

Maestix

COLLABORATORS

	<i>TITLE :</i> Maestix	
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>
WRITTEN BY		January 17, 2023
<i>SIGNATURE</i>		

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	Maestix	1
1.1	Maestix: Inhaltsverzeichnis	1
1.2	Maestix: English Introduction	3
1.3	Maestix: Einleitung	3
1.4	Maestix: Features	4
1.5	Maestix: Neuigkeiten	5
1.6	Maestix: Copyright	5
1.7	Maestix: Copyright-Bestimmungen	5
1.8	Maestix: Updates	7
1.9	Maestix: Support	8
1.10	Maestix: Kontaktadresse	8
1.11	Maestix: Bugs	10
1.12	Maestix: History	11
1.13	Maestix: Zukunft	12
1.14	Maestix: Installation	13
1.15	Maestix: Das DAT-Delay	13
1.16	Maestix: Shell-Tools	13
1.17	Maestix: SetMstx	13
1.18	Maestix: AllocMstx und FreeMstx	14
1.19	Maestix: Technik	14
1.20	Maestix: Benutzung der Library	16
1.21	Maestix: Allokieren	16
1.22	Maestix: Setzen	17
1.23	Maestix: Status-Abfrage	20
1.24	Maestix: Datenübertragung	21
1.25	Maestix: Realisation	22
1.26	Maestix: Senden	22
1.27	Maestix: Bitmanipulation	23
1.28	Maestix: Empfangen	23
1.29	Maestix: Echtzeitverarbeitung	24

1.30	Maestix: Echtzeiteffekte	24
1.31	Maestix: Muting-Effekt	26
1.32	Maestix: Bypass-Effekt	26
1.33	Maestix: ChannelSwap-Effekt	26
1.34	Maestix: LeftOnly-Effekt	26
1.35	Maestix: RightOnly-Effekt	27
1.36	Maestix: Mono-Effekt	27
1.37	Maestix: Surround-Effekt	27
1.38	Maestix: Volume-Effekt	27
1.39	Maestix: Karaoke-Effekt	28
1.40	Maestix: Foreground-Effekt	28
1.41	Maestix: Spatial-Effekt	28
1.42	Maestix: Echo-Effekt	29
1.43	Maestix: Mask-Effekt	30
1.44	Maestix: Offset-Effekt	30
1.45	Maestix: Demo-Sources	30
1.46	Maestix: AnalyzeInput	31
1.47	Maestix: CSBchanger	31
1.48	Maestix: LevelWindow	31
1.49	Maestix: Realtime	32
1.50	Maestix: RealtimeEcho	32
1.51	Maestix: SineTone	32
1.52	Maestix: SineTone2	33
1.53	Maestix: Surround	33
1.54	Maestix: Credits	33

Chapter 1

Maestix

1.1 Maestix: Inhaltsverzeichnis

TriTech Developments Proudly Presents

```
__ ____ /  
^ ^ / _ ( _ | · V  
/ X - \ ____ ) | | ^ V 39  
///
```

-- INHALTSVERZEICHNIS --

English Introduction

1. **Einleitung** Wie die Library entstand
 - 1.1 **Features** Das kann die Library
 - 1.2 **Neuerungen** was ist neu in dieser Version
 2. **Copyright** Benutzung und Verbreitung
 - 2.1 **Copyright** Bitte unbedingt lesen!
 - 2.2 **Updates** Neue Versionen, woher?
 - 2.3 **Support** bei Problemen mit der Lib
 - 2.4 **Kontaktadresse** des Autors
 - 2.5 **Bekannte Bugs** und wie man sie umgeht
 - 2.6 **History** Sämtliche Änderungen
 - 2.7 **Zukunft** Was ist geplant?
 3. **Installation** der Library
 - 3.1 **Das DAT-Delay** Wozu braucht man das?
 4. **Shell-Tools** Die zugehörigen Shell-Tools
 - 4.1 **SetMstx** Einstellen der Parameter
 - 4.2 **AllocMstx** Belegen der Karte
 5. **Über die Technik** Wie funktioniert alles?
 6. **Benutzung der Lib** Arbeiten mit der Lib
-

- 6.1 **Allokieren** Belegen der Karte
 - 6.2 **Setzen** des Eingangs, Ausgangs etc.
 - 6.3 **Status-Abfrage** Karten-Status
 - 6.4 **Datenübertragung** Ein- und Ausgabe per Message
 - 7. **Realisation** von Anwendungen
 - 7.1 **Bitmanipulation** Änderung der Steuerbits
 - 7.1 **Senden** Senden von Daten
 - 7.2 **Empfangen** Empfangen von Daten
 - 7.3 **Echtzeitverarb.** Empfangen und senden
 - 7.4 **Echtzeiteffekte** Die Echtzeiteffekte
 - 7.4.1 **RFX_Muting** Stummschaltung
 - 7.4.2 **RFX_Bypass** kein Effekt
 - 7.4.3 **RFX_ChannelSwap** Links <-> Rechts
 - 7.4.4 **RFX_LeftOnly** nur Links
 - 7.4.5 **RFX_RightOnly** nur Rechts
 - 7.4.6 **RFX_Mono** Mono-Signal
 - 7.4.7 **RFX_Surround** Hintergrundkanal
 - 7.4.8 **RFX_Volume** Lautstärkeregelung
 - 7.4.9 **RFX_Karaoke** Gesangsunterdrückung
 - 7.4.10 **RFX_Foregnd** Vordergrundkanal
 - 7.4.11 **RFX_Spatial** Basisbreite
 - 7.4.12 **RFX_Echo** Echo/Hall
 - 7.4.13 **RFX_Mask** Maskieren/Quantisieren
 - 7.4.14 **RFX_Offset** Gleichspannungsoffset
 - 8. **Demo-Sources** des Pakets
 - 8.1 **AnalyzeInput** Eingangssignal-Analyse
 - 8.2 **CSBchanger** Ändern der Subcodes
 - 8.3 **LevelWindow** Einlesen von Signalen
 - 8.4 **Realtime** Ein Echtzeiteffekt
 - 8.5 **RealtimeEcho** Ein Echtzeit-Echoeffekt
 - 8.6 **SineTone** Ausgabe von Signalen (fixed)
 - 8.7 **SineTone2** Variable Ausgabe
 - 8.8 **Surround** Echtzeitverarbeitung
 - A. **Credits** Danksagungen
-

1.2 Maestix: English Introduction

Hi!

Sorry, but I had not yet enough time to translate this documentation into English. The most important contents of this guide are:

- This packet is FreeWare. You are allowed to copy it freely as long as the packet is entire and unchanged. You are allowed to charge a fee of max. 8 DM (or an equivalent amount of other currencies). I hereby permit the inclusion on Fred Fish and AmiNet CD.
- You are using this library at your own risk! I am not responsible for any damages, data loss or blown up Amigas.
- It is NOT the fault of MacroSystem NOR does it represent the quality of their products if this library should fail.

If you have any questions or comments, or if you've found a bug or want to translate the German docs into your language, feel free to contact me:

shred@tfh.dssd.sub.org (international since August 1995)

Have fun!

maestix.library · © 1995 Richard Körber · All Rights Reserved

1.3 Maestix: Einleitung

Die MaestroPro-Soundkarte ist eine Karte mit hervorragenden Eigenschaften zur Audioverarbeitung in höchster Qualität oder zum Backuppen von Daten.

Der Benutzer ist allerdings auf die kleine Handvoll Programme beschränkt, die es für die MaestroPro gibt. Im Gegensatz zur Toccata-Soundkarte besitzt die MaestroPro nämlich momentan noch keine Library, mit der man selbst Programme für diese Karte schreiben kann.

Ich wollte diesem Mißstand ein Ende bereiten und diese Library programmieren. Leider konnte MacroSystem mich aus zeitlichen Gründen nicht unterstützen, und Hardware-Unterlagen über die Karte waren nicht vorhanden. Hier wurde mir klar, daß ich mich selbst "ans Werk" machen mußte...

Als erstes besorgte ich mir die Unterlagen über die zwei Spezialchips auf der Karte. Danach ermittelte ich die Startadresse der Karte durch den Eintrag in der AutoConfig-Liste. Nachdem ich die Registeradressen ermittelte, bekam ich durch Messungen schnell den größten Teil derer Belegung heraus. Ein paar weitere Experimente mit diesen Registern

brachte dann endgültigen Aufschluß über die Funktionsweise und Programmierung der Karte.

Diese Library ist nun das Endergebnis der zahlreichen Experimente. Ich habe sie ausgiebig und teilweise sogar unter extremen Bedingungen geprüft, ohne auch nur einen Fehler festgestellt zu haben.

