MME Erweiterung

Version 2.3
Stand 12.06.2006

© 06/2006 MARIAN, Kopieren, Verteilung, Publikation über Medien jeder Art ist nur mit schriftlicher Genehmigung von MARIAN OHG erlaubt.

1. MME und die Erweiterung

Die Aufnahme und Wiedergabe von Audiomaterial sowie die Steuerung von Audiogeräten ist prinzipiell über standardisierte Treiberschnittstellen verschiedenster Hersteller möglich. Viele Anwenderprogramme nutzen traditionell bedingt neben ASIO und EASI auch weiterhin die Multimediaschnittstellen von Microsoft Windows™. Die im Folgenden zu beschreibende Treibererweiterung nutzt ebenfalls das Microsoft Multimedia System und setzt auf einer Ebene auf, die herkömmlicherweise als MME bezeichnet wird. Gemeint sind damit die Funktionen *waveInxxx* und *waveOutxxx* die durch die Module *winmm.dll* (32 Bit) und *mmsystem.dll* (16 Bit) des Windows Multimedia Systems bereitgestellt werden. Diese Funktionen werden im Microsoft SDK hinlänglich beschrieben. Zusätzlich bietet MME zwei Funktionen, die eine Erweiterung des Standard-Funktionsumfangs erlauben: *waveInMessage* und *waveOutMessage*. Über diese Funktionen können Applikationen frei definierbare Messages mit zwei Parametern an ein WaveIn/Out Gerät schicken. Auf diesen Funktionen basiert die MARIAN MME Erweiterung und bietet damit Applikationen den Zugriff auf die zusätzlichen Features der MARIAN-Karten.

2. Verfügbarkeit der MME Erweiterung

Die MARIAN MME Erweiterung wurde bis zum März 2006 für folgende Produkte implementiert:
ARC 88, Prodif 96 PRO, Prodif PLUS, Marc 8 Midi, Marc 2, Marc 2 PRO, Marc 4 Midi, Marc 4 Digi. Sie steht sowohl unter den Betriebssystemen Windows™ 95/98/ME als auch unter Windows™ NT4/2000/XP zur Verfügung.

Unter Windows™ 2000/XP ist folgendes zu beachten:
Ab den Treiberversionen v2.0 werden die Aufnahme- und Wiedergabegeräte der MME an WDM-Audio (Kernel) angebunden und die Schnittstelle zu den MME-Geräten wird durch Microsoft geliefert. Diese Geräte „verstehen“ nicht die MARIAN-spezifischen Messages, die über *waveInMessage* und *waveOutMessage* gesendet werden können. Außerdem lassen Sie sich nicht als MARIAN-Produkt identifizieren (siehe *waveInGetDevCaps* und *waveOutDevCaps*). Deshalb müssen für Anwendungen, welche die MARIAN MME Erweiterung nutzen wollen, die klassischen MME-Treiber aktiviert werden. Eine entsprechende Option finden Sie in den Einstellungen der jeweiligen Treiberoberfläche. Nachdem diese Option aktiviert ist, stehen zusätzliche MME-Geräte zur Verfügung, welche die MARIAN-spezifischen Messages verstehen. Diese Geräte sind durch Ihre korrekte Identifikation und durch ein „(MME)“ im Gerätenamen zu erkennen. Ausschließlich diese Geräte lassen sich über die MARIAN MME Erweiterung steuern.

3. Identifizierung von MARIAN Aufnahme- und Wiedergabegeräten

Für jedes Stereopaar der verfügbaren Eingangskanäle einer MARIAN-Karte wird vom Treiber ein Aufnahmegerät zur Verfügung gestellt, für jedes Stereopaar der verfügbaren Ausgangskanäle jeweils ein Wiedergabegerät. Die Namen und Eigenschaften dieser Geräte können mittels der MME-Standardfunktionen *waveInGetNumDevs/waveOutGetNumDevs* und *waveInGetDevCaps/waveOutGetDevCaps* herausgefunden werden.

