabebibliothek für den internen PC Lautsprecher und die Aufnahmebibliothek für eine D/A-Karte die in Deutschland von der Firma Bitzer in 7060 Schorndorf vertrieben wird.

Neuer Werkzeugbalken: (2.4)
Um einen rascheren Zugriff auf häufig benötigte Kommandos zu bieten, wurde eine wahlweise einblendbare Werkzeugleiste geschaffen.

DDA-Schnittstelle: (2.3)
Die Verwendung von DDA erlaubt die Steuerung durch fremde Programme wie WinWord oder Excel.

Kommandozeilen Optionen: (2.5)
Mit der Option können Sie Audiodateien abspielen, ohne daß SoundTool sichtbar wird; die Option x erlaubt den Start fremder Programme wie WinWord oder Excel mit akustischer Untermalung (sofern Sie die SoundBlaster-Bibliotheken installiert haben).

Neue Filter Option: (2.4)
Hiermit ist es möglich, ein das Audiosignal durch ein ausgewähltes Frequenzband zu filtern. Sie können Hintergrundrauschen eliminieren oder menschliche Stimmen verfremden.

Audio Dateiformat bereinigt: (2.5)
Nachdem mir nun eine SPARCstation zur Verfügung steht, konnten die Routinen zum Lesen des Audio Formats (wie es von SUN und NeXT verwendet wird) neu organisiert werden. SoundTool kann jetzt lineare 16-bit saples und μLaw codierte 8-bit samples lesen.

Dateien löschen: (2.5)
SoundTool bietet nun die Möglichkeit einzelne Dateien zu löschen ohne den Dateimanager zu verwenden.

Wave Dateiformat: (2.5)
SoundTool kann jetzt lineare, PCM-codierte 8-bit samples im Microsoft Multimedia Format (RIFF/WAVE) lesen und schreiben.

Frequenzen werden aus SNDTOOL.INI gelesen: (2.5)
Die im Kombinationsfeld angebotene Auswahl an Frequenzen wird nun aus der Datei SNDTOOL.INI eingelesen. Damit kann SoundTool jetzt nahezu jede beliebige Frequenz bei Aufnahme und Wiedergabe verwenden, --vorausgesetzt Ihre Hardware ist in der Lage die Frequenz zu verarbeiten. Die Frequenzwerte können unter Einstellungen-Frequenzen... nach

Wunsch eingestellt werden.

Format der DLLs erweitert: (2.5)
Um das Abspielen bzw. Aufnehmen bei Verwendung von hintergrundfähigen Treibern abzubrechen wurden neue Funktionen in die DLLs integriert.

Speichern von Samples korrigiert: (2.6)
SoundTool speichert jetzt wirklich nur den ausgewählten Bereich eines Spektrums.

Treiber für Windows 3.1 verfügbar: (2.6)
SoundTool kann jetzt auch mit den Mutimedia Erweiterungen von Windows 3.1 arbeiten.

Neue Filter Optionen: (2.6)
Zwei mittelwertbildende Filter erlauben es, hochfrequentes Hintergrundrauschen zu eliminieren.

Ziehen von Dateien unterstützt: (2.7)
Anstelle der Auswahl über die Menüoption Datei-öffnen können Sie Dateien direkt aus dem Dateimanager in das Fenster oder auf das Sinnbild von SoundTool schieben um sie zu laden.

Zusatzinformationen in WAVE Dateien: (2.7)
SoundTool speichert nun auch die Beschreibung der Aufnahme in jeder WAVE-Datei und liest sie natürlich auch wieder ein.

Drag und Drop Unterstützung: (2.7)
Wenn Sie Windows 3.1 verwenden, unterstützt SoundTool das Laden von Dateien mittels Drag und Drop vom Dateimanger aus.

WAVE Dateien mit Zusatzinformation: (2.7)
Beim Abspeichern im WAVE Format wird nun auch die Beschreibung des Signals eingefügt; beim Einlesen wird einen eventuell vorhandene Beschreibung berücksichtigt.

Ausgabe im VOC Dateiformat ergänzt: (2.8)
SoundTool kann jetzt auch Dateien in Electronic Arts VOC Dateiformat ausgeben.

Rohe Dateien mit und ohne Vorzeichen: (2.8)
8-Bit daten können jetzt auch vorzeichenbehaftet sein (Bereich -127...128).

Tongenerator: (2.9)
Drei Tongeneratoren ermöglichen jetzt die Erzeugung von Sinus-, Rechteck- oder Sägezahn-Signalen mit beliebiger Länge und Frequenz.

Funktionsgenerator: (3.0)
Ein programmierbarer Funktionsgenerator kann zur Erzeugung beliebigen Signalen verwendet werden.

TouchTone: (3.1)
Diese Funktion kann Töne des (in den USA gebräuchlichen) Tonfrequenz-Wahlverfahrens erzeugen.

Drag and Play: (3.1)
Audiodateien können nun vom Dateimanager auf das *Sinnbild* von SoundTool gezogen und dort abgelegt werden. SoundTool lädt die Dateien, spielt sie sofort ab und löscht sie wieder aus seiner Dateiliste. Wenn SoundTool als *Vollbild* dargestellt wird, werden die Dateien geladen, aber nicht abgespielt.

LED-Ziffern Anzeige: (3.2)
Die Anzeige von Start, Ende und Länge der aktuellen Auswahl erfolgt jetzt durch Siebensegment-LED-Anzeigen

Fehlerbeseitigung: (3.2)
Die Anzeige von Start, Ende und Länge der aktuellen Auswahl erfolgt jetzt durch Siebensegment-LED-Anzeigen
Ein lange nicht aufzufindender Fehler, der dafür sorgte, daß bei oftmaligem Speichern keine Dateihandles für SoundTool mehr verfügbar waren, wurde beseitigt.

Graphische Verfeinerung der Oberfläche: (3.2)
SoundTool verwendet nun für alle Steuerelemente eine dreidimensionale Darstellung, wie sie in zukünftigen Windos-Versionen Standard sein wird.

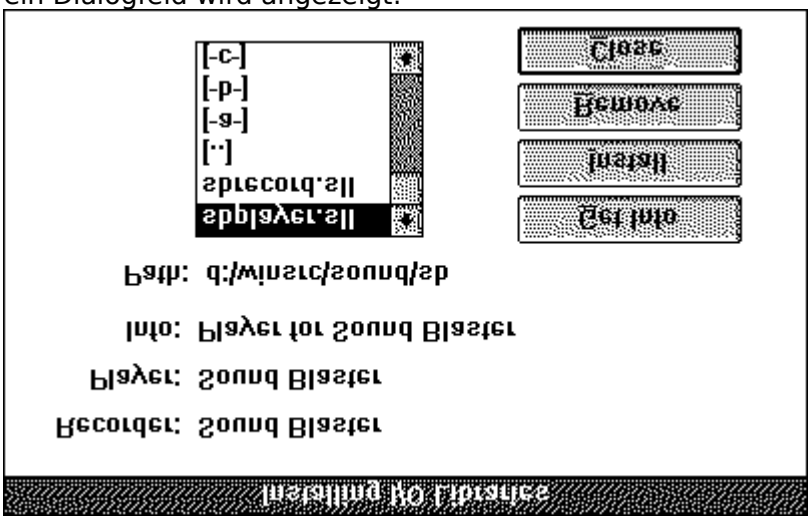
Ein- und Ausgabe Bibliotheken für SoundTool installieren

Wenn Sie SoundTool zum ersten Mal benützen, müssen Sie Bibliotheken für die Ein- und Ausgabe von Audiosignalen wählen und installieren.

Befolgen Sie hierzu die nächsten Schritte:

-- Wählen SIe den Menübefehl **Datei Installieren...**

-- ein Dialogfeld wird angezeigt:



-- wählen Sie eine Bibliothek aus und betätigen Sie **[Info]**. Rechts neben Info: wird eine kurze Beschreibung angezeigt. Um diese Bibliothek zu installieren wählen Sie **[Installieren]**. Sie sollten für SoundTool nur Bibliotheken installieren für die Sie auch die entsprechende Hardware besitzen.

-- Nachdem Sie die Bibliotheken installiert haben, betätigen Sie die Schltfläche **[Schließen]** .

-- Sie können eine installierte Bibliothek wieder entfernen, wenn Sie die Bibliothek nochmals auswählen und dann **[Löschen]** betätigen (Hinweis: es ist nicht unbedingt notwending, daß Sie die *selbe* Bibliothek auswählen. Nur der *Typ* ist wichtig damit SoundTool entscheiden kann, welche der beiden Bibliotheken Sie entfernen möchten(Wiedergabe oder Aufnahme)).

Ziehen und Ablegen (Drag und Drop) mit SoundTool

Sie können Ziehen und Ablegen (drag and drop) auf zwei Arten in Verbindung mit SoundTool verwenden:

a) Ziehen und Laden:
Ziehen Sie eine Audiodatei vom Windows Dateimanager auf das Fenster von SoundTool und legen sie die Datei dort ab, während SoundTool als **Vollbild** angezeigt wird. SoundTool lädt die Datei, wie wenn Sie den Menübefehl Datei öffnen verwendet hätten. Diese Methode ist auch mit mehreren Datein verwendbar

b) Ziehen sie eine Audiodatei über das **Sinnbild** von SoundTool und legen sie sie dort ab. SoundTool lädt die Datei, spielt sie ab und löscht sie wieder aus seiner Liste.

Kommandozeilen-Parameter von SoundTool

SoundTool kann mit oder ohne Kommandozeilen Optionen verwendet werden:

a) Installieren Sie SoundTool in einer Gruppe des Windows **Program Managers**. Dann können Sie SoundTool starten indem Sie auf das Sinnbild doppelklicken. Diese Methode startet SoundTool ohne ein Spektrum zu laden.

b) Öffnen Sie den Windows **Datei Manager** so daß SNDTOOL.EXE sichtbar ist und doppelklicken Sie auf die entsprechende Zeile. Auch diese Methode startet SoundTool ohne ein Spektrum zu laden.
Sie können eine Datei mit der Endung SOU, SND, SNP, TXT, AU, IFF, WAV oder NXT auf die Zeile SNDTOOL.EXE ziehen und dort fallenlassen. SoundTool wird gestartet und lädt diese Datei. Diese Methode funktioniert nur, wenn sich SNDTOOL.EXE und die Tonsignal-Datei im selben Verzeichnis befinden.

c) Starten Sie SoundTool. Öffnen Sie den Windows **Datei Manager** so daß eine Datei mit der Endung SOU, SND, SNP, TXT, AU, IFF, WAV oder NXT sichtbar ist.
Markieren und ziehen Sie eine oder mehrere dieser Dateien auf das Fenster oder das Sinnbild von SoundTool. Lassen Sie dort den Mausknopf los (Drag and Drop). Alle verschobenene Dateien werden nacheinander von SoundTool geladen.
Diese Funktion steht nur unter Windows 3.1 oder höher zur Verfügung.