Ich möchte Sie in diesem Zusammenhang trotzdem darauf aufmerksam machen, daß diese Library alleine auf experimenteller Basis entstand, ohne jegliche Unterlagen oder Informationen von MacroSystem. Ich kann daher Fehler oder Inkompatibilitäten nicht ausschließen. Da die Library nach außen hin jedoch vollständig unabhängig von der Hardwareprogrammierung ist, dürfte ich nahezu alle Probleme beseitigen können, ohne daß Sie Ihre Programme ändern müssen.

.....

||

| Wenn sich die Library fehlerhaft verhält, ist dies keinesfalls |
| die Schuld von MacroSystem, und es sagt auch nichts über die |
| Qualität ihrer Produkte aus! |

||

.....

1.4 Maestix: Features

Dies sind die Fähigkeiten der `maestix.library`:

- Empfangen und senden über die digitale Schnittstelle mit einer Auflösung von 16 Bit und einer Rate von max. 48kHz, Stereo
 - Auch gleichzeitiges Senden und Empfangen möglich
 - benötigt mindestens Kickstart-Version 2.04
 - unterstützt S/P-DIF und AES/EBU (Rundfunkstudio-Modus)
 - Senden und Empfangen der User Data Bits
 - Unterstützt Umschaltzeiten von DAT-Recordern
 - 14 integrierte Echtzeiteffekte, eigene Echtzeiteffekte möglich
 - unterstützt alle mir bekannten Features der MaestroPro Soundkarte
 - unterstützt auch ältere Versionen der MaestroPro
 - nach außen hin hardwareunabhängiges Konzept, dadurch flexible Erweiterungsmöglichkeiten
 - Keine Kenntnisse über die MaestroPro oder über digitale Datenübertragungsverfahren notwendig.
 - automatische Unterstützung beliebig vieler Datenpuffer mit Queue-Verwaltung
-

- durch Message-Übergaben kann der Klient auf neue Puffer warten
- unterstützt virtuellen Speicher
- keine (bekanntes ;^) Enforcer- und Mungwall-Hits
- Unterstützt alle Sprachen. C und Assemblerincludes sind inklusiv!
- 100% System-Konform, keine Hardware-Hacks

1.5 Maestix: Neuigkeiten

Für die V39 wurden die Zusatztools "SetMstx", "AllocMstx" und "FreeMstx" komplett neu geschrieben.

Die Maestix unterstützt ab jetzt die Umschaltzeit, die der DAT-Recorder benötigt, um sich auf eine neue Rate umzustellen.

Außerdem sind drei weitere Echtzeiteffekte hinzugekommen: Echo, Mask und Offset.

1.6 Maestix: Copyright

Bitte lesen Sie die folgenden Abschnitte aufmerksam durch.

Sollten Sie mit den Copyright-Bestimmungen nicht einverstanden sein, dann löschen Sie dieses Paket und alle dazugehörigen Dateien umgehend!

1.7 Maestix: Copyright-Bestimmungen

Sie verwenden die maestix.library, wie sie ist, und auf Ihr eigenes Risiko. Die Library und alle zugehörigen Dateien erheben keinen Anspruch auf Vollständigkeit oder Richtigkeit.

Der Autor (Richard Körber) ist nicht verantwortlich für alle Schäden, die direkt oder indirekt mit der Benutzung der Library in Verbindung stehen. Dies schließt Schäden an der Hardware oder Datenverlust ein, ist aber nicht begrenzt darauf.

Alle Rechte sind vorbehalten, insbesondere Veränderungen der Vermarktungsform der Library oder das der Wartung und Verbesserung.

Die Maestix-Library ist FreeWare! Das bedeutet, daß die Library zwar urheberrechtlich geschützt ist, aber kostenlos verwendet und weitergegeben werden darf, ohne daß eine Lizenz vom Autor eingeholt werden muß.

Verbreitung

Das Paket darf weiterkopiert, auf PD-Disketten oder CD-ROM gesammelt

oder in Mailboxen angeboten werden. Es ist jedoch nicht zulässig, mehr als 8 DM (acht Deutsche Mark) für alle Aufwendungen zu verlangen (dies schließt Sortier- und Kopieraufwand sowie das Medium oder evtl.

Verkaufskosten ein). Ansonsten muß vorher eine schriftliche Genehmigung vom Autor eingeholt werden!

Das Paket darf nur vollständig und unverändert weitergegeben werden!

Es ist lediglich eine Archivierung des kompletten Pakets durch die üblichen Programme (lha, lzx, lzh, arj, shrink etc.) gestattet.

Die Aufnahme in die PD-Reihe Fred Fish oder AmiNet wird hiermit ausdrücklich gestattet, ebenso die Verwendung in deren CD-Reihe.

Benutzung

Sie dürfen die Library in Ihren Programmen verwenden, ohne daß eine Lizenz einzuholen ist. Es gelten jedoch folgende Bedingungen:

- Bei allen kostenpflichtigen Programmen (ShareWare, CrippleWare, kommerzielle Software) ist mir eine vollständig lauffähige und registrierte Version kostenlos zuzusenden.
- Kommerzieller Software ist außerdem das komplette Maestix-Paket hinzuzufügen.
- Im Programm und dessen Anleitung muß erwähnt werden, daß die maestix.library von Richard Körber verwendet wird.
- In der Anleitung muß erwähnt werden, daß die Library auf eigene Gefahr verwendet wird und daß eventuelle Fehlfunktionen weder die Schuld von MacroSystem ist noch etwas über die Qualität ihrer Produkte aussagt.

Die dem Paket beiliegenden Demo-Sourcecodes können in Ihren Programmen frei verwendet werden.

Mit der MaestroPro ist es durchaus möglich, den SCMS-Kopierschutz abzustellen und beliebig viele digitale Kopien anzufertigen. Bitte beachten Sie hier die Urheberrechtsbestimmungen, nach der Kopien nur für den privaten Gebrauch angefertigt werden dürfen! Der Autor ist nicht verantwortlich für den Mißbrauch der Library und den dadurch eventuell entstehenden Schaden.

Bestandteile des Pakets

Das Paket ist nur vollständig mit folgenden Dateien:

maestix/c/AllocMstx

maestix/c/FreeMstx

maestix/c/SetMstx

maestix/demos/AnalyzeInput
maestix/demos/AnalyzeInput.s
maestix/demos/CSBchanger
maestix/demos/CSBchanger.s
maestix/demos/LevelWindow
maestix/demos/LevelWindow.s
maestix/demos/Realtime
maestix/demos/Realtime.s
maestix/demos/RealtimeEcho
maestix/demos/RealtimeEcho.s
maestix/demos/SineTone
maestix/demos/SineTone.s
maestix/demos/SineTone2
maestix/demos/SineTone2.s
maestix/demos/Surround
maestix/demos/Surround.s
maestix/includes/fd/maestix_lib.fd
maestix/includes/libraries/maestix.h
maestix/includes/libraries/maestix.i
maestix/includes/pragmas/maestix_pragmas.h
maestix/includes/proto/maestix_protos.h
maestix/includes/maestix_lib.i
maestix/libs/maestix.library
maestix/Install-D
maestix/Install-D.info
maestix/Install-E
maestix/Install-E.info
maestix/maestix.doc
maestix/maestix.doc.info
maestix/maestix.dok
maestix/maestix.dok.info
maestix/Maestix.guide
maestix/Maestix.guide.info
maestix.info
FILE_ID.DIZ

1.8 Maestix: Updates

Sie können Updates der Library in jeder gut sortierten Mailbox oder im AmiNet finden.

Die neueste Version finden Sie jedoch immer in der Support-Mailbox und gelegentlich auch im AmiNet.

1.9 Maestix: Support

Wenn Sie Fragen oder Probleme haben, oder Fehler gefunden haben, dann benutzen Sie bitte primär meine E-Mail-Adresse.

Eine Hotline stelle ich nicht zur Verfügung, da ich keine Zeit dafür habe und außerdem nicht noch in der Nacht belästigt werden möchte (ich habe diesbezüglich schon die besten Geschichten gehört!).

.....

||

| !!! WICHTIG !!! Senden Sie keine Anfragen bezüglich der Library |
| an MacroSystem und verwenden Sie auch nicht de- |
| ren Hotline! MacroSystem steht in keiner Verbin- |
| dung mit der maestix.library. |

||

.....

1.10 Maestix: Kontaktadresse

Wenn Sie mich anschreiben möchten, verwenden Sie bitte meine E-Mail-Adresse:

shred@tfh.dssd.sub.org (ab August 1995 international)

WICHTIG: Die E-Mail-Adresse "shred@eratosth.wwb.sub.de" ist ungültig geworden! Bitte verwenden Sie ausschließlich die oben genannte Adresse.

