Maestix

Richard Körber

COLLABORATORS			
	TITLE : Maestix		
ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY	Richard Körber	August 22, 2024	

REVISION HISTORY			
NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Maestix

1.1 Maestix: Inhaltsverzeichnis

TriTech Developments Proudly Presents

__ _____ /

/\/\ /\ /_ (_ | · \/

/ X--____) | | /\ V41

/ / /

-- INHALTSVERZEICHNIS --

1.2 Maestix: English Introduction

Hi!
Sorry, but I had not 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. 5 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.
· If you use Maestix in a commerical product (ShareWare, CrippleWare, full commercial Software), you must register me for free.
Commercial Software must include the whole Maestix packet!
If you have any questions or comments, or need further support, feel free to contact me:
shred@chessy.aworld.de
Or visit my home page:
http://www.is-koeln.de/einwohner/shred/
Have fun!

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 war allerdings auf die kleine Handvoll Programme beschränkt, die es für die MaestroPro gab. Im Gegensatz zur Toccata-Soundkarte gab es für die MaestroPro nämlich noch keine Library, mit der man selbst Programme für diese Karte schreiben konnte.
Ich wollte diesem Mißstand ein Ende bereiten und diese Library programmieren. Leider war die Unterstützung seitens MacroSystem zu diesem Projekt kaum der Rede wert. Hier wurde mir klar, daß ich mich selbst "ans Werk" machen mußte...
Als erstes besorgte ich mir die Unterlagen über die zwei Yamaha-Chips 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. Durch die Entwicklung über Jahre hinweg ist schließlich ein zuverlässiger Treiber für die Karte entstanden, der in vielen Programmen benutzt wird.

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 (full duplex)
· 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
· 15 integrierte Echtzeiteffekte, eigene Echtzeiteffekte möglich
· unterstützt alle wesentlichen Features der MaestroPro Soundkarte
· 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 (bekannten ;^) Enforcer- und Mungwall-Hits
· Unterstützt alle Sprachen. C und Assemblerincludes sind inklusiv!
· 100% System-Konform, keine Hardware-Hacks
· auf 68040, 68060 und CyberPPC getestet

1.5 Maestix: Neuigkeiten

Das Tool MaestroPEG ist hinzugekommen. Es spielt MPEG-Audiodateien über die mpega.library von Stéphane Tavenard ab.

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

COPYRIGHT

==========

ANMERKUNG: Sie akzeptieren die folgenden Bedingungen durch
den Start der Software, selbst wenn es nur zur Probe ist.
Maestix © 1995-98 Richard Körber, all rights reserved.
Dieses Archiv ist urheberrechtlich geschützt. Der Urheber ist Richard
Körber.
Sie haben nur das Recht, diese Software zu benutzen, aber keine Rechte
an der Software an sich. Disassemblieren, Resourcen und alle anderen
Arten des Reverse-Engineering sind verboten.
Die mpega.library ist (C) Stéphane Tavenard.
FREEWARE

Das Maestix-Paket ist FreeWare. Ihre Benutzung ist kostenlos.
VERVIELFÄLTIGUNG

Sie können dieses Paket so lange kopieren, wie es vollständig und
unverändert bleibt.
Das Paket darf mit den üblichen Kompressionsprogrammen (z.B. lha, lzx,
lzh, dms) komprimiert werden. Einzelne Dateien dürfen jedoch nicht
komprimiert werden (z.B. mit PowerPacker, Imploder).
VERTEILUNG

Sie dürfen nicht einen marktüblichen Preis für Arbeit und Material
überschreiten. Für Disketten gilt eine Grenze von 5 DM; für CD-ROMs,
die auch weitere PD-Software enthalten, 35 DM.
Ein Vertrieb auf Coverdisks oder zusammen mit kommerzieller Software
bedarf meiner schriftlichen Einverständniserklärung.
Ich gestatte ausdrücklich den Vertrieb über das AmiNet, Meeting Pearls
und andere bekannte PD-Serien.
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. Dies gilt auch für alle Updates.
- Kommerzieller Software ist außerdem das komplette Maestix-Paket hinzuzufügen.
- Bitte erwähnen Sie in der Anleitung, daß die maestix.library von Richard Körber verwendet wird und daß die Benutzung auf eigene Gefahr geschieht.