Option -
Sie können auch auf eine Zeile, welche eine Datei mit der Endung SOU, SND, SNP, TXT, AU, IFF, WAV oder NXT enthält, doppelklicken. Diese Methode startet SoundTool so daß sein Fenster unsichtbar bleibt, falls Sie eine Verknüpfung zwischen SNDTOOL.EXE - und z.B. *.SND Dateien vorgenommen haben (Beachten Sie das Lerzeichen, gefolgt von einem Minus Zeichen hinter SNDTOOL.EXE; Sie können auch SNDTOOL.EXE SNDFILE.SND von Hand ausführen). Die Tondatei wird geladen, einmal abgespielt und SoundTool wird wieder beendet (Dies arbeitet genauso wie Aaron Wallaces SOUNDER.EXE Programm, unterscheidet aber zwischen den verschiedenen Dateiformaten). Wenn Sie ein Format gewählt haben, das eine Übersetzung erfordert (wie z.B. eine SUN-Audio-Datei), dann entsteht eine deutliche Wartezeit, bis das Geräusch abgespielt wird; nur Geduld !

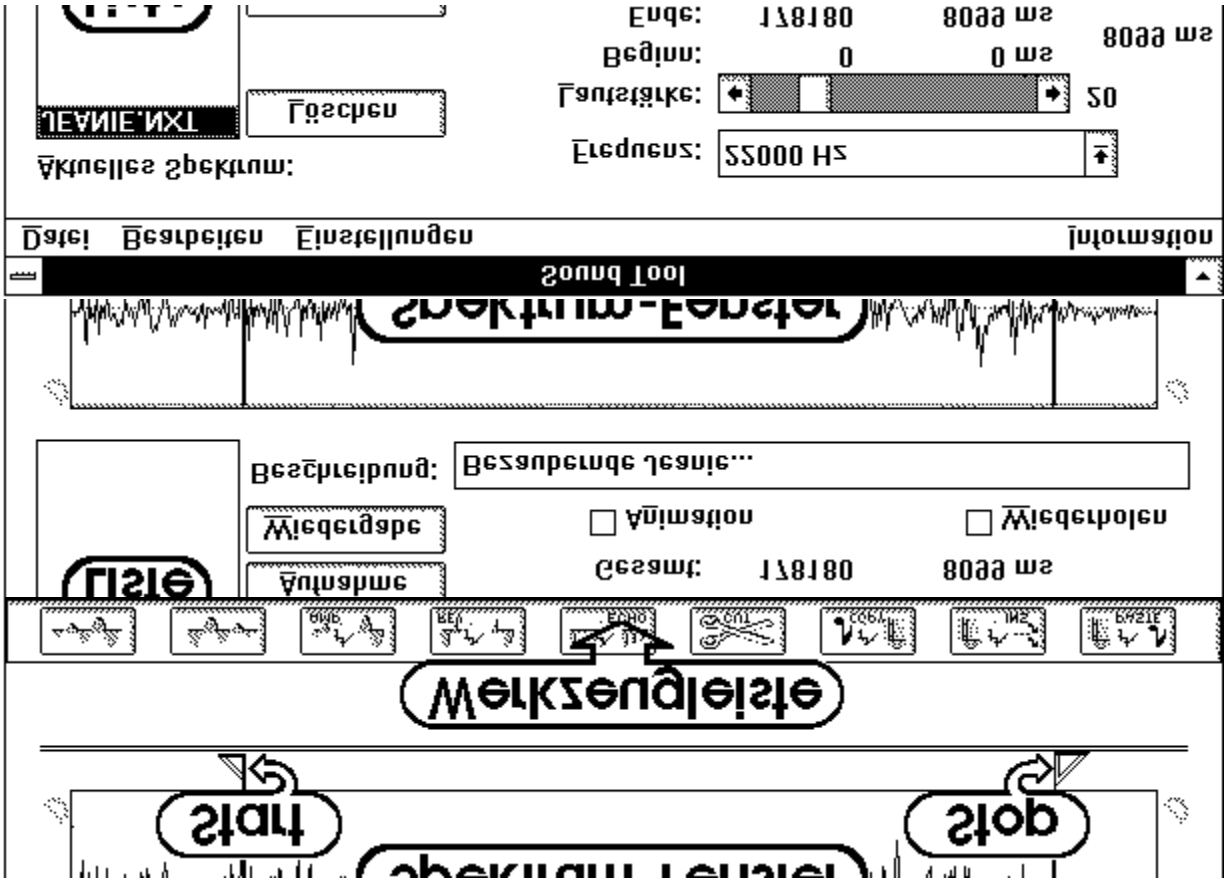
Beispiel:
```
SNDTOOL.EXE  -  HIDDEN.SND
```

Option -x
Sie können direkt nach Angabe der Option -x zwischen doppelten Anführungszeichen den Namen einer anderen Applikation angeben. Wenn Sie in der Kommandozeile außerdem eine Audiodatei mit einer der Standardendungen angeben, spielt SoundTool die Audiodatei ab und startet anschließend das nach -x angegebene Programm; wenn Sie eine Ausgabe-Bibliothek verwenden, die im Hintergrund wiedergeben kann, wird die Audiodatei gespielt *während* die Applikation gestartet wird. Damit die Datei mit der korrekten Frequenz wiedergegeben wird, sollte sie in einem der Formate vorliegen, welches die Frequenz enthält (also nicht als rohe .SOU Datei).

Beispiel:
```
SNDTOOL.EXE  -x"D:\EXCEL\EXCEL.EXE"  FANFARE.SND
```

Das Fenster von SoundTool

Nach dem Start von SoundTool wird folgendes Fenster angezeigt:



Elemente im SoundTool Fenster:

Das Fenster enthält die folgenden Kontrollelemente und Anzeigen:

Aktuelles Spektrum ein Listenfeld in dem die geladenen Spektren angezeigt werden. Ein Spektrum kann als aktuell ausgewählt werden. Alle späteren Manipulationen beziehen sich dann auf dieses Signal. Wenn die Daten in einem der beiden SoundTool Formate *.SND oder *.SNP vorliegen, wird die Beschreibung der Daten angezeigt, ansonsten der Dateiname.

Löschen löscht das aktuelle Spektrum aus dem Speicher. Die Daten sind verloren, wenn sie nicht vorher abgespeichert wurden !

Aufnahme Nach Auswahl dieser Schalfläche können Sie ein Signal aufnehmen (nur verfügbar in der registrierten Version von SoundTool). Durch Druck auf die [Escape] / [Eingabe Löschen] - Taste lässt sich die Aufnahme abbrechen (nicht bei Verwendung von DSOUND.DLL). Je nach Aufnahmegerät lassen sich verschiedene Aufnahmeparameter

unter dem Menüpunkt Einstellungen-Aufnahme verändern. Nach der Aufnahme entfernt SoundTool eventuell vorhandene Bereiche mit Stille am Anfang und am Ende der Aufnahme und fügt die Aufnahme an das Listenfeld Aktuelles Spektrum an.

Abspielen der ausgewählte (s.u.) Teil des Aktuellen Spektrums wird abgespielt. Durch Druck auf die [Escape] / [Eingabe Löschen] - Taste lässt sich die Wiedergabe abbrechen (nicht bei Verwendung von DSOUND.DLL). Je nach Wiedergabegerät lassen sich verschiedene Parameter unter dem Menüpunkt Einstellungen-Wiedergabe verändern.

Animation Wenn dieses Optionsfeld markiert ist, wird während der Wiedergabe eines Signals der bereits gespielte Teil farbig hervorgehoben; Sie können durch Halten der Taste [Eing Lösch] ([Esc] auf US-Tastaturen) abbrechen. Dann wird der Ende Zeiger an die Position verschoben, an der der Tastendruck erfolgte.
Bei Verwendung des internen Lautsprechers mit der Bibliothek SPEAKER.SLL wird das Signal in kleinen Häppchen gespielt. Daher ist in diesem Fall die Tonqualität sehr schlecht; --wie schlecht, hängt von der Länge des Spektrums ab.
Bei Verwendung der Sound Blaster Karte wird die Tonqualität nicht beeinflusst, da das Signal im Hintergrund abgespielt wird. Dafür können Tonausgabe und graphische Darstellung eventuell nicht exakt übereinstimmen.

Wiederholen Wenn dieses Optionsfeld markiert ist, spielt SoundTool das Spektrum nach Betätigung des Abspielen Feldes ständig wieder von vorne ab. Der Abbruch kann auch hier durch Druck auf die Taste [Eing Lösch] (bzw. [Esc] auf US-Tastaturen) erfolgen. Da erst am Ende des Abspielens geprüft wird, ob eine Taste gedrückt wurde, muß die Taste *gehalten* werden, bis SoundTool darauf reagiert.

Frequenz Hier können Sie die Frequenz auswählen, mit der das aktuelle Spektrum gespielt werden soll. Je nach Ausgabeberät können einzelne Frequenzen keine Wirkung zeigen, beispielsweise kann die normale SoundBlaster Karte maximal mit 22 kHz wiedergeben, bei Auswahl von 44kHz erfolgt die Ausgabe mit der höchstmöglichen Frequenz, also mit 22 kHz. Sie können eigene Frequenzen nach Auswahl des Menüpunkts Einstellungen-Frequenzen hinzufügen und vorhandene Werte löschen.

Lautstärke Durch Bewegung des Schiebereglers kann die Lautstärke nach Wunsch verändert werden. Die normale SoundBlaster Karte zeigt keine Wirkung auf diese Einstellung, hier muß das Spektrum (durch enstsprechende Verstärkung) verändert werden um eine Lautstärke-Änderung zu erzielen.

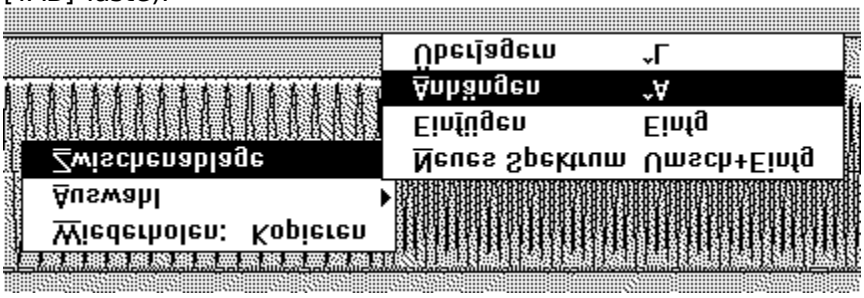
Beginn Dies ist der linke Schieber im Spektrum-Fenster. Er wird verwendet um den Beginn einer Auswahl zu markieren. Seine Position kann durch Druck auf den Mausknopf innerhalb des nach links zeigenden Dreiecks unterhalb des Spektrum-Fensters geändert werden. Ziehen Sie den Schieber zu einer neuen Position und lassen Sie den Mausknopf los. Sie können die Stellung auch mit den Cursortasten (← und →) verändern, wenn Sie den Eingabefokus mit der Maus oder mit der Tabulator-Taste auf den Schieber gesetzt haben (Dreieck blinkt); wird

gleichzeitig die STRG (CTRL) Taste gedrückt, geht es etwas schneller voran. Dann können auch die beiden Tasten Pos1 (Home) und End verwendet werden.
Neben der Textzeile **Start** unterhalb des Lautstärkereglers wird die Nummer des ersten Bytes und die Startzeit relativ zum Anfang des Spektrums angezeigt.