In der Support-Mailbox können ständig die neuesten Updates unter dem Brett "Shred's Corner" bezogen werden. Außerdem steht dort für die Maestix ein Brett zu Verfügung, in dem Probleme und Ideen diskutiert oder Maestix-Sources getauscht werden können.

O-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-O

||

| KÖLN'S GRÖßTE AMIGA & PC MAILBOX: |

| __ __ |

! | ZyXEL...19.2kdb /\ | .G.I.G.A.bYtE .fASTCALl SYStEM. :

: |__ ZyXEL...19.2kdb / |__ ..68030/33MhZ...: :

: \ZyXEL...16.8kdb /\ .O.N.I.I.N.E.: :

: __/ hSt.dUA1.14.4kdb (_ o __/ ::::5:NODES::::: :

: CTK.....14.4kdb :.MUltiUSERChAt.: :

::

O-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-O

NOT ONLY FUN ... _____
 _L_____ I~ T _____
 T~ / \ T T | ___ / Y / Y T ~ //
 T | _____ / \ | | T | ___ / | | _____ / ___ /
 H | \ | \ _ | l _ j | _ T | | \ | \ V
 E | _ ___ / | | _ | | T | | | _ / _ l _
 | Y | l _ \ | | | l _ / \
 | | | / \ | _ _ | | / \
 l _ _ | l _ _ _ _ / _ j \ _ j T | _ j _ _ _ _ j _ _ _ _ ^ _ ^ _ _ _ _ /
 [*]=====l___j===== [PRECiSiON-ADVERTISEMENT]===[*]
 O-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-O
 : :
 : >>> IHR 23-PFENNIG-ORTSNETZBREREICH <<< :
 : :
 : *** KÖLN (0221) *** LEVERKUSEN (0214) *** WESSELING (02236) *** :
 : *** KÖLN-PORZ (02203) *** RÖSRATH (02205) *** OPLADEN (02171) *** :
 : *** BERG. GL. (02202) *** LEICHLINGEN (02175) *** OVERATH (02206) *** :
 : *** SIEGBURG (02241) *** LOHMAR (02246) *** KÜR TEN (02268) *** :
 : *** MUCH (02245) *** DABRINGHAUS.(02193) *** BURSCHEID (02174) *** :
 . .
IEITUNG.1...:...o22o4.61630...hST.dUAL14.4oo... .
IEITUNG.2...:...o22o4.61499...ZYXEL.+19.2oo... .
IEITUNG.3...:...o22o4.23856...ZYXEL...16.8oo... .
IEITUNG.4...:...o22o4.23955...CTK.....14.4oo... .
IEITUNG.5...:...o22o4.Elite...ZYXEL.+19.2oo... .
 . .
 . 100 Anrufer pro Tag können sich nicht irren !! .
 . .
 O-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-O

1.11 Maestix: Bugs

Geschrieben wurde Maestix auf einem Amiga 4000/o3o/MMU mit Kick 3.0,
 10 MB RAM, FastLane Z3 mit 1GB HD, und einer MaestroPro :-)
 Getestet wird Maestix ständig auf folgenden Rechnern:
 · A4000/o3o (MMU) , Kick 3.0 ,10 MB RAM (8F/2C), 1GB HD,
 Fastlane, MaestroPro, CyberVision (2MB)
 · A4000/o3o (MMU,FPU) , Kick 3.0 , 6 MB RAM (4F/2C), 400MB HD,
 Emplant, MaestroPro

· A3000/o3o (MMU,FPU) , Kick 3.1 ,10 MB RAM (8F/2C), 1,4GB HD, MaestroPro, CyberVision (4MB)

Folgende Bugs sind in dieser Version bekannt:

· Eine Datenübertragung durch die maestix.library stört die Datenübertragung auf der internen seriellen Schnittstelle mit hohen Übertragungsraten erheblich. Momentan rate ich daher ab, gleichzeitig zur Datenübertragung die serielle Schnittstelle zu benutzen! Ich habe zwar schon eine Idee zur Lösung des Problems, kann aber leider nicht versprechen, ob dieser Bug überhaupt behoben werden kann.

· Die aktiven Demoprogramme hängen sich auf, wenn die Karte bereits belegt wurde. Das Problem ist die Beendigung des zweiten Prozesses, da in diesem Fall ein Deadlock entsteht. Es ist kein Fehler der maestix library. Fixing soon...

Folgende Enforcer-Hits oder Mungwall-Hits sind bekannt:

· keine bekannt :-)

1.12 Maestix: History

V39.00 · AllocMstx, FreeMstx komplett neu geschrieben. Unterstützen jetzt auch QUIET

· SetMstx komplett neu geschrieben. Unterstützt jetzt auch DELAY, QUIET und FORCE

· DAT-Delay eingebaut. Maestix wartet jetzt, bis der DAT-Recorder auf die neue Rate eingestellt ist.

· Drei neue Effekte: Echo, Mask und Offset

V38.10 · Ab jetzt richtige C-Includes

· Überarbeitete Autodocs

V38.00 · Ältere Maestros zeigten Probleme bei Sender-Synchronisation [Thomas Wenzel]

· Echtzeit-Effekt-Funktionen eingebaut

· 11 Echtzeit-Effekte eingebaut

V37.30 · Bug in Studio-Modus-Bits entfernt.

V37.20 · ResetLSA eingebaut

V37.10 · Ältere Maestros zeigten einige Probleme bei der Belegung und der Statusabfrage! [Thomas Wenzel]

· Bug in der Copyprotection-Erkennung: die Zustände waren genau vertauscht

V37.00 · Fehler in der C-Includedatei beseitigt

- UDBs können jetzt auch gelesen werden
- AllocMstx und FreeMstx geschrieben
- V36.20 · Weitere Optimierungen
- Autodocs ins Englische übersetzt
- Fehler im Surround-Demoprogramm entfernt: beim Startup keine Messages an Transmitter geschickt
- V36.10 · Zeitkritische Programmteile zum Teil erheblich optimiert
- Fehler in GetStatus (MSTAT_TFIFO und MSTAT_RFIFO) entfernt: Error wurde nicht korrekt ausgegeben
- 68040-CopyBack-Proof
- V36.00 · SetMstx geschrieben
- FlushTransmit() und FlushReceive() geschrieben
- V35.10 · Der Fall, daß zwar der externe Eingang gewählt wurde, dort aber kein Signal anliegt, wird jetzt auch korrekt behandelt.
- Einen Installer-Script geschrieben
- V35.00 · Library vollständig neu konzipiert und geschrieben, da das alte Konzept einige gravierende Mängel aufwies
- UDBs werden unterstützt
- Der Studio-Modus wird unterstützt
- V34.30 · Bugfix: gelegentlich wurden die Kanäle vertauscht
- V34.20 · der Interrupt-Server wurde etwas optimiert
- durch einen Bug gelegentlicher "Aufhänger" des Interrupts
- V34.10 · alle bis dato bekannten Bugs und Enforcer-Hits entfernt
- V34.00 · Grundversion

1.13 Maestix: Zukunft

Folgendes ist für die nächsten Versionen der Library geplant:

- Unterstützung mehrerer MaestroPro's :)
- Unterstützung der alten Maestro-Karte
- Konfig-Editor für Standard-Eingang, Delay etc.
- Automatische Eingangs-Selektion
- Verbesserte Verträglichkeit mit der seriellen Schnittstelle
- Sharing des Transmit- und Receive-Zweigs für zwei unabhängige Tasks.

Bitte betrachten Sie diese Liste nicht als verbindliche Zusage!

1.14 Maestix: Installation

Die Installation wird durch den Commodore-Installer bewerkstelligt. Es wird die Library ins LIBS:-Verzeichnis kopiert, sowie die Shell-Programme ins C:-Verzeichnis.

Der Installer fragt, welcher Maestro-Eingang standardmäßig verwendet wird. Je nachdem werden die Parameter für das SetMstx-Programm eingestellt, wenn es in die User-Startup eingefügt wird.

Die Installation ist dann abgeschlossen.

1.15 Maestix: Das DAT-Delay

1.16 Maestix: Shell-Tools

Dem Maestix-Paket sind noch Shell-Tools beigelegt, welche ins C:-Verzeichnis kopiert werden sollten. Diese Tools sind auch für die Endanwender interessant!

1.17 Maestix: SetMstx

Das Programm stellt die Default-Parameter der maestix.library ein. Der Installer bindet es in die User-Startup ein, es kann aber auch jederzeit aufgerufen werden, um die Parameter zu verändern.