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

HAFTUNG

Sie verwenden das Programm auf eigenes Risiko!
In keinem Fall haftet der Autor für Schäden und Folgeschäden, die auf den Gebrauch dieses Programms zurückzuführen sind, sofern kein Vorsatz nachgewiesen werden konnte.
Mithilfe der MaestroPro ist es möglich, den SCMS-Kopierschutz zu umgehen. Bitte beachten Sie hier unbedingt die Urheberrechtsbestimmungen ihres Landes!

NUTZUNGSEINSCHRÄNKUNG

Sie dürfen dieses Programm nicht verwenden

· für faschistische oder militärische Zwecke
· wenn Sie mit dieser Erklärung nicht einverstanden sind

In diesem Fall müssen Sie das Archiv umgehend löschen.

BESTANDTEILE DES PAKETS

Das Paket ist nur vollständig mit folgenden Dateien:

maestix/c/AllocMstx
maestix/c/FreeMstx
maestix/c/MaestroPEG
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/clib/maestix_protos.h
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.h
maestix/includes/maestix_lib.i
maestix/libs/maestix.library
maestix/maestix.doc
maestix/maestix.doc.info
maestix/maestix.dok
maestix/maestix.dok.info
maestix/Maestix.guide
maestix/Maestix.guide.info
maestix/MaestixFX
maestix/MaestixFX.info
maestix.info
FILE_ID.DIZ
Install-D
Install-D.info
Install-E
Install-E.info

1.8 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!). Ich möchte
Sie in dem Zusammenhang bitten, bei Fragen über die maestix.library
auch nicht die MacroSystem-Hotline in Anspruch zu nehmen.

1.9 Maestix: Kontaktadresse

Wenn Sie mich anschreiben möchten, verwenden Sie bitte eine meiner
E-Mail-Adressen:
shred@eratosthenes.starfleet.de
shred@chessy.aworld.de
richard.koerber@koeln.netsurf.de
Oder besuchen Sie meine Homepage:
http://www.is-koeln.de/einwohner/shred/
http://www.shredzone.home.pages.de
Ebenso stehen in der offiziellen Support-Mailbox updates bereit, und
Probleme können diskutiert werden.
Name: Eratosthenes
Nummer: 0228-239522 (V.34, ISDN)
Login: SUPPORT (kein Password)
Brett: /SUPPORT/SHRED

1.10 Maestix: Bugs

Geschrieben wurde Maestix auf einem:
Amiga 4000/o6o CyberStorm MKII mit AmigaOS 3.0, 44MB RAM, CyberVision 4MB,
CyberSCSI (4GB HD, CD-ROM, ZIP-Drive), MultiFace II, Toccata, MaestroPro.
Getestet wird Maestix ständig auf folgenden Rechnern:
· A4000/o6o (MMU,FPU)
Kick 3.0 ,44 MB RAM (42F/2C), 4GB HD,
MaestroPro, CyberVision (4MB), Toccata
· A3000/o6o/604 (PPC,MMU,FPU)
Kick 3.1 ,34 MB RAM (32F/2C), 4GB HD,
MaestroPro, CyberVision (4MB)

Folgende Bugs sind bekannt:
· Eine Datenübertragung durch die maestix.library stört die Datenübertragung auf der internen seriellen Schnittstelle mit hohen Übertragungsraten erheblich. Ich rate daher ab, gleichzeitig zur Datenübertragung die interne serielle Schnittstelle zu benutzen! Mit FIFO-Karten (z.B. MultiFace II) und schnellen Prozessoren treten jedoch keine Probleme auf.
· Die aktiven Demoprogramme hängen sich auf, wenn die Karte bereits belegt wurde. Das Problem ist die Beendung des zweiten Prozesses, da in diesem Fall ein Deadlock entsteht. Es ist kein Fehler der maestix library, sondern lediglich meine Faulheit in dem Demoprogramm. :)
Folgende Enforcer-, Mungwall- oder PatchWork-Hits sind bekannt:
· keine bekannt :-)

