Maestix

COLLABORATORS

	TITLE : Maestix	

ACTION	*NAME*	*DATE*	*SIGNATURE*
WRITTEN BY		July 22, 2024	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

Chapter 1

Maestix

1.1 Maestix: Inhaltsverzeichnis

TriTech Developments Proudly Presents

__ _____ /

/\/\ /\ /_ (_ | · \/

/ X--____) | | /\ V37

/ / /

-- INHALTSVERZEICHNIS --

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@eratosth.wwb.sub.de (internet)

Have fun!

1.3 Maestix: Einleitung

Die Maestro-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 Maestro gibt. Im Gegensatz zur Toccata-Soundkarte besitzt die Maestro 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 Inkompatibelitäten 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
· benötigt mindestens Kickstart-Version 2.04
· unterstützt S/P-DIF und AES/EBU (Rundfunkstudio-Modus)
· Senden und Empfangen der User Data Bits
· unterstützt alle mir bekannten Features der Maestro Soundkarte
· unterstützt auch ältere Versionen der Maestro
· nach außen hin hardwareunabhängiges Konzept, dadurch flexible Erweiterungsmöglichkeiten
· Keine Kenntnisse über die Maestro 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
· 100% System-Konform, keine Hardware-Hacks

1.5 Maestix: Neuigkeiten

In dieser Version habe ich die UDB-Abfrage implementiert sowie einen folgeschweren Fehler in den C-Includedateien berichtigt. Hoffentlich kann die Library jetzt auch in C benutzt werden...

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 Maestro 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/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/maestix.i
maestix/libs/maestix.library
maestix/Install-D
maestix/Install-D.info
maestix/Install-E
maestix/Install-E.info
maestix/maestix-d.adoc
maestix/maestix-d.adoc.info
maestix/maestix-e.adoc
maestix/maestix-e.adoc.info
maestix/Maestix.guide
maestix/Maestix.guide.info
maestix.info

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 mir oder der maestix.library. |
| |
`-------------------------------------------------------------------'
```

1.10 Maestix: Kontaktadresse

Wenn Sie mich anschreiben möchten, verwenden Sie bitte meine E-Mail-Adresse:
shred@tfh.dssd.sub.org (Deutschland) oder
shred@eratosth.wwb.sub.de (Internet).
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)-(O)-(O)-O
| |
| KÖLN'S GRÖßTE AMIGA & PC MAILBOX: |
| __ _ __ |
! | ZyXEl....19.2kbd / \ | .G.I.G.A.bYtE .fASTCAll SYStEM. :
: l___ ZyXEl....19.2kbd / l__ :..68O3O/33MhZ..: :
```