SetMstx kennt folgende Parameter:

INPUT=[optical|coaxial] stellt den Default-Eingang ein. Zur Erkennung reicht der erste Buchstabe aus. Üblicherweise ist dies der gleiche Eingang, der auch auf der Karte als Default-Eingang eingestellt wurde.

DELAY=[time/ms] stellt die DAT-Wartezeit ein. Wenn die Rate am MaestroPro-Ausgang wechselt, wartet die maestix.library den eingestellten Zeitraum, damit das DAT-Gerät sich auf die neue Rate einstellen kann.

Siehe [Delay](#)

QUIET Sämtliche Ausgaben werden unterdrückt.

FORCE Normalerweise sind die Änderungen nur dann erlaubt, wenn die MaestroPro nicht benutzt wird. Mit dieser Option wird der Schutz umgangen.

Beispiel:

```
SetMstx INPUT=C DELAY=2000 FORCE
```

1.18 Maestix: AllocMstx und FreeMstx

Die ursprünglichen Programme, die die MaestroPro verwenden (also Samplitude und MaestroBR), belegen direkt die Hardware, da die Library zu dieser Zeit noch nicht existierte. Die Library könnte in diesem Fall die Karte ein zweites mal belegen.

Um dies zu verhindern, habe ich AllocMstx und FreeMstx geschrieben.

AllocMstx kennzeichnet die Hardware für die Maestix als belegt, ohne sie jedoch wirklich zu belegen. Sollte die Karte bereits belegt worden sein, schlägt AllocMstx mit einem Returncode 10 fehl.

FreeMstx gibt dementsprechend die Maestro wieder frei. Diese Funktion kann auch aufgerufen werden, wenn AllocMstx fehlschlug.

Beide Programme werden ohne Parameter aufgerufen. Optional kann QUIET angegeben werden, wodurch sämtliche Ausgaben unterbunden werden.

Zum Start von den Originalprogrammen empfiehlt sich jetzt ein Script wie:

```
----- >8 ----- SCHNIPP -----
CD SYS:Sampling ; oder anderes Zielverzeichnis
FailAt 20
C:AllocMstx QUIET
IF NOT WARN
Samplitude-MS >NIL: ; das Programm starten
C:FreeMstx QUIET
ELSE
C:RequestChoice >NIL: "Samplitude" "MaestroPro ist belegt!" "Okay"
ENDIF
----- 8< ----- SCHNAPP -----
```

1.19 Maestix: Technik

Ich fasse hier einmal zusammen, wie digitale Daten über die IEC-958-Schnittstelle übertragen werden, damit Ihnen ein paar Zusammenhänge verständlich werden.

Die IEC-958-Schnittstelle gibt es im Heimbereich in zwei Varianten durch optische und koaxiale Übertragung. Bei der optischen Übertragung wird ein rotes Licht (660nm) übertragen, wodurch normalerweise je nach Qualität der Lichtleiter Entfernungen bis 5m überbrückt werden können. Die koaxiale Übertragung hat zwar Potentialprobleme, kann dafür aber preiswert und über längere Distanzen erfolgen. Außerdem kann eine

digitale Quelle auf mehrere Empfänger verteilt werden, was bei der optischen Übertragung nicht ohne weiteres möglich ist.

Die Datenübertragung erfolgt seriell, wobei Takt und Daten ineinander verwoben sind. Die Taktrate nimmt je nach verwendeter Abtastrate verschiedene Werte an. Bei einer Abtastrate von 32kHz (DSR, DAT-Longplay) beträgt die Datenrate 2,048MBit/s, bei 44,1kHz (CD) 2,8224MBit/s, bei 48kHz (DAT) sogar 3,072MBit/s.

Der Datenfluß ist in Blöcke von je 12288 Bit eingeteilt, wobei jeder Block aus 192 Rahmen (engl. Frames) besteht. Jeder Rahmen ist 64 Bit groß und in zwei Subframes unterteilt, die je einen Abtastwert des linken und rechten Kanals enthalten.

Das Subframe beginnt mit einer Präambel aus 4 Bit, die angibt, zu welchem Kanal das Datenwort gehört. Ein Block beginnt immer mit dem linken Kanal.

Nach der Präambel folgt das Audio-Datenwort mit einer Breite von 24 Bit, wovon im Heimbereich aber nur 16 Bit verwendet werden. Die MaestroPro ist nur in der Lage, ein 16 Bit breites Datenwort zu übertragen und zu empfangen.

Darauf folgt ein Validity-Bit, das angibt, daß das Datenwort gültig ist (es ist dann 0). Im Heimbereich ist dieses Bit üblicherweise immer auf 0 gesetzt und wird von wenigen Empfängern berücksichtigt. Das Validity-Bit kann über die `maestix.library` gesteuert werden

Das vorletzte Bit ist das User Data Bit. Dort können Informationen zur Langzeitsynchronisation oder auch andere Daten abgelegt werden; im Heimbereich ist dieses Bit jedoch nicht von Bedeutung. Die MaestroPro kann einen 32 Bit großen Bereich der UDBs senden und einen 8 Bit großen Bereich empfangen.

Zuletzt kommt das Channel Status Bit. Hier werden sämtliche Informationen über die Art des Audio-Signals übertragen, wie Kanalzahl, Abtastrate, Emphasis, Sender-Typ sowie dem Kopierschutzbit. Hier wird zwischen Heim- und Studiomodus unterschieden. Die MaestroPro unterstützt auf der Senderseite beide Formate, der Empfänger kann allerdings nur den Heim-Standard verarbeiten.

Abgeschlossen wird das Subframe mit einem Parity-Bit, das der Fehlerkorrektur und Synchronisierung dient. Das Parity-Bit erzeugt die MaestroPro automatisch.

Literaturquellen: ELRAD 9/92, Digitale Audiodaten-Schnittstelle
ELRAD 1/95, ICs für die digitale Audiotechnik
ELRAD 2/95, ICs für die digitale Audiotechnik
DIN EN 60958

1.20 Maestix: Benutzung der Library

Im Folgenden wird beschrieben, wie die Library aufgebaut ist, und wie die einzelnen Funktionen angewendet werden.

1.21 Maestix: Allokieren

Bevor die MaestroPro überhaupt verwendet werden darf, muß sie erst einmal über die Library belegt werden. Dadurch wird außerdem verhindert, daß zwei Programme gleichzeitig auf die Maestro zugreifen können.

Die zugehörige Funktion heißt `AllocMaestro()`. Ihr wird ein Zeiger auf eine Tagliste übergeben. Momentan sind noch keine Tags vorgesehen, es sollte also immer eine `NULL` oder ein Zeiger auf `TAG_DONE` übergeben werden. In zukünftigen Versionen kann durch die Tags beispielsweise eine von mehreren Maestro's ausgewählt werden, oder bestimmte globale Parameter eingestellt werden.

Das Ergebnis dieser Funktion ist ein Zeiger auf die `MaestroBaseStruktur`, der für die anderen Funktionsaufrufe benötigt wird. Wird eine `NULL` als Ergebnis übergeben, existiert entweder die `MaestroKarte` nicht, oder sie wurde bereits von einem anderen Programm belegt.

Wenn die Karte nicht mehr benötigt wird (beispielsweise am Programmende), muß sie mit `FreeMaestro()` wieder freigegeben werden. Der Funktion wird ein Zeiger auf die `MaestroBase-Struktur` übergeben. Ein Problem will ich nicht verschweigen: es gibt nämlich Probleme mit den Programmen, die ursprünglich für die `MaestroPro` geschrieben wurden, also `MaestroBR` und `Samplitude`. Diese Programme benutzen die `MaestroPro-Karte` direkt, ohne sie vorher durch die Library zu belegen (was nur natürlich ist, da die Library erst nach ihnen entstand).

Dadurch können Konflikte mit diesen Programmen und der `maestix.library` entstehen, wenn beide die Karte gleichzeitig benutzen möchten. Um dieses Problem zu lösen, wurden die Programme `AllocMstx` und `FreeMstx` geschrieben.

Programme, die die Karte über die `maestix.library` belegen, können sich die Karte untereinander allerdings nicht "abjagen"!

Die `MaestroBase-Struktur` ist übrigens privat. Aus Kompatibilitätsgründen ist es nicht zulässig, sie zu lesen oder zu verändern. Die Zusammensetzung kann (und wird) sich in den nächsten Versionen durchaus ändern!

1.22 Maestix: Setzen

Nachdem die MaestroPro belegt wurde, stellt sie sich auf Grundparameter ein. Diese werden je nach Anwendung erst einmal entsprechend eingestellt. Hierfür dient die Funktion SetMaestro(). Ihr wird der Zeiger auf die MaestroBase sowie ein Zeiger auf eine Tag-Liste übergeben.