Ende Mit diesem Schieber können Sie das Ende der Auswahl einstellen. Auch er kann, auf die selbe Art wie oben beschrieben, mit der Maus bewegt werden.
Neben der Beschriftung **Stop** unterhalb von **Start** wird das letzte ausgewählte Byte angezeigt und auch die Zeitspanne zum Anfang des Spektrums ausgegeben.
Zwischen den beiden Zeilen **Start** und **Stop** wird rechts die Differenz zwischen Beginn und Ende ausgegeben. Die Gesamtlänge des Spektrums ist unterhalb dieser Zeilen zu finden.

Beschreibung Dieser Text beschreibt das Signal mit bis zu 95 Buchstaben. Er wird zusammen mit den Daten abgespeichert, wenn eines der SoundTool Dateiformate *.SND oder *.SNP verwendet wird.

Spektrum Fenster Dieses Fenster zeigt eine graphische Darstellung des SIgnals. Die beiden Schieber unterhalb des Fensters (**Start** und **Stop**) können bewegt werden um eine Auswahl zu markieren. Sie haben einen raschen Zugriff auf die Befehle des Menüs Bearbeiten wenn Sie den rechten Mausknopf klicken während sich der Zeiger über dem Spektrum-Fenster befindet oder durch Eingabe von [Eingabe] während das Spektrum-Fenster aktiv ist (setzen Sie die Einfügemarke durch Anklicken mit der Maus oder durch wiederholte Betätigung der [TAB]-Taste).



Werkzeugleiste Hier finden Sie Schaltflächen für häufig verwendete Funktionen::

 Auswahl aufblenden

 Auswahl abblenden

 Auswahl verstärken

 Auswahl rückwärts anordnen

 Auswahl mit Echo versehen

 Auswahl in die Zwischenablage ausschneiden

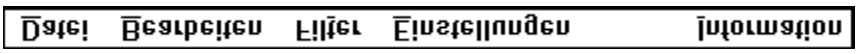
 Auswahl in die Zwischenablage kopieren

 Zwischenablage einfügen (an der Start Position)

Zwischenablage einfügen (als neues Spektrum)

Menupunkte:

SoundTool zeigt einen Menübalken an der Oberkante seines Fensters:



Datei

⇒ **Datei Öffnen...** Lädt die ausgewählte Datei in den Speicher und fügt sie an das Ende der Liste an.
Sie können einen der folgenden Datentypen laden:

- ⇒ **Roh 8-Bit** *.SOU Mit 8-Bit gesampelte Rohdaten ohne Kopf.
- ⇒ **Sound** *.SND Mit 8-Bit gesampelte Daten mit Kopf. Diese Dateien können in einem der folgenden beiden Formate sein:
  1. Dateien die von Aaron Wallace´s SOUNDER geschrieben wurden.
  2. Dateien, die von SoundTool geschrieben wurden.
  SoundTool erkennt automatisch um welches Format es sich handelt.
- ⇒ **Sound gepackt** *.SNP wie oben, jedoch etwas kompaktere Dateien durch Differenz-Kompression.
- ⇒ **NeXT** *.NXT
- ⇒ **SUN Audio** *.AU Sound Formate die von NeXT Computern und SUN SparcStations benutzt werden. Diese Dateien können lineare 16-Bit Samples enthalten, die in SoundTool auf 8-Bit reduziert werden oder die Dateien enthalten 8-Bit Samples im μLaw-Format.
- ⇒ **ANSI** *.TXT Textformat. Die Datei muß Ganzzahlen im Bereich 0...255 getrennt durch Leerzeichen, Tabulatorzeichen oder Zeilenende enthalten. Sie können solche Dateien zum Beispiel mit Microsoft Excel erzeugen, indem Sie eine Funktion wie ´255*sin(x)+1)´ verwenden.
  Beispiel für eine gültige Datei:

```
127 200 220
230
255
        220
        220
```

- ⇒ **Vocal** *.VOC Format welches von Soundblaster Software verwendet wird. Diese Dateien können 8-Bit Samples enthalten, die von SoundTool gelesen werden. Andere Daten werden ignoriert.
- ⇒ **IFF** *.IFF **I**nterchange **F**ile **F**ormal: ein Format das von Electronic Arts definiert wurde, wird hauptsächlich auf dem Commodore Amiga verwendet (wird auch mit AIFF für Amiga-IFF bezeichnet).
  SoundTool kann nur nicht komprimierte 8-Bit Samples lesen.
- ⇒ **WAVE** *.WAV **Microsoft WAVE Format**. SoundTool kann

augenblicklich nur 8-Bit PCM Mono Samples verarbeiten.

⇒ **Datei Speichern...** Schreibt das Tonsignal in eine Datei. Die verschiedenen Formate sind oben erklärt (Datei Öffnen...).

⇒ **Datei Löschen...** Löscht eine ausgewählte Datei nach Bestätigung.

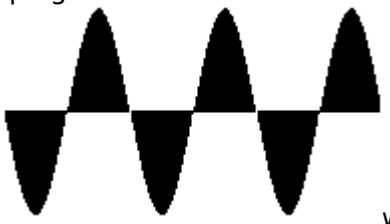
Bearbeiten

⇒ **Wiederholen**

Diese Option wiederholt das letzte Kommando, soweit dies möglich und sinnvoll ist

⇒ **Bearbeiten Auswahl**

⇒ **Spiegeln** spiegelt den markierten Teil des Spektrum in vertikaler Richtung



wird gespiegelt zu:

⇒ **Umkehren** dreht die Signalfolge in der Auswahl um

wird umgedreht zu:

⇒ **Zeit dehnen auf nnn%**

dehnt oder schrupft die Auswahl durch lineare Interpolation um vernünftige Ergebisse zu erzielen.
Der Effekt ist mit einer lokalen Änderung der Frequenz auf nnn% des vorherigen Wertes vergleichbar.

Wenn
auf 200% gedehnt wird, ergibt sich:

Bei einer Dehnung auf 50% ergibt sich:

⇒ **Verstärke auf nnn%** skaliert das Siganl im markierten Bereich mit einem Faktor nnn/100 .

Das Ergebnis ist eine lokale Änderung der Amplitude (Lautstärke). Wenn die Amplitude die Maximal- oder Minimalwerte überschreiten würde (±127) wird das Ergebnis beschnitten. Der Verstärkungsfaktor kann mit dem Menüpunkt Einstellungen-Vertärkung... festgelegt werden.

wird verstärkt zu:

⇒ **Einblenden** blendet das ausgewählte Spektrum mit einem linear von 0 % auf 100 % wachsenden Faktor ein.

wird nach Einblenden zu:

⇒ **Ausblenden** blendet das ausgewählte Spektrum mit einem linear von 100 % auf 0 % abnehmenden Faktor aus.

wird nach Ausblenden zu:

⇒ **Echo** Erzeugt ein Echo im ausgewählten Bereich des Spektrums. Wenn erforderlich wird die Länge des Spektrums vergrößert um ein Abschneiden des Echos zu vermeiden. Die Echo Parameter können unter dem Menüpunkt Einstellungen-Echo... angegeben werden.

⇒ **Kopieren** kopiert den ausgewählten Teil des Spektrums in die Zwischenablage.

⇒ **Ausschneiden** schneidet den ausgewählten Teil des Spektrums in die Zwischenablage aus.

⇒ **Löschen** löscht den ausgewählten Teil des Spektrums.

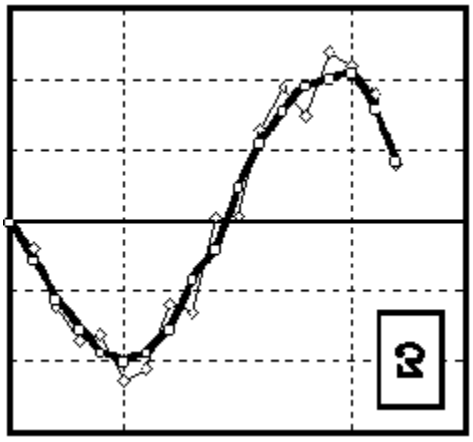
⇒ **Trimmen** stutzt die Daten auf den ausgewählten (beim Abspielen hörbaren Bereich) Teil des Spektrums zurück.

⇒ **Bearbeiten Zwischenablage**

- ⇒ **Neues Spektrum** fügt den Inhalt der Zwischenablge als *neues* Spektrum ein
- ⇒ **Einfügen** fügt den Inhalt der Zwischenablge in das aktuelle Spektrum an der durch den linken Schieber (Start) markierten Position ein.
- ⇒ **Anhängen** hängt den Inhalt der Zwischenablge an das aktuelle Spektrum an.
- ⇒ **Überlagern** überlagert dem aktuellen Spektrum den Inhalt der Zwischenablge beginnend an der durch den linken Schieber (Start) markierten Position; nützlich für Echo-Effekte. Falls nötig wird die Überlagerung auf die Länge des aktuellen Spektrums beschnitten.

Filter

⇒ **Mittel aus 3** Die aktuelle Auswahl wird so umgerechnet, daß jeder Samplewert durch den Mittelwert aus drei aufeinanderfolgenden Werten ersetzt wird. Die verwendeten Werte sind um den neuen Wert herum zentriert:

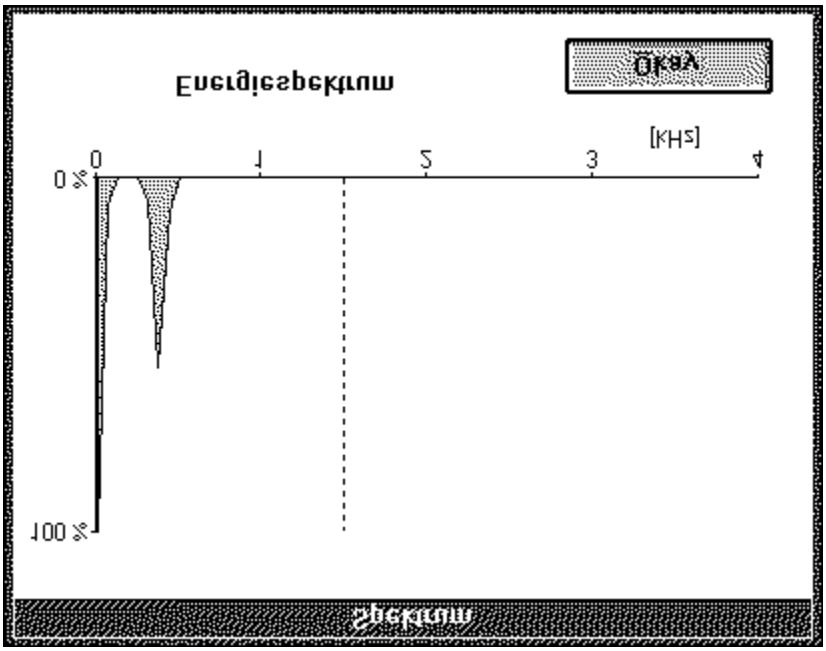


Die dicke Linie zeigt das Ergebnis der Anwendung des Mittelwertfilters Mittel aus 3 auf die Samples welche durch die dünne Linie dargestellt werden. Diese Filteroperation beseitigt bzw. glättet hochfrequente Störsignale. Feine Details können dabei verlorengehen.