: \ ZyXEl....16.8kbd / \ .O.N.l.I.N.E. :::::::::::::::::: :
: ___/ hSt.dUAl.14.4kbd (__ o ___/ :::::5:NODES::::: :
: CTK......14.4kbd :.MUltiUSERChAt.: :
: :
O-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-(O)-O
| |
! .xX.SOftWARE.AVAilABlE.Xx. !
: ~"""""""""""""""""""""""""""""~ :
: AMiGA.....:RIESIGE AUSWAHL AN AMiGA PROGRAMMEN AUS GRAFiK, :
: MUSiK, SYStEM UNd AllEN ANdEREN bEREICHEN ...oO> 72o MB :
: ~"""""~ :
: PC.....:REICHHALTiGES ANGEbOt DER AkTUELLEN RElEASES, UND :
: bEStEN PROGRAMME fUER AllE ANWENdUNGEN ......oO> 64o MB :
: ~"""""~ :
: GRAFiK.....:UNSERE GRAfIK SECtION iST RANDVOll MiT hOCHWERT- :
: iGEN bILDERN AllER GAENGiGEN FORMAtE .....oO> 58o MB :
: ~"""""~ :
: SOUNd.....:fUTTER fUER dIE SOUNdkARTE- MODUlE, SAMPlES, :
: ANRUftEXtE ETC., NUR VOM fEINSTEN .....oO> 55o MB :
: ~"""""~ :
: NETZE.....:fIDO,Z-NEtZ,T-NEtZ,Gl-NEtZ.UNd.VIElE.MEhR.MIt. :
: POiNTMOEGLiCHkEIT.FUER.EiNGETRAGENE.USER.....oO> 29o MB :
: ~"""""~ :
: :
: :
: .oOo._.oOo._.oOo._.oOo._.oOo._.oOo._.oOo._.oOo._.oOo._.oOo._.oOo._.oOo. :
. ! ! .
. :.....tAEGlICH.fRISCHE.UPDAtES.UNd.RElEASES.....: .
. : : .
. :.....AllE.fILES.1oo%.GEPRUEfT.UNd.QUAlITAtIV.hOCHWERtIG.....: .
. : : .
. :.....kOMPEtENTES.SYSOPtEAM.hILFt.iN.AllEN.COMPUtERPROBlEMEN.....: .
. : : .
. :.....CONSOlE.SECtION.MIt.NEUEStEN.INfOS.tIPS.tRICKS.UNd.tEStS.....: .
. : : .
. :....iNFO-SECtION..iMMER.dIE.AkTUELLStEN.UNd.hEISSESTEN.NACHRICHtEN...: .
. : : .
. o..GUtE.ONlINEGAMES.WIE..MECHASSAULt..SOCCER.MANAGER..LABYkILL..ETC...o .
. :~"""~: .

. . dAS iST jEDOCH NOCH lANGE NiCHT AllES WAS iN dER fUNhOUSE GEbOTEN . .

. WiRD. iN UNSEREN tEXT-SECtIONEN WiRD HEftIG UEbER ViELE tHEMEN .

. dISKUTIERT UNd dA WAEREN NOCH DiVERSE UPdATE-SERViCES UNd,Und, und.....

. .

O-(O)-(O)-(O)-(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 | | \ | \/

E| _ ___/ l | _ | | T | l l__ / _l__

| Y | l_ \ | | | | l_/ / \

| | | / |\ | ___ | | / / \

l____| l_____/___j ___j T I___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)-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) *** KÜRTEN (02268) *** :

: *** MUCH (02245) *** DABRINGHAUS.(02193) *** BURSCHEID (02174) *** :

. .

. ...lEITUNG.1...:...o22o4.61630...hST.dUAL14.4oo... .

. ...lEITUNG.2...:...o22o4.61499...ZYXEl.+.19.2oo... .

. ...lEITUNG.3...:...o22o4.23856...ZYXEl...16.8oo... .

. ...lEITUNG.4...:...o22o4.23955...CTK.....14.4oo... .

. ...lEITUNG.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)-(O)-O

1.11 Maestix: Bugs

Geschrieben wurde Maestix auf einem Amiga 4000/o3o mit Kick 3.0,
8 MB RAM, FastLane Z3 mit 420MB HD, und einer Maestro :-)
Getestet wird Maestix ständig auf folgenden Rechnern:
· A4000/o3o (MMU) , Kick 3.0 , 8 MB RAM (6F/2C), 420MB HD,
Fastlane, Maestro
· A4000/o3o (MMU,FPU) , Kick 3.0 , 6 MB RAM (4F/2C), 400MB HD,
Emplant, Maestro
· A3000/o3o (MMU,FPU) , Kick 3.1 ,10 MB RAM (8F/2C), 1,4GB HD,
Maestro
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 Beendung des zweiten Prozesses, da
in diesem Fall ein Deadlock entsteht. Fixing soon...
Folgende Enforcer-Hits oder Mungwall-Hits sind bekannt:
· keine bekannt :-)

1.12 Maestix: History

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

1.13 Maestix: Zukunft

Folgendes ist für die nächsten Versionen der Library geplant:
· Unterstützung mehrerer Maestro's :)
· Konfig-Editor für Standard-Eingang etc.
· Diverse Unterstützungsroutinen (z. B. für Echtzeiteffekte)
· Automatische Eingangs-Selektion
· Verbesserte Verträglichkeit mit der seriellen Schnittstelle
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 einge-
stellt, wenn es in die User-Startup eingefügt wird.
Die Installation ist dann abgeschlossen.