1.11 Maestix: History

V41.30 · Dual-Modus eingebaut (marginale Änderung in der ExtDataMessage-Struktur)
· MaestroPEG geschrieben
V41.20 · Bugfix: AES/EBU-Rate wurde stets auf 48kHz Manual gesetzt
· Semaphoren verwendet -> besseres Multitasking-Verhalten
· 68020+ optimiert, 68000-Version auf Anfrage!
· Es gibt keine "alten" MaestroPro's, sondern es gab nur eine falsch bestückte. Der Workaround von V37.10 wurde deshalb wieder entfernt, und eine Belegung über nicht-maestix-Programme wieder erkannt.
V41.10 · Saubererer Reset des Encoderchips
· Bugfix: bei Realtime-Effekten wurden die Eingangskanäle gelegentlich vertauscht
V41.00 · ReSample-Effekt implementiert
· LevelMeter in Realtime-Pfad eingebaut
· Foregnd-Ausgabequalität verbessert
· Robot erlaubt jetzt auch Open/Close von 0
· Zugriff auf Execbase jetzt nicht mehr über Chip-RAM
V40.20 · 68040/060-Timingproblem gelöst, dort waren die übertragenen Kanaldatenbits gelegentlich fehlerhaft.
· Studiomodus kann jetzt als Default eingeschaltet werden.
· SetMstx und Installer-Script dementsprechend überarbeitet.
V40.10 · Es können jetzt auch Puffer <1024 Bytes verwendet werden

· extended audio data message eingeführt
· Übergebene Puffer können jetzt auch Mono sein
· Der Robot-Effekt funktionierte nicht
· Kleine 68060-relevante Verbesserungen
· Bug: keine Soundausgabe, wenn erst nach TransmitData()
auf OUTPUT_FIFO umgeschaltet wurde
V40.00 · Kleine Aufräumarbeiten
· CopyBack-Routinen waren überflüssig und wurden entfernt
· Robot-Echtzeiteffekt
· Maestix fällt jetzt automatisch auf den internen Raten-
generator (SRC48K) zurück, wenn bei TransmitData() kein
gültiges Signal am Eingang anliegt
V39.30 · FlushTransmit() und FlushReceive() zerstörten das A4-
Register [Teijo Kinnunen]
V39.20 · Das Knarren und Fiepen bei Beendigung/Abbruch einer Daten-
übertragung wird jetzt unterdrückt.
· FreeMstx gibt jetzt nur noch eine Belegung der Soundkarte
durch AllocMstx frei.
· maestix.library ist vorbereitet auf die neue MacroSystem-
Soundlibrary.
V39.15 · Einige kleine Verbesserungen
V39.10 · Bug: TransmitData() lieferte nach dem Start direkt einen
FIFO-Error [Thomas Wenzel, Daniel Wicke]
· Das DAT-Delay-Kapitel fehlte
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-Re-
corder 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 Copyprotecion-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

1.12 Maestix: Zukunft

Das war’s dann... Die Library ist meines Erachtens komplett und
scheint mittlerweile auch bugfrei zu sein. Wenn jemand noch Ideen
hat, bitte melden!

1.13 Maestix: Installation

Zur Installation der maestix.library benötigen Sie mindestens einen
68020-Prozessor. Eine 68000-Version ist auf Anfrage erhältlich.
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 einge-
stellt, wenn es in die User-Startup eingefügt wird.
Die Installation ist dann abgeschlossen.

1.14 Maestix: Das DAT-Delay

Einige DAT-Recorder brauchen etwas Zeit, um sich an ein neues Signal
anzupassen. Die in dieser Zeit gesendeten Daten werden nicht
umgesetzt.
Die Library kümmert sich automatisch darum und wartet einen gewissen
Zeitraum, wenn sich das Ein- oder Ausgabesignal ändert. Dieser
Zeitraum läßt sich durch SetMstx auf den verwendeten DAT-Recorder
einstellen.
Der Sony DTC-690 benötigt zum Beispiel ca. eine Sekunde.