Ein Teil der Tags stellt die Channel Status Bits und User Data Bits ein. Wenn der Ausgang auf OUTPUT_INPUT oder OUTPUT_FIFO umgeschaltet wird, werden diese Informationen in den Subframes dem Datenstrom hinzugemischt.

Für die Channel Status Bits existieren folgende Tags:

MTAG_SetCSB (ULONG) setzt die 32 Channel Status Bits direkt.

Dieser Tag sollte nach Möglichkeit nicht verwendet werden, da er nicht sehr kompatibel ist.

MTAG_Studio (BOOL) gibt an, ob der Consumer- oder der Studio-Modus verwendet werden soll. Der Default-Wert ist FALSE, was den Heimbereich aktiviert. Sobald dieser Tag verwendet wird, werden alle CSBs auf die Defaultwerte zurückgesetzt.

MTAG_CopyProh (ULONG) gibt an, welche Art von Kopierschutz verwendet werden soll. Im Studio-Modus ist dieser Tag wirkungslos. Es stehen folgende Modis zur Verfügung:

CPROH_OFF deaktiviert den Kopierschutz ganz.

Es können beliebig viele digitale Kopien angefertigt werden.

CPROH_ON aktiviert den Kopierschutz. Es kann dann lediglich noch eine digitale Kopie angefertigt werden.

CPROH_PROHIBIT setzt den Kopierschutz. Eine digitale Aufnahme des Ausgangssignals ist dann nicht mehr möglich.

CPROH_INPUT Der Kopierschutz wird nach dem Eingangssignal auf CPROH_OFF oder CPROH_ON eingestellt. Der Decoder unterscheidet allerdings nicht zwischen CPROH_ON und CPROH_PROHIBIT. Wenn kein Eingangssignal vor-

liegt, ist der Kopierschutz abgeschaltet.

MTAG_Emphasis (ULONG) gibt an, welche Emphasis das Signal verwendet.

Es sind folgende Modis möglich:

EMPH_OFF Das Signal wurde ohne Emphasis aufgezeichnet. (Default)

EMPH_50us Das Signal wurde mit einer 50 μ s-Emphasis aufgezeichnet (entspricht EMPH_ON).

EMPH_INPUT schaltet je nach Eingangssignal die Emphasis aus oder ein. Ist das Eingangssignal nicht vorhanden, ist die Emphasis immer aus.

Im Studio-Modus gibt es außerdem:

EMPH_CCITT aktiviert den CCITT J.17-Modus.

EMPH_MANUAL die Emphasis wird am Empfänger manuell eingestellt.

MTAG_Source (ULONG) gibt einen Quellen-Kategoriecode für den Consumer-Bereich an. Möglich sind:

SRC_DAT Quelle ist ein DAT-Gerät (Default)

SRC_CD Quelle ist ein CD-Player

SRC_DSR Quelle ist ein DSR-Gerät

SRC_ADCONV Quelle ist ein A/D-Converter

SRC_INSTR Quelle ist ein Musikinstrument

SRC_INPUT wählt je nach Eingangstyp zwischen

SRC_DAT und SRC_CD. Fehlt das Eingangssignal, wird SRC_DAT ausgewählt.

MTAG_Rate (ULONG) gibt die Sampling-Rate an. Beachten Sie hier, daß die Sampling-Rate nur angegeben wird, aber nichts mit der tatsächlichen Rate zu tun hat. Sie müssen sich selbst darum kümmern, daß die Ausgaberate dementsprechend ist. Ansonsten erkennen die meisten Empfänger das Signal nicht an. Es gibt hier:

RATE_48000MANU 48kHz-Rate, Manuell verstellbar (nur im Studio-Modus)

RATE_48000 48kHz-Rate (DAT) (Default)

RATE_44100 44,1kHz-Rate (CD)

RATE_32000 32kHz-Rate (DSR)

RATE_INPUT wählt die Rate gemäß dem Eingangssignal. Fehlt das Eingangssignal, wird immer 48kHz ausgewählt.

Desweiteren steuern folgende Tags die Subcode-Daten:

MTAG_Validity (BOOL) gibt an, ob die ausgehenden Daten gültig sind.

In diesem Fall ist MTAG_Validity:=TRUE (Default).

MTAG_SetUDB (ULONG) setzt die ersten 32 Bits der User Data Bits.

MTAG_ResetUDB Ist dieser Tag angegeben, wird die Ausgabe der UDBs abgeschaltet.

Die MaestroPro selbst wird mit folgenden Tags gesteuert:

MTAG_Input (ULONG) wählt den Eingangsmodus aus. Es gibt hier folgende Möglichkeiten:

INPUT_STD wählt den vom Benutzer angegebenen Standard-Eingang aus (Default).

INPUT_OPTICAL wählt den optischen Eingang aus.

INPUT_COAXIAL wählt den koaxialen Eingang aus.

INPUT_SRC48K wählt eine interne Taktquelle aus, welche eine konstante Rate von 48kHz liefert.

Wenn INPUT_SRC48K nicht angewählt wurde, hängt die Systemrate von der Rate am gewählten Eingang ab. Liegt dort kein Signal an, wird automatisch eine 48kHz-Rate erzeugt. Die _INPUT-Werte und die Status-Werte werden automatisch angepaßt.

MTAG_Output (ULONG) wählt den Ausgangsmodus aus. Es gibt hier folgende Möglichkeiten:

OUTPUT_BYPASS Das Signal vom gewählten Eingang wird direkt an den Ausgang gelegt. Die Subcodes werden dabei nicht manipuliert. (Default)

OUTPUT_INPUT Das Signal vom Eingang wird decodiert und anschließend mit den neuen Subcodes wieder codiert.

OUTPUT_FIFO Die von den Transmit-FIFOs kommenden Daten werden mit den Subcodes codiert und ausgegeben.

Hierfür wird die Systemrate verwendet.

Sobald durch mindestens einem dieser beiden Tags der Zustand wirklich geändert wird, wird der DAT-Delay ausgelöst. Ausnahme hiervon ist ein Wechsel zwischen OUTPUT_INPUT und OUTPUT_FIFO.

1.23 Maestix: Status-Abfrage

Die Funktion `GetStatus()` ermittelt den aktuellen Status der MaestroPro. Ihr wird ein Status-Code übergeben, der angibt, aus welchem Bereich der Status ermittelt werden soll.

Es existieren die folgenden Codes:

`MSTAT_TFIFO` geben den Zustand der Transmitter/Receiver-FIFOs

`MSTAT_RFIFO` wieder. Das Ergebnis ist einer der folgenden Status-Codes:

`FIFO_Off` Die FIFO ist abgeschaltet. Es werden keine Daten gesendet bzw. empfangen.

`FIFO_Running` Die FIFO ist aktiviert. Es traten keine Fehler auf.

`FIFO_Error` Seit der letzten Abfrage des Status ist ein Fehler aufgetreten, so daß die Übertragung unterbrochen wurde.

`MSTAT_Signal` prüft, ob an dem gewünschten Eingang ein Signal anliegt. Das Ergebnis ist vom Typ `BOOL`. `INPUT_SRC48K` liefert immer ein Signal.

`MSTAT_Emphasis` gibt an, ob das Eingangssignal eine Emphasis verwendet. Das Ergebnis ist vom Typ `BOOL`. Liegt an dem gewählten Eingang kein Signal an, wird die interne Taktquelle verwendet. Diese hat keine Emphasis.

`MSTAT_DATsrc` meldet, ob das Eingangssignal von einem DAT- bzw. DCC-Recorder stammt. Das Ergebnis ist vom Typ `BOOL`. Ist an dem gewählten Eingang kein Signal vorhanden, wird die interne Taktquelle verwendet, welche eine DAT-Quelle simuliert.

`MSTAT_CopyProh` zeigt, ob das Eingangssignal einen Urheberrechtsschutz beantragt. Das Ergebnis ist vom Typ `BOOL`. Es ist nicht möglich, festzustellen, ob es sich beim Eingangssignal um ein Original oder eine Kopie handelt. Fehlt das Eingangssignal, wird die interne Quelle verwendet, welche keinen Kopierschutz beantragt.

`MSTAT_Rate` gibt die Verwendete Systemrate als `ULONG` zurück. Fehlt das Eingangssignal, wird die interne Quelle verwendet, welche konstant eine Rate von 48kHz liefert.

`MSTAT_UDB` liefert die aktuellen acht User Data Bits zurück. Dies ist zwar keine besonders brauchbare Lösung, aber vorläufig akzeptabel.