Zur Identifizierung des Wave-Gerätes als MARIAN-Gerät können die Felder `wMid` und `wPid` der Strukturen `WAVEINCAPS` bzw. `WAVEOUTCAPS` herangezogen werden. Die entsprechenden Werte entnehmen Sie bitte der folgenden Aufstellung oder auch dem beiliegenden Header-File.

wMid (Manufacturer Id)

Bezeichner	Wert	Hersteller
`MM_MARIAN`	190	MARIAN

wPid (Product Id)

Bezeichner	Wert	Gerät
`MM_MARIAN_ARC88_WAVEIN`	5	ARC88 Aufnahmegerät
`MM_MARIAN_ARC88_WAVEOUT`	6	ARC88 Wiedergabegerät
`MM_MARIAN_PD96PRO_ANLG_WAVEOUT`	7	Prodif 96 PRO Analog Wiedergabegerät
`MM_MARIAN_PD96PRO_ANLG_WAVEIN`	8	Prodif 96 PRO Analog Aufnahmegerät
`MM_MARIAN_PD96PRO_DIG_WAVEOUT`	9	Prodif 96 PRO Digital Wiedergabegerät
`MM_MARIAN_PD96PRO_DIG_WAVEIN`	10	Prodif 96 PRO Digital Aufnahmegerät
`MM_MARIAN_PD96PRO_ADAT_WAVEOUT`	11	Prodif 96 PRO Adat Aufnahmegerät
`MM_MARIAN_PD96PRO_ADAT_WAVEIN`	12	Prodif 96 PRO Adat Wiedergabegerät
`MM_MARIAN_PDPLUS_ANLG_WAVEOUT`	13	Prodif PLUS Analog Wiedergabegerät
`MM_MARIAN_PDPLUS_ANLG_WAVEIN`	14	Prodif PLUS Analog Aufnahmegerät
`MM_MARIAN_PDPLUS_DIG_WAVEOUT`	15	Prodif PLUS Digital Wiedergabegerät
`MM_MARIAN_PDPLUS_DIG_WAVEIN`	16	Prodif PLUS Digital Aufnahmegerät
`MM_MARIAN_PDPLUS_ADAT_WAVEOUT`	17	Prodif PLUS Adat Wiedergabegerät
`MM_MARIAN_PDPLUS_ADAT_WAVEIN`	18	Prodif PLUS Adat Aufnahmegerät
`MM_MARIAN_ARC8896_WAVEIN`	19	Marc 8 Midi Aufnahmegerät
`MM_MARIAN_ARC8896_WAVEOUT`	20	Marc 8 Midi Wiedergabegerät
`MM_MARIAN_MARC2_ANALOG_WAVEIN`	23	Marc 2 Analog Aufnahmegerät
`MM_MARIAN_MARC2_ANALOG_WAVEOUT`	24	Marc 2 Analog Wiedergabegerät
`MM_MARIAN_MARC2_DIGITAL_WAVEIN`	25	Marc 2 Digital Aufnahmegerät
`MM_MARIAN_MARC2_DIGITAL_WAVEOUT`	26	Marc 2 Digital Wiedergabegerät
`MM_MARIAN_MARC4M_WAVEIN`	31	Marc 4 Midi Aufnahmegerät
`MM_MARIAN_MARC4M_WAVEOUT`	32	Marc 4 Midi Wiedergabegerät
`MM_MARIAN_MARC4D_ANALOG_WAVEIN`	37	Marc 4 Digi Analog Aufnahmegerät
`MM_MARIAN_MARC4D_ANALOG_WAVEOUT`	38	Marc 4 Digi Analog Wiedergabegerät
`MM_MARIAN_MARC4D_DIGITAL_WAVEIN`	39	Marc 4 Digi Digital Aufnahmegerät
`MM_MARIAN_MARC4D_DIGITAL_WAVEOUT`	40	Marc 4 Digi Digital Wiedergabegerät
`MM_MARIAN_MARC2P_ANALOG_WAVEIN`	41	Marc 2 PRO Analog Aufnahmegerät
`MM_MARIAN_MARC2P_ANALOG_WAVEOUT`	42	Marc 2 PRO Analog Wiedergabegerät
`MM_MARIAN_MARC2P_DIGITAL_WAVEIN`	43	Marc 2 PRO Digital Aufnahmegerät
`MM_MARIAN_MARC2P_DIGITAL_WAVEOUT`	44	Marc 2 PRO Digital Wiedergabegerät