1.15 Maestix: Shell-Tools

Dem Maestix-Paket sind noch Shell-Tools beigefügt, welche ins C:-Ver-
zeichnis kopiert werden sollten. Diese Tools sind auch für die End-
anwender interessant!

1.16 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 einge-
stellten Zeitraum, damit das DAT-Gerät
sich auf die neue Rate einstellen kann.
Siehe Delay
NOSTUDIO Als Default wird der Studiomodus abge-
stellt. Dies ist die Voreinstellung und
ist für die meisten Heim-DAT-Recorder
mit S/P-DIF notwendig.
STUDIO Als Default wird der Studiomodus akti-
viert. Voraussetzung dafür ist ein DAT-
Recorder, der das Format AES/EBU erfor-
dert.
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 NOSTUDIO FORCE

1.17 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 Work: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 ----------

1.18 Maestix: MaestroPEG

MaestroPEG ist ein kleiner MPEG-Audio-Player, der die mpega.library von
Stéphane Tavenard verwendet. Da die dekodierten Audiodaten direkt zur
Soundkarte gelangen, wird Rechenzeit gespart. Eine Wiedergabe von MPEG1
Layer III-Samples in CD-Qualität sollte somit auch auf einen 68040/33
möglich sein. Aber richtig Spaß macht es erst auf einem 68060. ;-)
Momentan handelt es sich noch um eine Beta-Release.

FILE/A,Q=QUALITY/K/N,PRI=TASKPRI/K/N,INFO/S,NOTIME/S,NOPROH/S

Die Verwendung ist:

FILE/A Die abzuspielende Datei.

Q=QUALITY/K/N Wiedergabequalität, von 0 (niedrig) bis
2 (hoch). Voreingestellt ist 2.

PRI=TASKPRI/K/N Task-Priorität während der Wiedergabe.
Voreingestellt ist 20.

INFO/S Gibt Informationen über die MPEG-Datei aus.

NOTIME/S Keine Zeitanzeige während der Wiedergabe.

NOPROH/S Normalerweise werden die in der MPEG-Datei
eingetragenen Kopierschutzbits beachtet.
Falls dies bei Ihrem DA-Wandler zu Problemen
führt, können Sie den Kopierschutz mit die-
ser Option abschalten.

Die Wiedergabe kann jederzeit mit CTRL-C abgebrochen werden.

Die mpega.library finden Sie zum Beispiel im AmiNet
(util/libs/mpega_library.lha).

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 sowie Zusatzinformationen 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 das Kopierschutzbit. Hier wird zwischen Heim- und Studiomodus unterschieden. Die MaestroPro unterstützt auf der Senderseite beide Formate, der Empfänger kann allerdings nur einen Teil des Heim-Standards 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 wur-
den, 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
Kompatibelitätsgründen 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 Grundparame-
ter ein. Diese werden je nach Anwendung erst einmal entsprechend ein-
gestellt. 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 zurückge-
setzt.
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 di-
gitale Kopie angefertigt werden.
CPROH_PROHIBIT setzt den Kopierschutz. Eine
digitale Aufnahme des Aus-
gangssignals 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_PROHI-
BIT. Wenn kein Eingangssignal vor-
liegt, ist der Kopierschutz abge-
schaltet.
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 50μs-
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 ma-
nuell 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 Ein-
gangssignal, wird SRC_DAT ausge-
wählt.
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 verstell-
bar (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 Ein-
gangssignal. Fehlt das Ein-
gangssignal, 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 fol-
gende 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.
WICHTIG: Wenn Sie den Eingang wechseln, müssen alle
Tags, die als Parameter _INPUT hatten (CPROH_INPUT,
EMPH_INPUT, SRC_INPUT und RATE_INPUT), erneut übergeben werden, damit die Werte auf den neuen Eingang angepaßt werden.
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 Sta-
tus ist ein Fehler aufgetreten, so
daß die Übertragung unterbrochen
wurde.
Ich empfehle, vor dem ersten TransmitData() bzw.
ReceiveData() den Status der jeweiligen FIFOs abzufra-
gen, damit deren Zustand initialisiert wird.
MSTAT_Signal prüft, ob an dem gewünschten Eingang ein Singal
anliegt. Das Ergebnis ist vom Typ BOOL. INPUT_SRC48K
liefert immer ein Signal.
MSTAT_Emphasis gibt an, ob das Eingangssignal eine Emphasis verwen-
det. Das Ergebnis ist vom Typ BOOL. Liegt an dem ge-
wählten Eingang kein Signal an, wird die interne Takt-
quelle verwendet. Diese hat keine Emphasis.
MSTAT_DATsrc meldet, ob das Eingangssignal von einem DAT- bzw. DCC-
Recoder 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 Urheberrechtschutz
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 vor-
läufig 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. Seit der V40.10 ist ein Vielfaches von 1024 als
Pufferlänge nicht mehr erforderlich. Wenn möglich, sollte man sich