1.24 Maestix: Datenübertragung

Um Daten durch die Maestix übertragen zu können, müssen zunächst einmal Pufferspeicher im RAM angelegt werden (möglichst sogar im Fast-RAM). Diese Pufferspeicher müssen an einer durch 4 teilbaren Adresse liegen und ihre Längen müssen durch 1024 teilbar sein. Wichtig ist außerdem, daß der Puffer vom Speichertyp MEMF_PUBLIC ist, damit virtueller Speicher korrekt unterstützt wird.

Die Ein- und Ausgabe wird durch die Funktionen TransmitData() und ReceiveData() realisiert.

Dieser Funktion wird eine Message-Struktur übergeben, dessen Inhalt die Startadresse und Länge eines Pufferbereichs ist.

Alle Messages werden zunächst in eine Queue (Warteschlange) eingereiht. Sobald die Vorgänger-Message nun geleert bzw. gefüllt ist, wird die neue Message geholt und der damit angegebene Speicherbereich wird zur Datenübergabe verwendet. Die alte Message wird vorher zu dem in ihr angegebenen Reply-Port geschickt. Der Aufrufer kann den Speicherbereich nun "recyclen".

Die Queue wird im Interrupt geleert und ist deshalb abgesichert gegen Forbid(). Der Pufferbereich muß trotzdem groß genug sein, um auch Zeiträume zu überstehen, in denen das Multitasking abgeschaltet ist und daher keine neuen Puffer nachgeliefert werden können. Für einen stabilen Betrieb ist ein Puffer von 64kByte Länge schon so ziemlich die unterste Grenze; dieser Puffer ist immerhin in knapp einer Drittel Sekunde gefüllt! Vor allem, wenn ein langsamer Rechner benutzt wird, sollten noch weitaus größere Puffer verwendet werden!

Es ist außerdem ratsam, die komplette Bearbeitung der Messages in einen eigenen Prozeß zu legen und diesem eine Priorität zwischen 20 und 40 zu geben. So wird verhindert, daß durch langwierige Diskettenzugriffe oder bei blockierten Graphikzugriffen die Messages nicht mehr behandelt werden können. Es muß gewährleistet sein, daß der Prozeß immer rechtzeitig die verbrauchten Puffer besorgen und aktualisieren kann!

Der Weg über die Message erlaubt verschieden lange Pufferbereiche sowie ein einfaches Handling über die BetriebssystemFunktionen. Da der Task auf eine Nachricht warten kann, geht außerdem keine Zeit in Warteschleifen verloren.

Durch entsprechende Techniken ist es auch möglich, einen dynamischen Pufferbereich zu erzeugen, um zum Beispiel einen kurzfristig stark gebremsten Festplattenzugriff abzufangen.

Die Message-Struktur finden Sie in den Include-Dateien unter DataMessage-Struct.

1.25 Maestix: Realisation

Im Folgenden beschreibe ich die praktische Anwendung der MaestroPro-Funktionen.

Generell muß vorher ein ausreichend großer Pufferbereich belegt werden und auf ausreichend Messages aufgeteilt werden. Eine Faustregel dafür ist:

- zwei Puffer für die Eingabe
- zwei Puffer für die Ausgabe
- einen Puffer zum Berechnen
- (besser auch einen Puffer als Reserve)

Wenn Sie nur Daten einlesen möchten, reichen drei Messages also aus; für Echtzeitverarbeitung sollten aber schon fünf oder sechs Messages zur Verfügung stehen!

Die Messages müssen außerdem einen Zeiger auf einen gültigen Reply-Port enthalten, an dem die ausgeführten Messages an den Klienten zurückgesendet werden sollen. Die Karte muß belegt und der Ein- und Ausgang entsprechend eingestellt worden sein.

Wie bereits erwähnt, sollte der Pufferbereich mindestens 24KByte je Message betragen, damit ein reibungsloser Betrieb gewährleistet wird.

Um eine höhere Sicherheit zu gewähren, sollte der Pufferbereich in der Praxis sogar mindestens doppelt so groß sein!

Lediglich die Echtzeiteffekte benötigen keinen Pufferspeicher, da die Daten direkt umgewandelt werden.

1.26 Maestix: Senden

Das Senden geschieht durch einen Aufruf von TransmitData().

Dieser Funktion übergeben Sie einen Zeiger auf die MaestroBase sowie einen Zeiger auf die abzusendende Message. Die Funktion reiht nun die Message automatisch in die Sende-Queue ein. Wenn der Sendevorgang noch nicht gestartet wurde (oder durch einen Überlauf wieder gestoppt wurde), wird er außerdem gestartet.

Die Messages werden nun übertragen und anschließend an den angegebenen Message-Port zurückgesendet. Sie können nun neu gefüllt wieder durch TransmitData() abgesendet werden.

Verwenden Sie FlushTransmit(), wenn der Sendevorgang beendet werden soll. Diese Funktion unterbricht die Sendung und sendet alle bis dahin an die Maestix gesendeten Nachrichten zurück.

Ein FIFO_Running von GetStatus() gibt Ihnen Aufschluß darüber, ob der Sendevorgang läuft. Wenn während einer Übertragung FIFO_Error auftritt, waren nicht genügend Messages da und ein Unterlauf trat ein. Beachten Sie, daß in einem solchen Fall der Sendevorgang gestoppt wurde.

Es ist sinnvoll, während der Übertragung gelegentlich MSTAT_Signal auf TRUE zu überprüfen; besonders, wenn nicht die interne 48kHz-Taktquelle verwendet wird. Wenn die Synchronisation zwischenzeitlich ausfällt, sind die FIFOs möglicherweise nicht mehr richtig synchronisiert. Sie sollten die Übertragung dann abbrechen.

1.27 Maestix: Bitmanipulation

Eine reine Manipulation der UDBs und CSBs ist mit der MaestroPro ohne zusätzlichen Rechenaufwand möglich.

Hier müssen zunächst einmal die gewünschten CSBs und UDBs eingestellt werden.

Danach braucht nur noch der gewünschte Eingang eingestellt werden und als Ausgang OUTPUT_INPUT angegeben werden.

Das Eingangssignal wird dann decodiert und mit den neuen Subcodes direkt wieder codiert.

1.28 Maestix: Empfangen

Der Empfang von Daten geschieht entsprechend dem Senden mit der ReceiveData()-Funktion. Auch diese startet den Empfangsvorgang, wenn er noch nicht gestartet wurde oder durch einen Überlauf wieder gestoppt wurde. Ein Aufruf von StartReceive() ist nicht erlaubt.

Verwenden Sie FlushReceive(), wenn der Empfangsvorgang beendet werden soll. Diese Funktion unterbricht den Empfang und sendet alle bis dahin an die Maestix gesendeten Nachrichten zurück.

Ein FIFO_Running von GetStatus() gibt Ihnen Auskunft, ob der Empfangsvorgang läuft. Wird während einer Übertragung FIFO_Error übergeben, waren nicht genügend Messages da und ein Überlauf trat ein. Beachten Sie, daß in einem solchen Fall der Empfangsvorgang gestoppt wurde.

Es ist sinnvoll, auch während der Übertragung gelegentlich MSTAT_Signal auf TRUE zu überprüfen. Wenn die Synchronisation zwischenzeitlich ausfällt, sind die FIFOs möglicherweise nicht mehr richtig synchronisiert. Sie sollten die Übertragung dann abbrechen.

1.29 Maestix: Echtzeitverarbeitung

Wenn der Sende- und Empfangsvorgang kombiniert wird, kann auch eine Echtzeitverarbeitung mit der MaestroPro realisiert werden.

Jede Nachricht, die der Klient von dem Empfänger zurückerhält, sollte bearbeitet und anschließend an den Sender geschickt werden.

Nachrichten vom Sender können direkt wieder an den Empfänger gesendet werden. Gleichzeitig können die empfangenen Daten weiterverarbeitet werden.

1.30 Maestix: Echtzeiteffekte

Seit V38 ist die Maestix in der Lage, Echtzeiteffekte zu koordinieren.

Dazu stehen die Funktionen StartRealtime() und StopRealtime() zur Verfügung.

Bevor mit der Echtzeitverarbeitung begonnen werden kann, müssen bestimmte Voraussetzungen geschaffen werden. Die Karte muß natürlich erst einmal belegt werden; sämtliche Sende- und Empfangsvorgänge müssen beendet sein! Mittels SetMaestro() wird die MaestroPro dann auf folgende Parameter eingestellt:

- MTAG_Input muß auf die Datenquelle eingestellt werden.