1.15 Maestix: Shell-Tools

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

1.16 Maestix: SetMstx

Das Programm stellt die Default-Parameter der maestix.library ein.
Momentan kann nur der Default-Eingang eingestellt werden.
SetMstx INPUT optical wählt den optischen Eingang als
Standard-Eingang aus.
SetMstx INPUT coaxial wählt den koaxialen Eingang als
Standard-Eingang aus.
Das Programm wird üblicherweise in der User-Startup aufgerufen, kann aber auch jederzeit durch die Shell aufgerufen werden.

1.17 Maestix: AllocMstx und FreeMstx

Die ursprünglichen Programme, die die Maestro 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.
Zum Start von den Originalprogrammen empfiehlt sich jetzt ein Script wie:

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

1.18 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 Maestro 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 Maestro 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 Maestro 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 Maestro automatisch.

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

1.19 Maestix: Benutzung der Library

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

1.20 Maestix: Allokieren

Bevor die Maestro 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 Maestro geschrieben wurden,

also MaestroBR und Samplitude. Diese Programme benutzen die
Maestro-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!

1.21 Maestix: Setzen

Nachdem die Maestro 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 sie 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 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_48000 48kHz-Rate (DAT) (Default)
RATE_44100 44,1kHz-Rate (CD)
RATE_44100SUB 44,1kHz-Rate (CD-Submode)
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 Maestro 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 ge-
legt. Die Subcodes werden dabei
nicht manipuliert. (Default)
OUTPUT_INPUT Das Signal vom Eingang wird de-
codiert und anschließend mit den
neuen Subcodes wieder codiert.
OUTPUT_FIFO Die von den Transmit-FIFOs kom-
menden Daten werden mit den Sub-
codes codiert und ausgegeben.
Hierfür wird die Systemrate ver-
wendet.

1.22 Maestix: Status-Abfrage

Die Funktion GetStatus() ermittelt den aktuellen Status der Maestro.
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 wer-
den keine Daten gesendet bzw. em-
pfangen.
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.
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.

1.23 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.
Die Message-Struktur finden Sie in den Include-Dateien unter DataMessage-Struct.

1.24 Maestix: Realisation

Im Folgenden beschreibe ich die praktische Anwendung der Maestro-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 Echtzeiteffekte 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!

1.25 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.

1.26 Maestix: Bitmanipulation

Eine reine Manipulation der UDBs und CSBs ist mit der Maestro 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.

1.27 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.

1.28 Maestix: Echtzeiteffekte

Wenn der Sende- und Empfangsvorgang kombiniert wird, können auch Echtzeiteffekte mit der Maestro realisiert werden.
Jede Nachricht, die der Klient von dem Empfänger zurückerhält, sollte mit dem Effekt bearbeitet und anschließend an den Sender geschickt werden. Nachrichten vom Sender können direkt wieder an den Empfänger gesendet werden.

1.29 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.

1.30 Maestix: AnalyzeInput

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

1.31 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 Maestro selbstständig die CSBs um.

1.32 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 Maestro 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 Maestro freigibt und dem Hauptprogramm dann signalisiert, daß er nun beendet ist.
Das Hauptprogramm gibt anschließend alle Ressourcen frei.

1.33 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.

1.34 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.

1.35 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.

1.36 Maestix: Credits

Ich möchte mich bei folgenden Leuten bedanken:
Sven Arke daß er mich mit seiner Mailbox
unterstützt.
Andreas Benden für's Beta-Testen
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.
Commodore Amiga für die Entwicklung des besten
Computers!
Ohne sie wäre die Maestix-Library niemals entstanden oder entwickelt
worden.
//
\ // -- Amiga - Jetzt erst recht! --
\X/