⇒ **Mittel aus 5** Die aktuelle Auswahl wird so umgerechnet, daß jeder Samplewert durch den Mittelwert aus fünf aufeinanderfolgenden Werten ersetzt wird. Die fünf verwendeten Werte sind um den neuen Wert herum zentriert. Die Glättung ist stärker als beim Filter Mittel aus 3.

⇒ **Filter...** Hier können Sie die Ober- und Untergrenze für die Bandweite des Filters angeben. Wenn Sie sich das Leistungsspektrum des gewählten Ausschnitts ansehen möchten, markieren Sie das entsprechende Feld. Das Filter wird auf die aktuelle Auswahl angewendet, wenn Sie den Befehl Bearbeiten-Auswahl-Filtern ausführen.

⇒ **Filtern** Wendet die Filter-Routine auf den markierten Teil des Spektrums an. Sie können die Bandbreite des Filters unter dem Menüpunkt Einstellungen-Filter... eingeben.
SoundTool führt eine *Fast Hartley Transformation* aus, setzt die Koeffizienten der aus dem Band fallenden Anteile zu Null und wendet dann eine *Inverse Fast Hartley Transformation* an um die neuen Daten zu erzeugen. Wenn Sie Unter- und Obergrenze auf Null bzw. die Aufnahmefrequenz setzen, wird das Signal nicht verändert, aber SIe können sich das Frequenzspektrum ansehen (siehe: Einstellungen-Filter... weiter unten). Diese Filter Option arbeitet nicht perfekt, sie wird hoffentlich in späteren Versionen verbessert...
Hinweis: Filtern kann viel Rechenzeit beanspruchen; bearbeiten Sie zunächst einen kleineren Abschnitt von 1000 oder 2000 Bytes. Sie können den Vorgang allerdings auch durch Druck auf die [Escape] Taste abbrechen.
Ein typisches Ergebnis sieht so aus:



Das Beispiel enthält hauptsächlich Frequenzen im Bereich von 10 bis 500 Hz, woraus die beiden Spitzen in diesem Bereich resultieren.

Die gepunktete vertikalen Linie bei 1500 Hz zeigt die obere Grenze der Bandbreite des Filters an.

Tongenerator

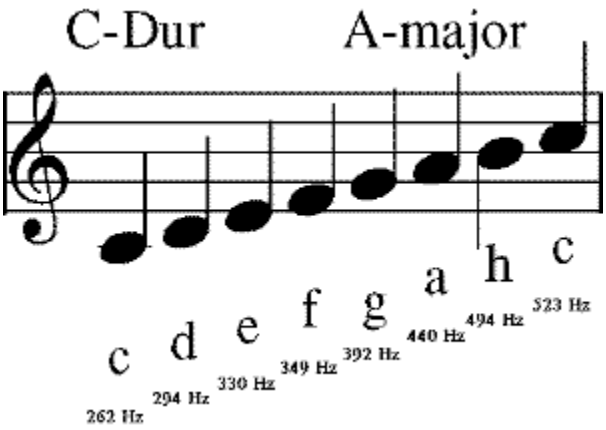
⇒ **Sinuswelle...** Nach Eingabe von Frequenz und dauer erzeugt SoundTool ein entsprechendes Tonsignal in Form einer Sinuswelle.

⇒ **Rechteckwelle...** wie oben, allerdings wird anstelle der Sinusfunktion ein Rechtecksignal verwendet.

⇒ **Sägezahnwelle...** wie oben, allerdings wird anstelle der Sinusfunktion ein Sägezahnsignal verwendet.

Anmerkung: Eine Frequenz von 440 Hz ergibt ein A der C-Dur Tonleiter. Dies entspricht der MIDI-Note 69. Alle Signale werden mit einer Abtastfrequenz von 12000 Hz erzeugt.

Die folgende Abbildung zeigt die Frequenzen für alle Noten der eingestrichenen Oktave in C-Dur:



⇒ **Funktion...** Nach Eingabe der gewünschten Anzahl **NS** und eines gültigen

Formelausdrucks in der Form **f(x)** wertet SoundTool die Formel aus, beginnend mit **x=0** ; **x** wird so lange um **1** erhöht, bis **x** den Wert **NS** erreicht.
Das erzeugte Signal entspricht also dem Wert von **f(x)** an jeder Stelle **x**.
Die Abtastfrequenz ist auf 12000 Hz festgelegt.

Interessante Funktionen sind zum Beispiel:

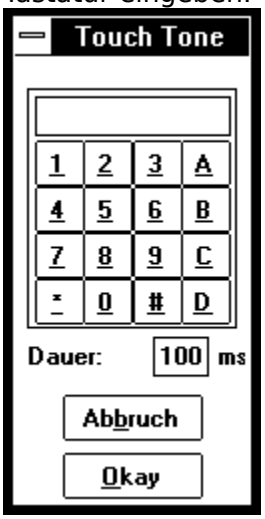
sin(x) oder *sin(0.1*x)* oder *sin((0.01*x)^1.5)*

Bei der Definition der Funktion ist zu berücksichtigen, daß die Ergebnisse in den Bereich von -1 bis +1 fallen müssen, größere bzw. kleinere Werte werden abgeschnitten.
Durch Auswahl der Option *Automatisch skalieren* kann SoundTool die Ergebnisse so skalieren, daß sie in den gültigen Bereich passen.

Die folgenden Grundfunktionen stehen zur Verfügung:

sin(x) cos(x) tan(x) sqrt(x) exp(x) ln(x) ^
*+ - * /*

⇒ **TouchTone...** In den USA ist im Telefonnetz das (Mehrfrequenz-) Tonwahlsystem gebräuchlich. Dabei wird jeder Ziffer bzw. jedem Buchstaben eine eindeutige Kombination von zwei Tonfrequenzen zugeordnet. SoundTool kann aus einer Telefonnummer das entsprechende Tonsignal erzeugen. Sie müssen dazu lediglich die entsprechenden Tasten im Dialogfeld TouchTone betätigen oder die Nummer über die Tastatur eingeben.



Sie können außerdem die Dauer des für jede Ziffer erzeugten Tonsignals angeben, ein Mindestwert von 100 ms (1/10 Sekunde) kann dabei nicht unterschritten werden.

Einstellungen

⇒ **Echo...** Geben Sie hier die Zahl der Echos und die Verzögerungszeit zwischen zwei Echos an:
Wenn die Verzögerung sehr gering ist, erhalten Sie ein Ergebnis, das sich wie eine direkte Rückführung anhört (Computerstimme).

⇒ **Aufnahme...** Hier können Sie die Länge der Aufnahme und weitere Aufnahme-Parameter eingeben.
(Steht nur zur Verfügung, wenn Sie eine A/D-Karte und eine passende Aufnahmebibliothek installiert haben).

⇒ **Verstärkung...** Geben Sie hier den Verstärkungsfaktor ein, der von der Menü-Option Bearbeitent-Auswahl-Verstärkung auf nnn% benützt wird.

⇒ **Dehnung...** Geben Sie hier den Dehnungsfaktor ein, der von der Menü-Option Bearbeiten-Auswahl-Zeit dehnen auf nnn% benützt wird.

⇒ **Zwischenablage...**
Wählen Sie hier die Formate, in denen Daten in die Zwischenablage kopiert werden sollen (Zusätzlich zum Standard Format CF_SOUND). Sie können den kopierten Ausschnitt im Text-Format (legt eine kurze Beschreibung der Daten in die Ablage) und in Metafile-Format (erzeugt eine vollständige Abbildung der Auswahl, ähnlich wie die Darstellung im Spektrum-Fenster).
Warnung: Metafile-Bilder können recht groß werden und viel Darstellungszeit benötigen,da sie die Daten mit einem Liniensegment pro Byte darstellen.

⇒ **Optionen...** Bietet die Möglichkeit zur Einstellung verschiedener Optionen. In dieser Version kann die Werkzeugleiste an- oder ausgeschaltet werden.

⇒ **Frequenzen...** Zeigt ein Dialogfeld an, in welchem Sie die im Kombinationsfeld Frequenzen zur Verfügung stehenden Werte ändern und ergänzen können. Die vorgenommenen Änderungen werden in SNDTOOL.INI vermerkt und stehen beim nächsten Programmstart automatisch zur Verfügung.

⇒ **Aufnahme...** Erlaubt die Eingabe zusätzlicher Parameter, soweit die Bibliothek Ihrer Soundkarte dies erfordert. (nur verfügbar wenn eine A/D-Karte und die passende SLL-Bibliothek installiert ist).

⇒ **Wiedergabe...** Erlaubt die Eingabe zusätzlicher Parameter, soweit die Bibliothek Ihrer Soundkarte dies erfordert. (nur verfügbar wenn eine D/A-Karte und eine passende SLL-Bibliothek installiert ist).

Information

⇒ **F1 Hilfe** Wie zum Teufel ist es Ihnen gelungen diesen Hilfstext anzuzeigen ?

⇒ **Über SoundTool...**
der übliche, langweilige Sermon, oder... vielleicht mal kurz abwarten...

Dateiformate

SoundTool kann Daten in verschiedenen Formaten speichern:

8 bit Roh: Dies ist nichts anderes als eine Folge von Bytes ohne zusätzliche Informationen. Der Wertebereich liegt zwischen 0 und 255.

ANSI: Dies entspricht dem obigen Rohformat, wird jedoch in lesbaren Textformat ausgegeben, ein Byte pro Zeile.

SUN Audio format, NeXT Audio format: 8 bit Samples nach einem μLaw Gesetz oder lineare 16 bit Samples. Diese Dateien verwenden folgenden Kopf:

```
char szMagic[4] = { ´.´,´s´,´n´,´d´ }
LONG_BE lStartOfData;/* Offset zu den Daten */
LONG_BE lDataSize;   /* Länge der Daten (optional) */
LONG_BE lEncoding;   /* 1 = 8-bit ISDN u-Law */
                     /* 3 = 16-bit linear PCM */
LONG_BE lSampleRate; /* Frequenz in [Hz] */
LONG_BE lChannels;   /* 1 = mono */
```

(LONG_BE bedeutet 32 bit Werte im big endian (Motorola) format.)
An diesen Kopf schließt sich der entsprechend formatierte Datenstrom an.

Sounder SND: wurde von Aaron Wallace definiert und wird von seinem Program Sounder benutzt. Es besteht aus einem kurzen Kopf von 32 Bytes von denen 8 Bytes benutzt werden:

```
WORD wSampleSize
WORD wFrequency
WORD wVolume
WORD wShift
```

An diesen Kopf schließt sich der rohe 8-Bit Datenstrom an.

SoundTool SND: enthält mehr Informationen in seinem Kopf:

```
char szMagic[6] = { ´S´,´O´,´U´,´N´,´D´, 0x1a }
GLOBALHANDLE hGSound;/* reserviert */
DWORD dwBytes;       /* Länge des gesamten Samples */
DWORD dwStart;       /* Erstes Byte das gespielt wird */
DWORD dwStop;        /* Erstes Byte, das NICHT mehr
                     gespielt wird */
WORD wFreq;          /* Frequenz */
WORD wSampleSize;
WORD wVolume;
WORD wShift;
char szName[96];     /* name of sound */
```

Der Teil diese Kopfes, welcher der magischen Nummer folgt, ist mit der Struktur identisch, die für den Austausch über die Zwischenablage verwendet wird.
An diesen Kopf schließt sich der rohe 8-Bit Datenstrom an.