4. Erweiterte Funktionen für Wiedergabegeräte

Zur Funktionserweiterung wurden für die Funktion *WaveOutMessage* folgende Botschaften eingeführt, die dem Parameter `uMessage` dieser Funktionen übergeben werden müssen. Diese Botschaften wurden auch im beiliegenden Headerfile implementiert.

```
DRVM_USER           = 4000h
WODM_GETMUTE        = DRVM_USER + 1;
WODM_SETMUTE        = DRVM_USER + 2;
WODM_MON_GETVOLUME  = DRVM_USER + 3;
WODM_MON_SETVOLUME  = DRVM_USER + 4;
WODM_ENABLELEVEL    = DRVM_USER + 7;
WODM_DISABLELEVEL   = DRVM_USER + 8;
WODM_GETLEVEL       = DRVM_USER + 9;
WODM_GETROUTING     = DRVM_USER + 10;
WODM_SETROUTING     = DRVM_USER + 11;
WODM_GETSAMPLERATE  = DRVM_USER + 12;
WODM_GETFORMAT      = DRVM_USER + 13;
WODM_SETFORMAT      = DRVM_USER + 14;
```

WaveOutMessage ist wie folgt deklariert:

```
DWORD waveOutMessage(
  HWAVEOUT hwo,
  UINT     uMsg,
  DWORD    dwParam1,
  DWORD    dwParam2
);
```

Parameter:

`hwo`	Handle des Wiedergabegerätes (Rückgabewert von waveOutOpen)
`uMsg`	Eine der neu definierten WODM Botschaften
`dwParam1`	botschaftsabhängiger Parameter 1
`dwParam2`	botschaftsabhängiger Parameter 1

Return: *WaveOutMessage* gibt sinnentsprechend die Standard `mmsyserr_xxx` Codes zurück. *WaveOutMessage* gibt den allgemeinen Fehlercode `mmsyserr_error` zurück, wenn eine Funktion auf Grund vn hardwarspezifischen Restriktionen nicht ausgeführt werden kann.

Headerfile: mmsystem.h

WIN32 Anwendungen unter Windows™ 95/98/ME

Für verschiedene Funktionen wird als Parameter eine Pointer erwartet. Da Wave-Treiber unter Windows 95/98™ eine 16 Bit Protect Mode Architektur aufzuweisen haben, werden diese Pointer im Format SEG:OFS interpretiert.
Nutzen Sie folgende Funktionen der WOW32.DLL (Bestandteil von Windows95/98™), um derartige Pointer sowohl in Ihrer WIN32 Applikation als auch dem Treiber zugänglich zu machen:

```
DWORD WINAPI WOWGlobalAllocLock16(
    WORD   wFlags,        // object allocation flags
    DWORD  cb,            // number of bytes to allocate
    LPWORD phMem          // handle to global memory object
   );

LPVOID WINAPI WOWGetVDMPointer(
    DWORD vp,             // 16:16 address
    DWORD dwBytes,        // size of vp block
    BOOL  fProtectedMode // protected mode flag
   );
```

Weitere Informationen zu diesen Funktionen und deren Umfeld erhalten Sie im „Microsoft SDK“ unter „Programming and Tools Guides“ | „Programming Techniques“ | „Generic Thunks“.

4.1 Lautstärke

4.1.1 WODM_GETMUTE

Verfügbarkeit: Arc88 Analog Out, Prodif PLUS Analog Out, Marc 8 Midi Analog Out, Marc 2 Analog Out, Marc 2 PRO Analog Out, Marc 4 Midi/Digi Analog Out

Mit dieser Funktion können Sie den Mutestatus der zwei Kanäle des Wiedergabegerätes abholen.

uMessage	dwParam1	dwParam2
WODM_GETMUTE	Pointer auf 32bit Wert, der wie folgt gefüllt wird: Lower WORD = 0 = Linker Kanal An Lower WORD = 1 = Linker Kanal Aus Higher WORD = 0 = Rechter Kanal An Higher WORD = 1 = Rechter Kanal Aus	unbenutzt

4.1.2 WODM_SETMUTE

Verfügbarkeit: ARC 88 Analog Out, Prodif PLUS Analog Out, Marc 8 Midi Analog Out, Marc 2 Analog Out, Marc 2 PRO Analog Out, Marc 4 Midi/Digi Analog Out

Mit dieser Funktion setzen Sie den Mutestatus der zwei Kanäle des Wiedergabegerätes.