Normalerweise ist dies INPUT_STD. INPUT_SRC48K ist nicht sinnvoll!

- MTAG_Output muß auf OUTPUT_FIFO eingestellt werden. Jeder andere Wert ist nicht sinnvoll.

- Die anderen Tags können wie gewohnt eingestellt werden. Ich empfehle hier, jeweils die xxx_INPUT-Variante zu benutzen.

StartRealtime() wird anschließend ein Zeiger auf eine TAG-Liste sowie ein Zeiger auf die MaestroBase übergeben.

In der Tagliste können folgende Werte übergeben werden:

- MTAG_Effect wählt den Effekt-Typ aus. (siehe unten)

- MTAG_A0 \

- MTAG_A1 | sind Parameter für die Effekte

- MTAG_D2 |

- MTAG_D3 /

- MTAG_CustomCall übergibt einen Zeiger auf eine eigene Effekt-Routine (mehr siehe unten).

Die Effektbearbeitung beginnt sofort und endet mit StopRealtime().

Beachten Sie, daß die Echtzeiteffekte beendet werden müssen, wenn Sie den normalen Sende-Empfangs-Modus verwenden möchten.

Folgende Effekte (MTAG_Effect) stehen zur Verfügung:

RFX_Muting Stummschaltung

RFX_Bypass kein Effekt

RFX_ChannelSwap Links <-> Rechts

RFX_LeftOnly nur Links

RFX_RightOnly nur Rechts

RFX_Mono Mono-Signal

RFX_Surround Hintergrundkanal

RFX_Volume Lautstärkeregelung

RFX_Karaoke Gesangsunterdrückung

RFX_Foregnd Vordergrundkanal

RFX_Spatial Basisbreite

RFX_Echo Echo/Hall

RFX_Mask Maskieren/Quantisieren

RFX_Offset Gleichspannungsoffset

Durch MTAG_CustomCall können außerdem eigene Effekte erstellt werden.

Die Routine für den Effekt muß im PUBLIC Memory-Pool stehen, beachten

Sie das unbedingt! Ein normal gestartetes Programm steht nicht

unbedingt automatisch im PUBLIC Memory-Pool! Die Routine sollte außerdem in hochoptimiertem Assemblercode verfaßt sein.

Die Routine bekommt während der Laufzeit folgende Parameter übergeben:

-> D0 (WORD) Datenwort vom linken Kanal

-> D1 (WORD) Datenwort vom rechten Kanal

-> D2 (ULONG) Parameter vom MTAG_D2-Tag

-> D3 (ULONG) Parameter vom MTAG_D3-Tag

-> D6 (ULONG) Scratch-Register

-> D7 (ULONG) Scratch-Register

-> A0 (ULONG) Parameter vom MTAG_A0-Tag

-> A1 (ULONG) Parameter vom MTAG_A1-Tag

-> A2 (APTR) Rücksprungadresse

Das Ergebnis muß in folgenden Registern stehen:

<- D0 (WORD) Datenwort für den linken Kanal

<- D1 (WORD) Datenwort für den rechten Kanal

Die Register D2,D3,D4,D5,A0,A1,A2,A3,A4,A5,A6,A7 dürfen NICHT verändert werden!

Wenn die Routine beendet ist, muß mit einem JMP (A2) zurückgesprungen werden. Ein RTS ist nicht zulässig.

BEISPIEL:

```
----->8-----  
Effect: exg d0,d1 ;tausche Links und Rechts  
asr #1,d1 ;Rechts halbe Lautstärke  
jmp (a2) ;und Ende...  
-----8<-----
```

1.31 Maestix: Muting-Effekt

TAG-Wert: RFX_Muting

Beschreibung: Es werden beide Kanäle leisegetastet.

Parameter: Keine.

Formel: $L := 0$

$R := 0$

1.32 Maestix: Bypass-Effekt

TAG-Wert: RFX_Bypass

Beschreibung: Das Signal wird nicht verändert.

Parameter: Keine.

Formel: $L := L$

$R := R$

1.33 Maestix: ChannelSwap-Effekt

TAG-Wert: RFX_ChannelSwap

Beschreibung: Der linke und rechte Kanal werden vertauscht.

Parameter: Keine.

Formel: $L := R$

$R := L$

1.34 Maestix: LeftOnly-Effekt

TAG-Wert: RFX_LeftOnly

Beschreibung: Es wird nur der linke Kanal durchgeschaltet. Der rechte wird stummgetastet.

Parameter: Keine.

Formel: $L := L$

$R := 0$

1.35 Maestix: RightOnly-Effekt

TAG-Wert: RFX_RightOnly

Beschreibung: Es wird nur der rechte Kanal durchgeschaltet. Der linke wird stummgetastet.

Parameter: Keine.

Formel: $L := 0$

$R := R$

1.36 Maestix: Mono-Effekt

TAG-Wert: RFX_Mono

Beschreibung: Durch Addition der Wellen wird das Stereosignal zu einem Monosignal zusammengefaßt.

Parameter: Keine.

Formel: $L := (L+R)/2$

$R := (L+R)/2$

1.37 Maestix: Surround-Effekt

TAG-Wert: RFX_Surround

Beschreibung: Aus dem eingehenden Signal wird der Surround-Hintergrundkanal dekodiert.

Parameter: Keine.

Formel: $L := (L-R)/2$

$R := (L-R)/2$

1.38 Maestix: Volume-Effekt

TAG-Wert: RFX_Volume

Beschreibung: Die Lautstärke beider Kanäle wird getrennt verändert.

Parameter: MTAG_D2 (UWORD) gibt die linke Lautstärke an.

MTAG_D3 (UWORD) gibt die rechte Lautstärke an.

Lautstärke 0 -> Stummschaltung

:

256 -> maximale Lautstärke

:

512 -> doppelte Lautstärke (Verzerrungen!)

Beachte: Ein 68020 sollte mindestens verwendet werden!

Formel: $L := (L*D2)/256$

$R := (R*D3)/256$

1.39 Maestix: Karaoke-Effekt

TAG-Wert: RFX_Karaoke

Beschreibung: Aus dem eingehenden Signal wird ein Musiksignal berechnet, das in der Regel keine Gesangsstimme mehr hat.

Der Effekt macht sich zunutze, daß die Gesangsstimme normalerweise immer genau in der Mitte ist.

Parameter: Keine.

Formel: $L := L - (L+R)/2$

$R := R - (L+R)/2$

1.40 Maestix: Foreground-Effekt

TAG-Wert: RFX_Foregnd

Beschreibung: Hier wird aus dem eingehenden Signal die Surround-Hintergrundkanal-Information herausgefiltert.

Parameter: Keine.

Formel: $L := L - (L-R)/2$

$R := R - (L-R)/2$

1.41 Maestix: Spatial-Effekt

TAG-Wert: RFX_Spatial

Beschreibung: Es wird ein Basisbreiten-Effekt gewonnen. Die Lautsprecher scheinen weiter auseinander zu stehen.

Parameter: MTAG_D2 (UWORD) gibt den Basisbreiten-Faktor an.

Basisbreite 0 -> normales Stereosignal

:

64 -> guter Basisbreiten-Wert

:

256 -> normales Mono-Signal

Beachte: Ein 68020 sollte mindestens verwendet werden!

Formel: $L := (L + (R*D2)/256) / 2$

$R := (R + (L*D2)/256) / 2$

1.42 Maestix: Echo-Effekt

TAG-Wert: RFX_Echo

Beschreibung: Hierdurch wird ein Echo erzeugt. Vorher muß die MRTorus-Struktur ausgefüllt werden, um einen Ringpuffer zu übergeben.

Parameter: MTAG_D2 (UWORD) gibt die Einstiegslautstärke eines Samples in den Ringpuffer an.

MTAG_D3 (UWORD) gibt die Echolautstärke eines Samples in dem Ringpuffer an. Durch die Formel $256/(512-D3)$ läßt sich berechnen, wie oft das Sample wiederholt wird.

MTAG_A0 (APTR) Zeiger auf die MRTorus-Struktur. Dort wird für jeden Kanal je ein Pufferspeicher und die Größe des Puffers übergeben. Die Größe muß stets durch Zwei teilbar sein!

Einstiegsst. 0 -> kein Echo

:

64 -> Einstieg mit halber Lautstärke

:

256 -> Einstieg mit voller Lautstärke

:

512 -> Einstieg mit doppelter Lautstärke

Echolautst. 0 -> ein Echo, kein Nachhall

:

64 -> leicher Hall

:

256 -> zweifacher Hall

:

512 -> dauernder Nachhall

Beachte: Ein 68030(!) sollte mindestens verwendet werden!