SoundTool SNP: gepacktes Format:

```
char szMagic[6] = { ´S´,´N´,´D´,´P´,´K´, 0x1a }
GLOBALHANDLE hGSound;/* reserviert */
DWORD dwBytes;       /* Länge des gesamten Samples */
DWORD dwStart;       /* Erstes Byte das gespielt wird */
DWORD dwStop;        /* Erstes Byte, das NICHT mehr
                     gespielt wird */
WORD wFreq;          /* Frequenz */
WORD wSampleSize;
WORD wVolume;
WORD wShift;
```

char szName[96]; /* Name des Samples */

An diesen Kopf schließt sich der komprimierte Datenstrom an. Die Kompression erfolgt so, daß jeweils die Differenz zum vorhergehenden Wert gespeichert wird. Da diese Differenz meist sehr klein ist, kann sie in der Regel in einem Halb-Byte (4-Bit-Nibble) gespeichert werden. So ergibt sich eine Kompression auf etwa 65 % der Ausgangslänge bei einer schnellen Bearbeitung (ein Huffman Kompressor lieferte eine Reduktion auf 69% und benötigte die dreifache Zeit zum Schreiben bei gleicher Lesezeit, das Programm LHarc erzeugt eine Datei von 59 % der Ausgangslänge bei dreifacher Bearbeitungszeit).

WAVE: Dieses Format wird von den Multimedia-Erweiterungen innerhalb einer RIFF-Datei verwendet. SoundTool kann nur lineare 8-Bit PCM-Signale im Mono-Format lesen und schreiben.

Einträge in SNDTOOL.INI

SoundTool verwendet eine Datei SNDTOOL.INI um einige Parameters unter der Überschrift [SoundTool] zu speichern. Aufnahme- und Wiedergabe-Bibliotheken können dort ebenfalls Einträge vornehmen.wenn sie nicht die selben Schlüsselworte wie SoundTool verwenden.

[SoundTool]
; Pfad, der gespeichert wurde wenn Pfad speichern im Dialogfeld Datei öffnen gewählt wurde
path=d:\winsrc\sound\sun
; Info über Formate die in die Zwischenablage kopiert werden,
; ist die Summe der folgenden Werte:
; 0 = Standard SoundTool Sample.
; 1 = CF_TEXT, eine kurze Beschreibung des Samples
; 2 = CF_METAFILEPICT, eine Abbildung des Samples im Metafile Format
Clipboard=0
; Werkzeugleiste anzeigen ein/aus:
; 0 = aus
; 1 = ein
Ribbon=0
; Spektrum nach Filern anzeigen ein/aus:
; 0 = aus
; 1 = ein
Spectrum=0
; letztes gelesenen Dateiformat
; (Zahlen können sich in zukünftigen Versionen ändern)
FileType=...
; Wiedergabe-Bibliothek
PlayerDLL=D:\WINSRC\SOUND\SB\SBPLAYER.SLL
; Aufnahme-Bibliothek
RecorderDLL=D:\WINSRC\SOUND\SB\SBRECORD.SLL
; Frequenzliste (in aufsteigender Folge)
Frequency0=4000
Frequency1=8000
Frequency2=11000
Frequency3=22000

Die folgenden Schlüsselworte werden von meinen Wiedergabe-Bibliotheken verwendet::

; length of recorded sample
RecordBytes=1000000
; frequency at which we are sampling (may need additional adjustment)
RecordFrequency=11000
; used by btz13rec.sll:
; loop count to adjust the frequency
RecordDelay=192
; write to this port starts sampling of one byte.
RecordStartPort=0x0301
; the sampled byte can be read from this port:
RecordReadPort=0x0300

Datenstruktur für Transfer über die Zwischenablage

SoundTool registriert ein Zwischenablage-Format "CF_SOUND" das verwendet werdn kann um Audio-Daten zwischen Applikationen auszutauschen. Zwischenablagedaten im "CF_SOUND" Format bestehen aus einer Struktur mit algemeinen Daten, an die sich die Bytes anschließen, welche das eigentliche Tonsignal darstellen.

Die folgende Datenstruktur wird für den Austausch über die Zwischenablage und auch innerhalb von SoundTool verwendet:

```
#define  DESCR_LEN     96          /* max. Länge der Beschreibung */
typedef struct sound_tag
  {
   GLOBALHANDLE hGSound;           /* nicht benutzt für Zwischenablage */
   DWORD dwBytes;                  /* Länge des gesamten Samples */
   DWORD dwStart;                  /* erstes wiederzugebendes Byte */
   DWORD dwStop;                   /* erstes nicht mehr wiederzugebendes Byte
                     */
   unsigned short usFreq;
   unsigned short usSampleSize;
   unsigned short usVolume;
   unsigned short usShift;
   char szName[DESC_LEN];          /* Beschreibung des Datensatzes */
  } SAMPLE;
```

`usFreq` muß einen der folgenden Werte haben:
{ 5500, 7330, 11000, 22000 }

Zwei weitere Formate werden von SoundTool unterstützt: die Windows Standard Formate CF_TEXT und CF_METAFILEPICT.

Die CF_TEXT Repräsentation legt die folgende Beschreibung in die Ablage:

```
  Sound Sample
  Description:  Das hört sich an wie ein Beispiel
  Length:       884 Bytes
  Frequency:    22000 Hz
  Volume:       20
  Time:         19:26:19
  Date:         09/28/91
```

Die Abbildung im CF_METAFILEPICT Format enthält nur MoveTo und LineTo Abschnitte, die die Daten wiedergeben. Die Amplitudenwerte variieren von 0 bis 2550, der Zeitmaßstab entspricht im Verhältnis 1:1 den Bytes des Samples.

Beispiele zum Transfer über die Zwischenablage

Die folgenden beiden Programmfragmente zeigen, wie Audio-Daten über die Zwischenablage transferiert werden können.

```
/*************************************************************************/
static SAMPLE Sound;
/*************************************************************************/
BOOL CopySound( HWND hWnd )
/* copy a sound sample to the clipbaord */
{
   GLOBALHANDLE hGSample;
   SAMPLE FAR * lpSample;
   BYTE HUGE * lpCopySound;
   BYTE HUGE * lpSound;
   BOOL bReturn;
   DWORD dwBytes;

   bReturn = FALSE;
   dwBytes = min( Sound.dwBytes, (Sound.dwStop - Sound.dwStart) );
   if( NULL != (hGSample =
               GlobalAlloc( GMEM_MOVEABLE, sizeof(SAMPLE) + dwBytes )) )
     {
      if( NULL != (lpSample = (SAMPLE FAR *)GlobalLock( hGSample )) )
        {
         lpCopySound = sizeof(SAMPLE) + (BYTE HUGE *)lpSample;
         lpSound = (BYTE HUGE *)GlobalLock( Sound.hGSound );
         lpSound += Sound.dwStart;
         lpSample->dwBytes      = dwBytes;
         lpSample->dwStart      = 0;
         lpSample->dwStop       = dwBytes;
         lpSample->usFreq       = Sound.usFreq;
         lpSample->usSampleSize = Sound.usSampleSize;
         lpSample->usVolume     = Sound.usVolume;
         lpSample->usShift      = Sound.usShift;
         lstrcpy( lpSample->szName, Sound.szName);
         while( dwBytes-- )
           {
            *lpCopySound++ = *lpSound++;
           }
         GlobalUnlock( Sound.hGSound );
         GlobalUnlock( hGSample );

         if( OpenClipboard( hWnd ) )
           {
            EmptyClipboard();
            SetClipboardData( wFormat, hGSample );
            CloseClipboard();
            bReturn = TRUE;         /* yes, we finally did it ! */
           }
         else
           {
            ;/* cannot open clipbard, tell user about problem */
           }
        }
      else
```

```
        {
         ;/* cannot lock sample structure, tell user about problem */
        }
      }
   else
      {
       ;/* cannot allocate sample structure, tell user about problem */
      }

   return( bReturn);
}

/*************************************************************************/

BOOL PasteSound( HWND hWnd )
/* pastes sample from clipboard into next free slot */
{
   GLOBALHANDLE hGSample;
   SAMPLE FAR * lpSample;
   BYTE HUGE * lpCopySound;
   BYTE HUGE * lpSound;
   BOOL bReturn;
   DWORD dwBytes;

   bReturn = FALSE;

   if( wSounds >= MAXSOUNDS )
     {
      /* cannot paste any more sounds, tell user about problem */
      return( bReturn);
     }

   if( FALSE == OpenClipboard( hWnd ) )
     {
      /* cannot open clipboard, tell user about problem */
      return( bReturn);
     }

   if( NULL != (hGSample = GetClipboardData( wFormat )) )
     {
      if( NULL != (lpSample = (SAMPLE FAR *)GlobalLock( hGSample )) )
        {
         if( NULL != (Sound.hGSound =
                     GlobalAlloc( GMEM_MOVEABLE, lpSample->dwBytes )) )
           {
            if( NULL != (lpSound =
                        (BYTE HUGE *)GlobalLock( Sound.hGSound )) )
              {
               lpCopySound = sizeof(SAMPLE) + (BYTE HUGE *)lpSample;
               Sound.dwBytes =      lpSample->dwBytes;
               Sound.dwStart =      lpSample->dwStart;
               Sound.dwStop  =      lpSample->dwStop;
               Sound.usFreq  =      lpSample->usFreq;
               Sound.usSampleSize = lpSample->usSampleSize;
               Sound.usVolume =     lpSample->usVolume;
               Sound.usShift  =     lpSample->usShift;
               lstrcpy( (Sound.szName), lpSample->szName );
```

```
                dwBytes = Sound.dwBytes;
                while( dwBytes-- )
                  {
                   *lpSound++ = *lpCopySound++;
                  }
                GlobalUnlock( Sound.hGSound );
                bReturn = TRUE;   /* we finaly arrived here */
               }
             else
               {
                /* cannot lock destination array, free it */
                /* and tell user about problem */
                GlobalFree( Sound.hGSound );
               }
            }
          else
            {
             /* cannot alloc destination array, tell user about problem */
            }
          GlobalUnlock( hGSample );
         }
       else
         {
          /* cannot lock clipboard structure, tell user about problem */
         }
      }
   CloseClipboard();

   return( bReturn );
}
/*************************************************************************/
/*                          end of sample code                           */
/*************************************************************************/
```

Eine Bibliothek zur Wiedergabe von Audio-Signalen zufügen

SoundTool beinhaltet einen Mechanismus, über den ein Windows-Programmierer Funktionen zur Wiedergabe von Audio-Signalen hinzufügen kann, indem er eine DLL nach der im folgenden beschriebenen Schnittstelle hinzufügt. Jedesmal wenn SoundTool startet, wird eine installierte Wiedergabe Bibliothek in den Hauptspeicher geladen und die Menüleiste von SoundTool bietet einen Punkte um Einstellungen in diese Bibliothek auszuführen:

- Einstellungen → Wiedergabe...