uMessage	dwParam1	dwParam2
WODM_SETMUTE	32bit Wert, dessen Inhalt wie folgt ausgewertet wird: Lower WORD = 0 = Linker Kanal An Lower WORD = 1 = Linker Kanal Aus Higher WORD = 0 = Rechter Kanal An Higher WORD = 1 = Rechter Kanal Aus	unbenutzt

4.1.3 WODM_MON_GETVOLUME

Verfügbarkeit: ARC 88 Analog Out

Mit dieser Funktion können Sie die eingestellte Lautstärke, mit der zwei Eingangskanäle zu den entsprechenden Ausgangskanälen gemischt werden, abholen. Diese Funktion entspricht in ihrem Verhalten der von MME Standardfunktion *WaveOutGetVolume*, bezieht sich aber auf og. Mixerlautstärke.

Beachten Sie, dass jede routebare Eingangsquelle (siehe WIDM_SETROUTING im Kapitel 5.3.3) seinen eigenen Volumenwert hat. Das heißt: nach einer Veränderung des Signalroutings mittels *WIDM_SETROUTING* wird durch *WODM_MON_GETVOLUME* ein anderer, der neuen Eingangsquelle zugeordneter Wert zurückgegeben wird.

uMessage	dwParam1	dwParam2
WODM_MON_GETVOLUME	Pointer auf 32bit Wert. Das niederwertige WORD des 32bit Wertes wird mit dem Volumenwert des linken Kanals gefüllt, das höherwertige WORD mit dem Volumenwert des rechten Kanals. 0000h entspricht -34.5 dB FFFFh entspricht +12 dB Zwischenschritte werden innerhalb dieses Bereichs linear aufgelöst.	unbenutzt

4.1.4 WODM_MON_SETVOLUME

Verfügbarkeit: ARC 88 Analog Out

Mit dieser Funktion setzen Sie die Lautstärke, mit der zwei Eingangskanäle zu den entsprechenden Ausgangskanälen gemischt werden. Diese Funktion entspricht in ihrem Verhalten der von *WaveOutSetVolume*, bezieht sich aber auf og. Mixerlautstärke.

Beachten Sie, dass jede routebare Eingangsquelle (siehe WIDM_SETROUTING im Kapitel 5.3.3) seinen eigenen Volumenwert hat. Das heißt, dass Sie durch *WODM_MON_SETVOLUME* den Volumenwert der gerade gerouteten Eingangsquelle verändern.

uMessage	dwParam1	dwParam2
WODM_MON_SETVOLUME	32bit Wert. Das niederwertige WORD des 32bit Wertes enthält Volumenwert des linken Kanals, das höherwertige WORD den Volumenwert des rechten Kanals. 0000h entspricht -34.5 dB FFFFh entspricht +12 dB Zwischenschritte werden innerhalb dieses Bereichs linear aufgelöst.	unbenutzt

4.2 Levelerfassung

4.2.1 WODM_ENABLELEVEL

Verfügbarkeit: Alle MARIAN Wiedergabegeräte

Mit dieser Funktion können Sie das Erfassen und Auswerten der Levelinformationen von Audiodaten, die zu einem Wiedergabegerät geschickt werden, einschalten. Das Einschalten dieser Funktion hat allerdings eine Herabsetzung der Systemperformance zu Folge.

uMessage	dwParam1	dwParam2
WODM_ENABLELEVEL	unbenutzt	unbenutzt

4.2.2 WODM_DISABLELEVEL

Verfügbarkeit: Alle MARIAN Wiedergabegeräte

Mit dieser Funktion schalten Sie das Erfassen und Auswerten der Levelinformationen von Audiodaten, die zu einem Wiedergabegerät geschickt werden, ab.

uMessage	dwParam1	dwParam2
WODM_DISABLELEVEL	unbenutzt	unbenutzt

4.2.3 WODM_GETLEVEL

Verfügbarkeit: Alle MARIAN Wiedergabegeräte

Mittels dieser Funktion holen Sie sich den zuletzt gespeicherten Spitzenwert der Signalamplitude ab. Danach setzt der Treiber diesen internen Wert auf 0 und setzt die Erfassung von Levelspitzenwerten fort. Sie müssen zuerst die Funktion *WODM_ENABLELEVEL* aufrufen, um diese Funktion nutzen zu können.

uMessage	dwParam1	dwParam2
WODM_GETLEVEL	Pointer auf Speicherbereich, der 2 aufeinanderfolgende 32 Bit Werte, speichern kann und wie folgt gefüllt wird: Lower DWORD = PEAK linker Kanal Higher DWORD = PEAK rechter Kanal	unbenutzt

4.3 Routing

Über die Ausgangs-Routing-Funktionen WODM_GETROUTING und WODM_SETROUTING (Kapitel 0 und 4.3.3) können Sie die Signalquelle für den Ausgang eines Wiedergabegerätes feststellen bzw. bestimmen.