aber trotzdem daran halten, da die Verarbeitungsgeschwindigkeit dann optimal ist. 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. Vor dem ersten TransmitData() sollte der Status der FIFO gelesen werden, damit ein eventuell vorgemerkter Error-Zustand wieder gelöscht wird. Das Ergebnis der Statusabfrage ist nicht von Interesse.
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.
Vor der Übertragung des ersten Pakets sollte der Ausgang auf OUTPUT_FIFO umgestellt werden.

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. Vor dem ersten ReceiveData() sollte der Status der FIFO gelesen werden, damit ein eventuell vorgemerkter Error-Zustand wieder gelöscht wird. Das Ergebnis der Statusabfrage ist nicht von Interesse.
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

RFX_Robot Robot-Effekt

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 und sämtliche Adreßregister dürfen NICHT
verändert werden! D6 und D7 dienen als Scratch-Register.
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< ----------
GetStatus() gibt auch hier Auskunft über den Zustand. Es ist jedoch
egal, ob der Zustand des Transmitters oder des Receivers abgefragt
wird. Wird während einer Übertragung FIFO_Error übergeben, trat ein
Überlauf ein (z. B. wenn nicht genügend Rechenleistung für den Effekt
zur Verfügung stand). Beachten Sie, daß in einem solchen Fall der
Echtzeiteffekt gestoppt wurde. Vor dem Start sollte der Status der
FIFO gelesen werden, damit ein eventuell vorgemerkter Error-Zustand
wieder gelöscht wird. Das Ergebnis der Statusabfrage ist nicht von
Interesse.

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-Hin-
tergrundkanal-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!
Einstiegslst. 0 -> kein Echo
:
64 -> Einstieg mit viertel 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 Mas-
ken 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! Ex-
perimente sollten stets mit geringer Lautstärke durch-
geführt werden!
Formel: L := L & D2
R := R & D3

1.44 Maestix: Offset-Effekt

TAG-Wert: RFX_Offset
Beschreibung: Auf das Eingangssignal wird eine Gleichspannung auf-
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: Robot-Effekt

TAG-Wert: RFX_Robot
Beschreibung: Das Eingangssignal wird für eine bestimmte Zeitspanne
aufgeschaltet oder stummgeschaltet. Man erreicht damit
viele Effekte von knistern über blechern klingenden
Sound (wie bei einem Roboter) bis hin zu tremoloartigen
Effekten.
Parameter: MTAG_D2 (LONG) Anzahl der Samples, die aufgeschaltet
werden sollen (0..)
MTAG_D3 (LONG) Anzahl der Samples, die stummgeschaltet
werden sollen (0..)
MTAG_A0 (APTR) Zeiger auf eine initialisierte MRRobot-
Struktur.
Achtung: MTAG_D2 und MTAG_D3 dürfen nicht zusammen auf
0 stehen.
Beachte: Ein 68020 oder besser wird empfohlen.
Formel: L := (L solange MTAG_D2) | (0 solange MTAG_D3)
R := (R solange MTAG_D2) | (0 solange MTAG_D3)

1.46 Maestix: ReSample-Effekt

TAG-Wert: RFX_ReSample
Beschreibung: Das Eingangssignal wird mit einer scheinbar geringeren
Sampling-Rate wieder ausgegeben.
Parameter: MTAG_D2 (ULONG) Neue Sampling-Rate, links (1..Rate)
MTAG_D3 (ULONG) Neue Sampling-Rate, rechts (1..Rate)
MTAG_A0 (APTR) Zeiger auf eine initialisierte
MRReSample-Struktur.
Beachte: Ein 68020 oder besser wird empfohlen.
Formel: L := (Lold solange (Lcnt+MTAG_D2)<Lmax), sonst Lold=L, Lcnt-=Lmax
R := (Rold solange (Rcnt+MTAG_D3)<Rmax), sonst Rold=R, Rcnt-=Rmax

1.47 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.