Außerdem kann durch Betätigung der Schaltfläche Wiedergabe das aktuelle Signal über die Wiedergabe-Bibliothek abgespielt werden.

Damit die Bibliothek von SoundTool aufgerufen werden kann, muß sie mindestens sieben Funktionen exportieren, welche die Ordnungszahlen @1 bis @7 besitzen müssen:

@1 WEP, die übliche Windows Exit Prozedur wie sie von jeder dynamisch gebundenen Bibliothek benötigt wird.

@2 Diese Routine wird aufgerufen, wenn der Menüpunkt Einstellungen → Wiedergabe... ausgewählt wurde. Die Routine muß folgendermaßen aufgerufen werden können:

```
BOOL FAR PASCAL PlaySetup( HWND );
```

Die Routine sollte alle erforderlichen Parameter vom Anwender erfragen und im Datensegment der Bibliothek speichern. Da die Bibliothek beim Verlassen von SoundTool freigegeben wird, ist es ratsam die Parameter in SNDTOOL.INI unter der Überschrift [SoundTool] zu vermerken. Diese Parameter können zur Ladezeit der Bibliothek in `LibMain` eingelesen werden.

@3 Diese Routine wird aufgerufen, wenn die Schaltfläche Wiedergabe aktiviert wird. Der Funktions-Prototyp muß folgendermaßen aussehen:

```
void FAR PASCAL PlaySample( HWND, SAMPLE FAR * )
```

Der Erste Parameter ist SoundTool´s Fensterbezug der zweite Parameter zeigt auf eine SAMPLE Struktur die alle notwendigen Informationen enthält um das Signal abzuspielen.
Die Routine sollte nichts weiter tun als die Daten welche über den GLOBALHANDLE Bezug in der Struktur adressiert werden abzuspielen. Anfang und Ende sollten der Struktur entnommen werden. Der SAMPLE Zeiger gilt nur beim Aufruf der Prozedur, bei Wiedergabe im Hintergrund können sich die Daten verschieben;, allerdings bleibt der Bezug über GLOBALHANDLE gültig. Der Anwender kann die Daten nicht löschen solange sie wiedergegeben werden. Wenn die Wiedergabe beendet ist, **muß** die Routine die Meldung WM_PLAYDONE an SoundTool´s Fenster schicken. falls Sie das vergessen, werden Sie SoundTool nicht verlassen können.

@4 Diese Prozedur muß einen LONG Wert zurückgeben in dem eine Versionsnummer im LOWORD und eine Kennung im HIWORD enthalten ist (Beispiel siehe unten). Es ist unbedingt erforderlich, daß die Funktion die Kennung richtig setzt, andernfalls verweigert SoundTool die Zusammenarbeit mit der Bibliothek. Um dem Benutzer sichtbar zu machen für welches Gerät die Bibliothek gedacht ist, muß der Puffer auf den lpBuffer zeigt mit einer Identifikation beschrieben werden.
Falls die Bibliothek die Wiedergabe im Hintergrund durchführt (sofortige Rückkehr zu SoundTool beim Abspielen) muß die Versionsnummer mit der Flag 0x8000 geODERt werden. Hintergrund-Transfer erfordert üblicherweise das Anlegen eines (unsichtbaren) Fensters innerhalb der DLL und das Warten auf eine Nachricht vom Hardwaretreiber,

was unter Umständen nicht ganz einfach sein kann. Meine Sound Blaster Bibliothk macht so etwas...
Die Funktion wird folgendermaßen aufgerufen:

```
LONG FAR PASCAL GetPLAYDLLVersion( LPSTR lpBuffer, int nMaxLength )
```

@5 Diese Prozedur wird aufgerufen, wenn der Anwender während einer Wiedergabe die [ESC]-Taste drückt. Sie muß einen BOOL Wert zurückgeben der TRUE sein soll, wenn der Abbruch erfolgreich war. Außerdem muß SoundTool der Abbruch der Aufnahme durch PostMessage mit der Nachricht WM_PLAYDONE signalisiert werden.
BOOL FAR PASCAL StopPlayback(HWND)

@6 In Version 2.6 noch nicht verwendet. Die Prozedur muß einen BOOL Wert zurückgeben der TRUE sein soll, wenn die Unterbrechung erfolgreich war. SoundTool bleibt im Bereitschaftszustand bis die Wiedergabe beendet oder abgebrochen wird.
`BOOL FAR PASCAL Pause`Playback`( HWND )`

@7 In Version 2.6 noch nicht verwendet. Die Prozedur muß einen BOOL Wert zurückgeben der TRUE sein soll, wenn die Fortsetzung erfolgreich war.
`BOOL FAR PASCAL Continue`Playback`( HWND )`

Die folgenden Ausschnitte zeigen SPEAKER wobei Aaron Wallace´s DSOUND.DLL zur Wiedergabe verwendet wird. Die Bibliothek **muß** mindestens die 4 exportierten Ordnungszahlen aufweisen.

SPEAKER.DEF Datei zeigt EXPORTS mit Ordnungszahlen.

```
LIBRARY   SPEAKER
DESCRIPTION 'Player Library for use with SoundTool and IBM-PC speaker © Martin
                  Hepperle 1991'

EXETYPE   WINDOWS

CODE    PRELOAD MOVEABLE DISCARDABLE
DATA    MOVEABLE SINGLE

HEAPSIZE  1024

EXPORTS
    WEP                 @1 RESIDENTNAME
    PlaySetup           @2
    PlaySample          @3
    GetPLAYDLLVersion   @4
    StopPlayback       @5    ;necessary for SoundTool
    PausePlayback       @6    ;necessary for SoundTool
    ContinuePlayback       @7    ;necessary for SoundTool
    PlayDlgProc         @8
```

PlaySetup wird von SoundTool aufgerufen und soll Parameter vom Benutzer anfordern.

```
BOOL FAR PASCAL PlaySetup( HWND hWnd )
{
```

```
   FARPROC lpProcDialog;
   BOOL bReturn;

   lpProcDialog = MakeProcInstance( (FARPROC)PlayDlgProc, hInst);
   bReturn = DialogBox( hInst, "PLAY_DLG", hWnd, lpProcDialog);
   FreeProcInstance( lpProcDialog );

   return( bReturn );
}
```

PlaySample Routine, wird von SoundTool aufgerufen und gibt das übergeben Spektrum aus.

```
void FAR PASCAL PlaySample( HWND hWnd, SAMPLE FAR * lpSample )
/*
 * hWnd is the window handle of SoundTool.
 * must post a WM_PLAYDONE message to SoundTool when done.
 * This allows SoundTool to repeat the sample if necessary.
 */
{
   DWORD dwLength;
   BYTE FAR * lpBuffer;

   if( NULL == lpSample->hGSound )
      return;

   dwLength = min( lpSample->dwBytes,
                  (lpSample->dwStop - lpSample->dwStart) );

   if( 0L == dwLength )
      return;

   if( NULL != (lpBuffer = (BYTE HUGE *)GlobalWire( lpSample->hGSound )) )
     {
      GlobalPageLock( HIWORD(lpBuffer) );

      PlaySound( lpBuffer + lpSample->dwStart,
                 dwLength, lpSample->usFreq, lpSample->usSampleSize,
                 lpSample->usVolume, lpSample->usShift );

      GlobalPageUnlock( HIWORD(lpBuffer) );
      GlobalUnWire( lpSample->hGSound );
      PostMessage( hWnd, WM_PLAYDONE, 0, 0L );
     }
}
```

PlayDlgProc wird intern von PlaySetup gerufen und erfragt (eventuell benötigte) Benutzerparameter; diese Beispielbibliothek benötigt keine Daten.

```
BOOL FAR PASCAL PlayDlgProc( HWND hDlg, unsigned message, WORD wParam, LONG
                    lParam)
{
   switch( message )
     {
      case WM_COMMAND:
```

```
         if( ID_OK == wParam )
           {
            EndDialog( hDlg, TRUE );
           }
         break;

      case WM_INITDIALOG:
         return( FALSE );
         break;
     }
   return( FALSE );
}
```

LibMain Routine wird beim erstmaligen Laden durch LIBENTRY.ASM aufgerufen.

```
BOOL FAR PASCAL LibMain( HANDLE hInstance, WORD wDataSegment,
                         WORD cbHeapSize, LPSTR lpszCmdLine )
{
   hInst = hInstance;

   if( cbHeapSize > 0 )
      UnlockData( 0 );

   return( TRUE );
}
```

GetPLAYDLLVersion Routine wird von SoundTool aufgerufen u.a. wenn die DLL geladen wird.

```
LONG FAR PASCAL GetPLAYDLLVersion( LPSTR lpBuffer, int nMaxLength )
/* *must* return PLAY_MAGIC in its loword                              */
/* hiword should contain a version number multiplied by 100 (1000 == 10.00) */
/* if this is a background player or 0x8000 to the version number */
/* LONG FAR PASCAL GetPLAYDLLVersion() must have ordinal @4  */
{
   LoadString( hInst, S_MESSAGE+4, lpBuffer, nMaxLength );

   return( MAKELONG(VERSION,PLAY_MAGIC) );
}
```

StopRecord wird von SoundTool aufgerufen wenn [ESC] gedrückt wird.

```
BOOL FAR PASCAL StopPlayback( HWND hWnd )
{
   /* Tut nichts */
   PostMessage( hWnd, WM_PLAYDONE, 0, 0L );
   return( TRUE );
}
```

```
BOOL FAR PASCAL PausePlayback( HWND hWnd )
{
   /* Tut nichts */
```

```
   return( TRUE );
}
```

```
BOOL FAR PASCAL ContinuePlayback( HWND hWnd )
{
   /* Tut nichts */
   return( TRUE );
}
```

Globale Variable und **#defines** für SPEAKER

```
HANDLE hInst;            /* library instance handle */
char szAppName[] = "SoundTool";
#define VERSION 100              /* == 1.00 */
#define PLAY_MAGIC 0x5059        /* == 'PY' */
#define WM_PLAYDONE       WM_USER+1
```

Makefile um PLAYDLL zu erstellen (Microsoft nmake + C):

```
all: speaker.dll

speaker.obj: speaker.c speaker.h
    cl -c -Asnw -Gsw -Oas -Zpe -FPi -W3 speaker.c

libentry.obj: libentry.asm
    masm -Mx libentry,libentry;

speaker.res:   speaker.rc speaker.dlg speaker.h
               rc -r speaker.rc

speaker.sll: libentry.obj speaker.obj speaker.def speaker.res
    link speaker+libentry, speaker.sll,,/NOD /NOE sdllcew+libw+dsound,
                  speaker.def
    rc speaker.res speaker.sll
```

SPEAKER.DLG wird für SPEAKER.RC benötigt

```
PLAY_DLG DIALOG LOADONCALL MOVEABLE DISCARDABLE 9, 26, 186, 42
CAPTION "Wiedergabe Parameter"
STYLE WS_BORDER | WS_CAPTION | WS_DLGFRAME | WS_POPUP
BEGIN
    CONTROL "Nichts einzustellen.", -1, "static", SS_RIGHT | WS_GROUP |
                  WS_CHILD, 8, 22, 76, 10
    CONTROL "&Ok", ID_OK, "button", BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,
                  134, 6, 46, 14
END
```

Eine Bibliothek zur Aufnahme von Audio-Signalen zufügen

SoundTool beinhaltet auch einen Mechanismus zur Aufnahme von Audio-Signalen durch eine externe DLL nach der im folgenden beschriebenen Schnittstelle. Jedesmal wenn SoundTool startet, sucht es nach einer installierten Bibliothek, die in den Hauptspeicher geladen wird. Die Menüleiste von SoundTool bietet einen weiteren Punkt um diese Bibliothek aufzurufen:

- Einstellungen → Aufnahme...