4.3.1 OUTROUTING Formate

Die Funktionen WODM_GETROUTING und WODM_SETROUTING erwarten einen DWORD Parameter, der das entsprechende Ausgangs-Routing festlegt. Dieser Parameter ist jedoch gerätespezifisch und soll im Folgenden erläutert werden.

4.3.1.1 OUTROUTING Format Prodif 96 PRO, Prodif PLUS

```
typedef struct {
  BYTE     Route;
  BYTE     AdatLeftChn;
  BYTE     AdatRightChn;
  BYTE     reserved;
} OUTROUTING, *POUTROUTING;
```

`Route`

Signalquelle für Ausgang.
Zulässige Werte:

```
#define oroute_Play    0      // Wiedergabesignal der Applikation

#define oroute_OppOut  1      // Ausgangssignal ist Signal des „anderen"
                              // Ausgangs. Bei Anwendung auf Analog Out,
                              // das Signal an Digital Out, Bei Anwendung
                              // auf Digital Out, das Signal an Analog Out

#define oroute_AnaInp  2      // Ausgangssignal ist Signal am Analog In

#define oroute_DigInp  3      // Ausgangssignal ist Signal am Digital In

#define oroute_AdatInp 4      // Ausgangssignal ist Signal am Adat In
                              // Welche Adat Eingangskanäle am linken bzw.
                              // rechten Kanal des Ausgangs wiedergegeben
                              // werden, wird durch die Felder AdatLeftChn
                              // und AdatRightChn festgelegt.
```

`AdatLeftChn`, `AdatRightChn`

Adatkanal für linken bzw. rechten Kanal des Analogausgangs bei `Route == oroute_AdatInp`
Zulässige Werte:

0..7 für Adat Kanäle 1..8

Bemerkungen:

Der Routingwert `orouteAdatInp` ist nur auf das analoge Wiedergabegerät anwendbar und hat ggf. eine Umschaltung in die ADAT Betriebsart der Karte zur Folge. Die Routingwerte `orouteDigInp` und `orouteOppositeOut` (für Analog Out) haben ggf. eine Umschaltung in die Stereo Digital Betriebsart der Karte zur Folge. Mehr zu den Betriebsarten der Prodif's lesen Sie bitte unter "Besonderheiten Prodif 96 Pro und Prodif PLUS" im Kapitel 8.

Beachten Sie bitte auch die automatischen Eingangsmonitor-Modi, die unter „INPROUTING Prodif 96 PRO, Prodif PLUS und MARC 2" im Kapitel 5.3.1.2 beschrieben sind.

4.3.1.2 OUTROUTING Format Marc 8 Midi

```
typedef struct {
  BOOLEAN InpMonOnRec;
  BOOLEAN InpMonOnPunch;
  __int16 Route;
} OUTROUTING, *POUTROUTING;
```

Route:

Signalquelle für den Ausgang.
Wertebereich:

```
#define oroutePlay -1    // Wiedergabesignal der Applikation
#define orouteInp12 0    // Signal am Eingang 1-2
#define orouteInp34 1    // Signal am Eingang 3-4
#define orouteInp56 2    // Signal am Eingang 5-6
#define orouteInp78 3    // Signal am Eingang 7-8
```

InpMonOnRec:

Schaltet Auto-Record.Monitor-Mode ein oder aus. Ist dieser Modus aktiviert, so findet ein automatisches Routing des festgelegten Eingangs zum Ausgang statt, wenn eine Aufnahme für diesen Eingang gestartet wird (durch *waveInStart*).
Wertebereich:

```
TRUE || FALSE
```

InpMonOnPunch:

Schaltet den Monitor-OnPunch-Mode ein oder aus. Ist dieser Modus aktiviert, so findet ein automatisches Routing des festgelegten Eingangs zum Ausgang statt, wenn diesem Eingang durch WIDM_MON_SETMUTE das PunchIn oder PunchOut einer Aufnahme signalisiert wird. Mehr hierzu lesen Sie bitte unter „MonitoringMonitoring“ im Kapitel 6.
Wertebereich:

```
TRUE || FALSE
```