1.48 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 Ausgabetext entsprechend gesetzt.
Anschließend wird die Karte wieder zurückgesetzt und dann freigegeben.

1.49 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.

1.50 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.

1.51 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.

1.52 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 vorhergehen-
den (0..256).
Wenn das Steuerfenster dann geschlossen wird, wird der Echtzeiteffekt
gestoppt und die MaestroPro freigegeben.

1.53 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.

1.54 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.

1.55 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.

1.56 Maestix: Häufige Fragen und Antworten

Bei MaestixFX erscheinen nur zwei Aussteuerungsanzeigen und ein paar Leuchtdioden. Ist das etwa alles?
Nein. Ziehen Sie das Fenster richtig groß, dann erscheint der Rest von der GUI. Das Programm sollte ansonsten selbsterklärend sein, jedenfalls bin ich zu faul, eine Anleitung zu schreiben.
Wo bekomme ich eine MaestroPro her?
Von dem Hersteller MacroSystem wird sie nicht mehr hergestellt und

auch nicht mehr vertrieben. Sie wird also nur noch gebraucht zu bekommen sein.
Solange der Vorrat reicht, ist ein kleiner Restposten der Karte (inklusive Samplitude, aber ohne optische Kabel) für ca. 300 DM zzgl. Porto und Verpackung (Nachnahme) erhältlich von:
C.H.S. Pommer
Schürbankstraße 18
44805 Bochum
Tel: 0234/866526
Fax: 0234/860854
(Preisänderungen und Irrtum vorbehalten.)
Ich habe noch eine alte Maestro (Produkt-ID 3). Kann ich auch diese noch verwenden, bzw. ist das geplant?
Leider nein. Die Maestro hat weder einen Decoderchip, noch FIFOs zur Pufferung der anfallenden Daten. Der Treiber wäre aufwendig zu schreiben, und müßte im Betrieb zwangsläufig das Multitasking abschalten. Die Verwendung wäre damit äußerst eingeschränkt, weshalb ich keinen Sinn darin sehe, dafür einen Treiber zu schreiben.
Mein Equipment erfordert den Studio-Modus (AES/EBU). Die meisten Programme lassen jedoch eine Wahl des Betriebsmodus nicht zu.
Geben Sie bei SetMstx die Option STUDIO an.
Wenn ich eine Wiedergabe starte, schaltet der DAT-Recorder um und "verschluckt" so die erste Sekunde.
Stellen Sie mit SetMstx einen entsprechenden Delay ein. Die meisten Programme warten so den entsprechenden Zeitraum, bevor sie die Wiedergabe beginnen.
Ich bekomme dauernd FIFO-Überläufe.
Ihr Prozessor ist zu langsam. Auch wenn die maestix.library hochoptimiert ist, braucht sie viel Zeit zum Schaufeln der Daten. Sie sollten sich wenigstens einen 68020 zulegen. MaestixFX benötigt sogar einen 68030 zum uneingeschränkten Arbeiten. MaestroPEG läuft mit optimaler Qualität erst ab einem 68040/50.

1.57 Maestix: Credits

Ich möchte mich bei folgenden Leuten bedanken:
Andreas Benden für’s Beta-Testen und für die Support-Box.
Thomas Wenzel für das erste Tool, welches die

maestix.library verwendet (Play16).

Teijo Kinnunen für OctaMed Sound Studio

Stéphane Tavenard für die mpega.library

Henning Friedl für seine freundliche Hilfe

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.

Ohne sie wäre die Maestix-Library niemals entstanden oder entwickelt

worden.

//

\\ // -- Amiga - jetzt erst recht --

\X/