Formel: $Lout := (Lin + RLout) / 2$

$RLin := ((Lin * D2) / 256 + (RLout * D3) / 256) / 2$

$Rout := (Rin + RRout) / 2$

$RRin := ((Rin * D2) / 256 + (RRout * D3) / 256) / 2$

Erhöhen des Ringpuffer-Zeigers

(Lout -> linkes Ausgangssignal [Rout],

Lin -> linkes Eingangssignal [Rin],

RLout -> linker Ringpuffer-Ausgang [RRout],

RLin -> linker Ringpuffer-Eingang [RRin]).

1.43 Maestix: Mask-Effekt

TAG-Wert: RFX_Mask

Beschreibung: Bei diesem Effekt wird das Signal durch eine Bitmaske mit dem logischen Und maskiert. Durch entsprechende Masken kann eine Quantisierung auf verschiedene Auflösungen erreicht werden (z.B. 0xFF00 für 8 Bit-Quantisierung).

Parameter: MTAG_D2 (UWORD) linke Bitmaske

MTAG_D3 (UWORD) rechte Bitmaske

Beachte: Vorsicht ist geboten, da durch falsche Einstellungen schnell die Lautsprecher zerstört werden können! Experimente sollten stets mit geringer Lautstärke durchgeführt werden!

Formel: $L := L \& D2$

$R := R \& D3$

1.44 Maestix: Offset-Effekt

TAG-Wert: RFX_Offset

Beschreibung: Auf das Eingangssignal wird eine Gleichspannung addiert. So können Gleichspannungsfehler von schlechten Samplern gezielt ausgeglichen werden.

Parameter: MTAG_D2 (UWORD) linker Offset (-32767..32768)

MTAG_D3 (UWORD) rechter Offset (-32767..32768)

Formel: $L := L + D2$ mit Überlaufbehandlung

$R := R + D3$ mit Überlaufbehandlung

1.45 Maestix: Demo-Sources

Zum Maestix-Paket gehören einige Demonstrations-Programme sowie ihre Sources. Sie können (und sollten) diese Beispiele studieren, um zu sehen, wie die `maestix.library` in der Praxis angewendet wird.

Ich muß allerdings (zu meiner Entschuldigung ;-) erwähnen, daß die Programme nicht gerade gut programmiert sind. Sie würden sonst nur unnötig aufgebläht werden und das wesentliche verdecken.

1.46 Maestix: AnalyzeInput

Dieses Programm zeigt an, ob ein Signal am Standard-Eingang vorliegt, und dekodiert es.

Zunächst wird die MaestroPro belegt. Anschließend wird der Standard-Eingang gewählt.

Die folgenden GetStatus()-Aufrufe ermitteln nun die Daten, die in den Channel Status-Bits des Eingangssignals übertragen wurden. Die einzelnen Ergebnisse werden analysiert und der Ausgabertext entsprechend gesetzt.

Anschließend wird die Karte wieder zurückgesetzt und dann freigegeben.

1.47 Maestix: CSBchanger

In diesem Beispielprogramm werden die Subcodes in Echtzeit verändert.

Es wird zunächst auf den Standard-Eingang umgeschaltet. Der AusgangsModus ist OUTPUT_Input, und es werden die neuen CSBs angegeben.

Während das Programm nun lediglich auf das Schließen des Fensters wartet, ändert die MaestroPro selbstständig die CSBs um.

1.48 Maestix: LevelWindow

Dieses bereits etwas komplexere Programm öffnet ein kleines Fenster, in dem die Aussteuerung des Eingangssignals angezeigt wird.

Hierfür wird ein zweiter Prozeß verwendet, der die MaestroPro belegt, drei Messages erzeugt und anschließend die Übertragung startet. Von den zurückgegangenen Nachrichten wird der Pegel berechnet und in zwei Variablen geschrieben. Ein Signalbit signalisiert dem Hauptprogramm, daß die Anzeige aufgefrischt werden sollte. Danach wird die Message wieder an die maestix.library gesendet.

Das Hauptprogramm überprüft in einer Schleife, ob das Fenster geschlossen wurde oder ob es refreshed werden muß.

Wenn das Programm beendet wurde, wird ein Signal an den Level-Prozeß geschickt, welcher die Arbeit abbricht, die MaestroPro freigibt und dem Hauptprogramm dann signalisiert, daß er nun beendet ist.

Das Hauptprogramm gibt anschließend alle Ressourcen frei.

1.49 Maestix: Realtime

Dieser Source verdeutlicht die Realtime-Effekt-Funktion der Maestix.

Zuerst wird die Karte belegt und entsprechend eingestellt.

Anschließend erfolgt der Aufruf von `StartRealtime()`, welcher durch die TAG-Liste den Spatial-Effekt startet. Durch Austausch der Parameter in der TAG-Liste können aber auch die anderen Effekte getestet werden.

Wenn das Steuerfenster dann geschlossen wird, wird der Echtzeiteffekt gestoppt und die MaestroPro freigegeben.

1.50 Maestix: RealtimeEcho

Dieser Source verdeutlicht die Realtime-Effekt-Funktion der Maestix durch ein Echo.

Zuerst wird die Karte belegt und entsprechend eingestellt.

Anschließend erfolgt der Aufruf von `StartRealtime()`, welcher durch die TAG-Liste den Echo-Effekt startet. Die Echo-Parameter werden in den entsprechenden Equates am Anfang des Programms eingestellt:

`BUFFMS` stellt die Echolänge in Millisekundengenauigkeit ein (bezogen auf 44100 Hz Samplingrate).

`VOLUME` ist die Lautstärke des ersten Echos (0..256).

`DECAY` ist der Lautstärkeverfall jedes weiteren Echos zum vorhergehenden (0..256).

Wenn das Steuerfenster dann geschlossen wird, wird der Echtzeiteffekt gestoppt und die MaestroPro freigegeben.

1.51 Maestix: SineTone

In diesem Programm wird eine Ausgabe realisiert.

Es werden Puffer belegt und mit einer Sinus-Schwingung gefüllt, welche nach 32 Worten eine komplette Schwingung durchgeführt hat.

Durch einen Prozeß (ähnlich dem LevelWindow-Prozeß) wird dieser Puffer nun wiederholt ausgegeben. Als Taktbasis dient der interne 48kHz-Taktgenerator, was als Resultat einen 1,5 kHz-Sinuston am Ausgang ergibt.

Wenn das Fenster geschlossen wird, wird die Übertragung abgebrochen und die Puffer freigegeben.

1.52 Maestix: SineTone2

Dieses Programm unterscheidet sich prinzipiell nicht von SineTone.

Der einzige Unterschied ist hier, daß das Eingangssignal als Taktquelle verwendet wird.

Dementsprechend liegt bei einem Eingangstakt von 48kHz am Ausgang ein 1,5kHz-Sinuston an, während bei 44,1kHz ein 1,378kHz-Sinuston und bei 32kHz ein 1kHz-Sinuston am Ausgang anliegt.

1.53 Maestix: Surround

In diesem Beispiel wurde die Ein- und Ausgabe kombiniert.

Es werden zunächst fünf Puffer bereitgestellt, von denen die Hälfte an den Transmitter und der Rest an den Receiver gesendet werden.

Von dem eingehenden Signal wird die Differenz des linken Kanals (1. Wort) und des rechten Kanals (2. Wort) gebildet und zurückgeschrieben.

Dadurch entsteht ein surroundartiger Effekt.

Der veränderte Puffer wird nun an den Transmitter gesendet. Wenn die Übertragung des Puffer dann abgeschlossen ist, wird sie direkt wieder dem Receiver zugeführt.

Wenn das Fenster geschlossen wurde, wird die Ein- und Ausgabe gestoppt und die Puffer freigegeben.

1.54 Maestix: Credits

Ich möchte mich bei folgenden Leuten bedanken:

Sven Arke daß er mich mit seiner Mailbox unterstützt.

Andreas Benden für's Beta-Testen

Thomas Wenzel für das erste Tool, welches die maestix.library verwendet.

Frank Wille für den besten Assembler, der momentan zu bekommen ist!

Dietmar Eilert für seinen hervorragenden Editor GoldEd.

Elrad für die Beschreibung des Subcode-Formats und der Spezialchips.

MacroSystem für die Entwicklung der Maestro Professional.

ESCOM dafür, daß sie sich so für unsere
Freundin eingesetzt haben (und
einsetzen werden?!?) THANXX!!!

Commodore Amiga für die Entwicklung des besten
Computers!

Ohne sie wäre die Maestix-Library niemals entstanden oder entwickelt
worden.

//

\\// -- Amiga - Jetzt erst recht! --

\\X/