Damit die Bibliothek von SoundTool aufgerufen werden kann, muß sie mindestens sieben Funktionen exportieren, die die Ordnungszahlen @1 bis @7 besitzen:

@1 WEP, die übliche Windows Exit Prozedur wie sie von jeder dynamisch gebundenen Bibliothek benötigt wird.

@2 Diese Funktion wird aufgerufen, wenn der Menüpunkt Einstellungen → Aufnahme... gewählt wird. Sie muß der folgenden Aufrufsequenz entsprechen:

```
BOOL FAR PASCAL RecordSetup( HWND );
```

Diese Routine soll vom Benutzer alle für die Aufnahme notwendigen Parameter erfragen und im Datensegment der Bibliothek speichern. Die Bibliothek wird freigegeben, wenn der Anwender SoundTool beendet, so daß es ratsam ist, die Parameter in der Datei SNDTOOL.INI unter der Überschrift [SoundTool] zu speichern. Diese Parameter können eingelesen werden, wenn `LibMain` beim Laden der Bibliothek aufgerufen wird.

@3 Diese Routine wird aufgerufen, wenn der Menüpunkt Aufnahme gewählt wurde. Die Funktion muß der folgenden Aufrufsequenz entsprechen:

```
BOOL FAR PASCAL RecordSample( HWND, SAMPLE FAR * );
```

Die Funktion sollte einen globalen Speicherblock für die Aufnahme allokieren, die Aufnahme durchführen und die SAMPLE Struktur mit den entsprechenden Daten füllen. Der Zeiger auf diese Struktur wird von SoundTool zur Verfügung gestellt und darf nicht geändert werden. Alle Elemente der Struktur müssen entsprechend der obigen Definition gesetzt werden (Transfer über die Zwischenablage). Die Routine muß nach Beendigung der Aufnahme unbedingt die Nachricht WM_RECDONE an SoundTool senden (mit PostMessage), sonst wartet SoundTool ewig. Damit ist es möglich im Hintergrund Aufzunehmen während andere Windows-Applikationen weiterlaufen.

@4 Diese Prozedur muß einen LONG Wert zurückgeben in dem eine Versionsnummer im LOWORD und eine Kennung im HIWORD enthalten ist (Beispiel siehe unten). Es ist unbedingt erforderlich, daß die Funktion die Kennung richtig setzt, andernfalls verweigert SoundTool die Zusammenarbeit mit der Bibliothek. Um dem Benutzer sichtbar zu machen für welches Gerät die Bibliothek gedacht ist, muß der Puffer auf den lpBuffer zeigt mit einer Identifikation beschrieben werden.
Falls die Bibliothek die Aufnahme im Hintergrund durchführt (sofortige Rückkehr zu SoundTool beim Abspielen) muß die Versionsnummer mit der Flag 0x8000 geODERt werden. Hintergrund-Transfer erfordert üblicherweise das Anlegen eines (unsichtbaren) Fensters innerhalb der DLL und das Warten auf eine Nachricht vom Hardwaretreiber, was unter Umständen nicht ganz einfach sein kann. Meine Sound Blaster Bibliothk macht so etwas...
Die Funktion wird folgendermaßen aufgerufen:

```
LONG FAR PASCAL GetRECDLLVersion( LPSTR, int )
```

@5 Drückt der Anwender während einer Aufnahme die [ESC]-Taste, so wird diese Routine

aufgerufen. Diese Prozedur muß einen BOOL Wert zurückgeben der TRUE sein soll, wenn der Abbruch erfolgreich war. Außerdem muß SoundTool der Abbruch der Aufnahme durch PostMessage mit der Nachricht WM_RECDONE signalisiert werden.
BOOL FAR PASCAL StopRecord(HWND)

@6 In Version 2.6 noch nicht verwendet. Die Prozedur muß einen BOOL Wert zurückgeben der TRUE sein soll, wenn die Unterbrechung erfolgreich war. SoundTool bleibt im Bereitschaftszustand bis die Aufnahme beendet oder abgebrochen wird.
`BOOL FAR PASCAL PauseRecord( HWND )`

@7 In Version 2.6 noch nicht verwendet. Diese Prozedur muß einen BOOL Wert zurückgeben der TRUE sein soll, wenn die Fortsetzung erfolgreich war.
`BOOL FAR PASCAL ContinueRecord( HWND )`

Die folgenden Beispiele zeigen Beispiele aus meiner Bibliothek RECDLL die einen 8-bit A/D Wandler verwendet um Audio Signale mit bis zu 40 kHz aufzunehmen.

RECDLL.DEF zeigt die EXPORTS Anweisung mit Ordnungszahlen.

```
LIBRARY   RECDLL
EXETYPE   WINDOWS
CODE                PRELOAD MOVEABLE DISCARDABLE
DATA                MOVEABLE SINGLE
HEAPSIZE  1024
EXPORTS
    WEP                 @1 RESIDENTNAME  ;necessary for Windows
    RecordSetup         @2     ;necessary for SoundTool
    RecordSample        @3     ;necessary for SoundTool
    GetRECDLLVersion    @4     ;necessary for SoundTool
    StopRecord       @5    ;necessary for SoundTool
    PauseRecord       @6    ;necessary for SoundTool
    ContinueRecord       @7    ;necessary for SoundTool
    RecordDlgProc       @8    ;used internally by RECDLL
```

RecordSetup wird von SoundTool aufgerufen und erfragt verschiedene Parameter vom Anwender.

```
BOOL FAR PASCAL RecordSetup( HWND hWnd )
{
   FARPROC lpProcDialog;
   BOOL bReturn;

   lpProcDialog = MakeProcInstance( (FARPROC)RecordDlgProc, hInst);
   bReturn = DialogBox( hInst, "RECORD_DLG", hWnd, lpProcDialog);
   FreeProcInstance( lpProcDialog );

   return( bReturn );   /* return TRUE if OK */
}
```

RecordSample wird von SoundTool aufgerufen und gibt die Aufnahmedaten zurück..

```
void FAR PASCAL RecordSample( HWND hWnd, SAMPLE FAR * lpSample )
```

```
{
   GLOBALHANDLE hGSound;
   BYTE HUGE * lpSound;
   BYTE HUGE * lpSoundStart;
   DWORD dwElapsed, dwStartTick, dwStopTick, dwBytes, dwFrequency;
   char szFormat[80];

   dwBytes = dwRecordBytes;

   if( NULL != (hGSound = GlobalAlloc( GMEM_MOVEABLE, dwBytes )) )
     {
      if( NULL != (lpSound = (BYTE HUGE *)GlobalWire( hGSound )) )
        {
         lpSample->hGSound = hGSound;
         lpSample->dwBytes = dwBytes;
         lpSample->dwStart = 0;
         lpSample->dwStop  = dwBytes;
         lpSample->usSampleSize = 0;
         lpSample->usVolume = 20;
         lpSample->usShift  = 4;
         MessageBeep(0);
         lpSoundStart = lpSound;
         dwStartTick = GetTickCount(); /* ms */
         if( nRecordDelay )
           {
            _asm
              {
               mov dx, usStartPort              /* start first conversion */
               out dx, al
              }
            while( dwBytes-- )
              {
               _asm
                 {
                  mov dx, usReadPort
                  in al, dx                     /* get a byte from A/D
                   converter */
                  les bx,DWORD PTR lpSound
                  mov BYTE PTR es:[bx],al
                  mov dx, usStartPort           /* start next conversion */
                  out dx, al
                  mov cx, nRecordDelay          /* loop counter */
                  WaitLoop:
                  loop WaitLoop                 /* empty loop */
                 }
               lpSound++;                       /* increment huge pointer */
              }
           }
         else  /* fastest sampling rate */
           {
            _asm
              {
               mov dx, usStartPort
               out dx, al                       /* start next conversion */
              }
            while( dwBytes-- )
              {
```

```
            /* get a byte from A/D converter */
            _asm
              {
               mov dx, usReadPort
               in al, dx                    /* get a byte from A/D
                converter */
               les bx,DWORD PTR lpSound
               mov BYTE PTR es:[bx],al
               mov dx, usStartPort
               out dx, al                   /* start next conversion */
              }
            lpSound++;                      /* increment huge pointer */
           }
        }
      dwStopTick = GetTickCount(); /* ms */
      MessageBeep(0);

      lpSound = lpSoundStart;
      dwBytes = lpSample->dwBytes;
      while( dwBytes-- )
        {
         if( *lpSound > 127 )
            *lpSound -= 128;
         else
            *lpSound += 128;
         *lpSound++;
        }
      GlobalUnWire( hGSound );

      if( dwStopTick < dwStartTick )
         dwElapsed = 0xffffffff - dwStartTick + dwStopTick;
      else
         dwElapsed = dwStopTick - dwStartTick;
      dwFrequency = (DWORD)(1000.0 * ((double)lpSample->dwBytes /
                (double)dwElapsed));
      lpSample->usFreq  = (unsigned int)dwFrequency;
      LoadString( hInst, S_MESSAGE+3, szFormat, sizeof(szFormat)-1 );
      wsprintf( szBuffer, szFormat, dwFrequency );
      lstrcpy( lpSample->szName, szBuffer );
     }
    else
      {
       /* cannot lock new sound bytes */
       GlobalFree( hGSound );
       lpSample->hGSound = NULL;
       ShowMessageBox( hWnd, 1, MB_OK );
      }
   }
  else
   {
    /* cannot allocate memory for new sound bytes */
    ShowMessageBox( hWnd, 2, MB_OK );
    lpSample->hGSound = NULL;
   }
  PostMessage( hWnd, WM_RECDONE, 0, 0L );
}
```

RecordDlgProc wird von RecordSetup aufgerufen und erfragt verschiedene Parameter vom Anwender.

```
BOOL FAR PASCAL RecordDlgProc( HWND hDlg, unsigned message, WORD wParam, LONG
                  lParam)
{
   int nDelay;
   DWORD dwBytes;

   switch( message )
     {
      case WM_COMMAND:
         if( ID_OK == wParam )
           {
            GetDlgItemText( hDlg, ID_RECORDNUMBER, szBuffer, sizeof(szBuffer )
                  );
            dwBytes = (DWORD)atol( szBuffer );
            GetDlgItemText( hDlg, ID_RECORDDELAY, szBuffer,
                  sizeof(szBuffer ) );
            nDelay = atoi( szBuffer );

            if( nRecordDelay != nDelay )
              {
               nRecordDelay = nDelay;
               wsprintf( szBuffer, "%d", nRecordDelay );
               WriteProfileString( szAppName, szRecordDelay, szBuffer );
              }

            if( dwRecordBytes != dwBytes )
              {
               dwRecordBytes = dwBytes;
               wsprintf( szBuffer, "%lu", dwRecordBytes );
               WriteProfileString( szAppName, szRecordBytes, szBuffer );
              }
            EndDialog( hDlg, TRUE );
           }
         else if( ID_CANCEL == wParam )
           {
            EndDialog( hDlg, FALSE );
           }
         break;

      case WM_INITDIALOG:
         wsprintf( szBuffer, "%lu", dwRecordBytes );
         SetDlgItemText( hDlg, ID_RECORDNUMBER, szBuffer );
         wsprintf( szBuffer, "%d", nRecordDelay );
         SetDlgItemText( hDlg, ID_RECORDDELAY, szBuffer );
         SetFocus( GetDlgItem(hDlg,ID_RECORDDELAY) );
         return( FALSE );
         break;
     }
   return( FALSE );
}
```

LibMain, wird beim Laden der Bibliothek von LIBENTRY.ASM aufgerufen.

```
BOOL FAR PASCAL LibMain( HANDLE hInstance, WORD wDataSegment,
                         WORD cbHeapSize, LPSTR lpszCmdLine )
{
   hInst = hInstance;

   /* get stored parameters from SNDTOOL.INI */
   GetPrivateProfileString( szAppName, szRecordBytes, "?",
                       szBuffer, sizeof(szBuffer), szIniFile );
   if( '?' == szBuffer[0] )
     {
      /* no entry found, use default */
      dwRecordBytes = 50000;
      wsprintf( szBuffer, "%lu", dwRecordBytes );
      WritePrivateProfileString( szAppName, szRecordBytes, szBuffer, szIniFile
                  );
     }
   else
     {
      dwRecordBytes = (DWORD)atol( szBuffer );
     }
   GetPrivateProfileString( szAppName, szRecordDelay, "?",
                       szBuffer, sizeof(szBuffer), szIniFile );
   if( '?' == szBuffer[0] )
     {
      /* no entry found, use default */
      nRecordDelay = 0;
      wsprintf( szBuffer, "%d", nRecordDelay );
      WritePrivateProfileString( szAppName, szRecordDelay, szBuffer, szIniFile
                  );
     }
   else
     {
      nRecordDelay = atoi( szBuffer );
     }
   GetPrivateProfileString( szAppName, szRecordPort, "?",
                       szBuffer, sizeof(szBuffer), szIniFile );
   if( '?' == szBuffer[0] )
     {
      /* no entry found, use default */
      usPort = 0x300;
      wsprintf( szBuffer, "%u", usPort );
      WritePrivateProfileString( szAppName, szRecordPort, szBuffer,
                  szIniFile );
     }
   else
     {
      usPort = (unsigned int)atoi( szBuffer );
     }


   if( cbHeapSize > 0 )
      UnlockData( 0 );     /* make segment moveable */

   return( TRUE );
}
```

GetRECDLLVersion wird von SoundTool aufgerufen wenn die DLL geladen wird.

```
LONG FAR PASCAL GetRECDLLVersion( LPSTR lpBuffer, int nMaxLength )
{
   LoadString( hInst, S_MESSAGE+4, lpBuffer, nMaxLength );
   return( MAKELONG(VERSION,REC_MAGIC) );
}
```

StopRecord wird von SoundTool aufgerufen wenn [ESC] gedrückt wird.

```
BOOL FAR PASCAL StopRecord( HWND hWnd )
{
   /* Tut nichts */
   PostMessage( hWnd, WM_RECDONE, 0, 0L );
   return( TRUE );
}
```

```
BOOL FAR PASCAL PauseRecord( HWND hWnd )
{
   /* Tut nichts */
   return( TRUE );
}
```

```
BOOL FAR PASCAL ContinueRecord( HWND hWnd )
{
   /* Tut nichts */
   return( TRUE );
}
```

Globale Variablen und **#defines** für RECDLL

```
#define VERSION 100   /* == 1.00 */
#define REC_MAGIC 0x5243  /* == 'RC' */
#define WM_RECDONE        WM_USER+2
DWORD dwRecordBytes;     /* Zahl der Bytes für Aufnahme */
int nRecordDelay;        /* Verzögerungszähler */
HANDLE hInst;            /* Instance handle der Bibliothek */
unsigned int usPort;     /* A/D Wander Port Adresse */
char szBuffer[80];       /* Puffer auf dem Stapel vermeiden (DS != SS) */
char szAppName[] = "SoundTool";
char szRecordBytes[] = "RecordBytes";
char szRecordDelay[] = "RecordDelay";
char szRecordPort[] = "RecordPort";
```

Makefile zur Erstellung von RECDLL

```
all: recdll.dll

recdll.obj: recdll.c recdll.h
    cl -c -Asnw -Gsw -Oas -Zpe -FPi -W3 recdll.c
```

```
libentry.obj: libentry.asm
    masm -Mx libentry,libentry;

recdll.res:   recdll.rc recdll.dlg recdll.h
               rc -r recdll.rc

recdll.dll: libentry.obj recdll.obj recdll.def recdll.res
    link recdll+libentry, recdll.dll,,/NOD /NOE sdllcew+libw, recdll.def
    rc recdll.res recdll.dll
```

RECDLL.DLG wird für RECDLL.RC benötigt

```
RECORD_DLG DIALOG LOADONCALL MOVEABLE DISCARDABLE 9, 26, 186, 42
CAPTION "Aufnahme Parameter"
STYLE WS_BORDER | WS_CAPTION | WS_DLGFRAME | WS_POPUP
BEGIN
    CONTROL "&Länge der Aufnahme:", -1, "static", SS_RIGHT | WS_GROUP |
                    WS_CHILD, 10, 8, 74, 10
    CONTROL "", ID_RECORDNUMBER, "edit", ES_LEFT | WS_BORDER | WS_TABSTOP |
                    WS_CHILD, 90, 8, 32, 12
    CONTROL "&Verzögerungszähler:", -1, "static", SS_RIGHT | WS_GROUP |
                    WS_CHILD, 8, 22, 76, 10
    CONTROL "", ID_RECORDDELAY, "edit", ES_LEFT | WS_BORDER | WS_TABSTOP |
                    WS_CHILD, 90, 22, 32, 12
    CONTROL "&Abbruch", ID_CANCEL, "button", BS_PUSHBUTTON | WS_GROUP |
                    WS_TABSTOP | WS_CHILD, 134, 6, 46, 14
    CONTROL "&Ok", ID_OK, "button", BS_DEFPUSHBUTTON | WS_TABSTOP | WS_CHILD,
                    134, 24, 46, 14
END
```

Verwendung von Dynamischem Daten Austausch (DDA)

SoundTool besitzt eine einfache DDA Schnittstelle die benützt werden kann um Audio Möglichkeiten zu anderen Anwendungen hinzuzufügen, wenn diese DDA unterstützen.
Sie können Kommandos in SoundTool ausführen oder den aktuellen Zustand per DDA erfragen.
Es gibt ein Thema (General) und ein Objekt (State) über das Sie sich mit SoundTool unterhalten können. SoundTool beachtet Groß- und Kleinschreibung nicht, wenn es DDA Nachrichten bearbeitet.
Sie können also General ebenso wie GENERAL oder GeNeRaL verwenden.

Befehle in SoundTool ausführen:
Der Anwender kann Audiosignale aufnehmen und abspielen indem er SoundTool einen Befehl über DDA sendet:
Es gibt zwei Kommandos, die als Teil einer *DDE-Execute* Nachricht versandt werden können:

Record.Data("*soundname*")
Play.Data("*soundname*")

Jeder der beiden Befehle benötigt ein Argument, eine Zeichenkette die den Namen unter dem die Daten gespeichert werden sollen angibt (max. 8 Buchstaben und eine Pfadangabe, *keine Erweiterung*).
DDA Befehle können nur mit gepackten Dateien (SNP) ausgeführt werden !
Bei der *Aufnahme* via DDA legt SoundTool eine Datei mit der Erweiterung SNP an und schreibt die Audiodaten im komprimierten Format auf Platte nachdem der Befehl Record.Data ausgeführt wurde.
Die *Wiedergabe* von gepackten Audiodateien erfolgt auf die gleiche Weise, wobei SoundTool an "*soundname*" die Endung ".SNP" anhängt, bevor es versucht die Datei zu laden.
Befehlszeilen müssen in eckigen Klammern [ ] eingeschlossen sein.

Abfrage von Information für SoundTool:
Statusinformation:

Der Anwender kann den augenblicklichen Zustand von SoundTool erfragen indem er eine DDA-Anfrage für das Objekt State startet. SoundTool wird eine mit Null abgeschlossene Zeichenkette im CF_TEXT Format zurückgeben die einer der folgenden Ketten entspricht:

"*Bereit*",
"*Aufnahme*" *and*
"*Wiedergabe*".

Versionsnummer:

Der Anwender kann die Versionsnummer von SoundTool erfragen indem er eine DDA-Anfrage für das Objekt Version startet. SoundTool wird eine mit Null abgeschlossene Zeichenkette im CF_TEXT Format mit der Versionsnummer zurückgeben. Der Rückgabewert der augenblicklichen Version lautet:

"*3.0*"

Beispiele:
Um mit SoundTool zu kommunizieren sind folgende Schritte nötig:
1. öffnen Sie einen DDA Kanal zum Thema General
2. senden Sie das gewünschten Kommando einmal oder mehrfach
3. schließen Sie den DDA Kanal

Beispiel unter Verwendung von Microsoft Excel:
Makro um eine Aufnahme abzuspielen:

	A	B
1	KANAL.ÖFFNEN("SoundTool";"General")	DDE Konversation beginnen
2	AUSFÜHREN(A1;"[Play.Data(""c:\tmp\msg1""]")	Befehl ausführen lassen
3	KANAL.SCHLIESSEN(A1)	Konversation beenden

Beispiel unter Verwendung von Microsoft Excel:
Makro um den aktuellen Zustand von SoundTool zu erfragen:

	A	B
1	KANAL.ÖFFNEN("SoundTool";"General")	DDE Konversation beginnen
2	ABFRAGEN(A1;"State")	State Information anfordern
3	KANAL.SCHLIESSEN(A1)	Konversation beenden

Es besteht die Möglichkeit eine heiße (oder warme) Verbindung zwischen einer Anwendung und SoundTool anstelle eines Makros zu verwenden. Dies wird für alle Objekte, die angefordert werden können unterstützt; in SoundTool gibt es nur eines: State.
Bei Verwendung einer heißen (warmen) Verbindung ist es nicht notwendig einen DDA Kanal zu öffenen, dies muß die Applikation automatisch erledigen.

Beispiel unter Verwendung von Microsoft Excel:
Ausgabe des Status von SoundTool in einer Zelle einer Tabelle. Der Zelleninhalt wird automatisch aktualisiert, wenn sich der Status von SoundTool ändert:

	A	B
1	='SoundTool'|'General'!'State'	eine heiße Verbindung zu SoundTool

martin@mecha2.verfahrenstechnik.uni-stuttgart.de (eine INTERNET Adresse)

aaron@jessica.stanford.edu (eine INTERNET Adresse)