4.3.1.3 OUTROUTING Format Marc 2 und Marc 2 PRO

```
typedef struct {
  BYTE     Route;
  BYTE     reserved;
  BYTE     reserved;
  BYTE     reserved;
} OUTROUTING, *POUTROUTING;
```

Route

Signalquelle für Ausgang.
Zulässige Werte:

```
#define oroute_Play   0       // Ausgangssignal ist Wiedergabesignal der
                              // Applikation
#define oroute_OppOut 1       // Ausgangssignal ist Signal des „anderen“
                              // Ausgangs. Bei Anwendung auf Analog Out,
                              // das Signal an Digital Out, Bei Anwendung
                              // auf Digital Out, das Signal an Analog Out

#define oroute_AnaInp 2       // Ausgangssignal ist Signal am Analog In

#define oroute_DigInp 3       // Ausgangssignal ist Signal am Digital In
```

Beachten Sie bitte auch die automatischen Eingangsmonitor-Modi, die unter „INPROUTING Prodif 96 PRO, Prodif PLUS und MARC 2“ im Kapitel 5.3.1.2 beschrieben sind.

4.3.1.4 OUTROUTING Format Marc 4 Midi

```
typedef struct {
  BOOLEAN InpMonOnRec;
  BOOLEAN InpMonOnPunch;
  __int16 Route;
} OUTROUTING, *POUTROUTING;
```

Signalquelle für den Ausgang.
Wertebereich:

```
#define oroutePlay -1    // Wiedergabesignal der Applikation
#define orouteInp12 0    // Signal am Eingang 1-2
#define orouteInp34 1    // Signal am Eingang 3-4
```

`InpMonOnRec:`

Schaltet Auto-Record.Monitor-Mode ein oder aus. Ist dieser Modus aktiviert, so findet ein automatisches Routing des festgelegten Eingangs zum Ausgang statt, wenn eine Aufnahme für diesen Eingang gestartet wird (durch *waveInStart*).
Wertebereich:

```
TRUE || FALSE
```

`InpMonOnPunch:`

Schaltet den Monitor-OnPunch-Mode ein oder aus. Ist dieser Modus aktiviert, so findet ein automatisches Routing des festgelegten Eingangs zum Ausgang statt, wenn diesem Eingang durch WIDM_MON_SETMUTE das PunchIn oder PunchOut einer Aufnahme signalisiert wird. Mehr hierzu lesen Sie bitte unter „Monitoring“ im Kapitel 6.
Wertebereich:

```
TRUE || FALSE
```

4.3.1.5 OUTROUTING Format Marc 4 Digi

```
typedef struct {
  BYTE     Route;
  BYTE     reserved;
  BYTE     reserved;
  BYTE     reserved;
} OUTROUTING, *POUTROUTING;
```

`Route`

Signalquelle für Ausgang.
Zulässige Werte:

```
#define oroute_AnaPlay12  0   // Ausgangssignal ist Wiedergabesignal einer
                                // Applikation auf „Analog 1-2“
#define oroute_AnaInp12   1   // Ausgangssignal ist Eingangssignal
                                // „Analog 1-2“
#define oroute_AnaPlay34  2   // Ausgangssignal ist Wiedergabesignal einer
                                // Applikation auf „Analog 3-4“
#define oroute_AnaInp34   3   // Ausgangssignal ist Eingangssignal
                                // „Analog 3-4“
#define oroute_DigPlay    4   // Ausgangssignal ist Wiedergabesignal einer
                                // Applikation auf „Digital“
#define oroute_DigInp     5   // Ausgangssignal ist Eingangssignal
                                // „Digital“
```

4.3.2 WODM_GETROUTING

Verfügbarkeit: Alle MARIAN Wiedergabegeräte außer ARC 88, nicht anwendbar auf die ADAT Wiedergabegeräte der Prodif 96 PRO und Prodif PLUS

Über diese Funktion finden Sie die Quelle des Signals heraus, die am entsprechenden Ausgang des Wiedergabegerätes ausgegeben wird. Ggf. enthält der zurückgegebene Wert auch Informationen über die automatischen Routingmodi (siehe „Monitoring“ im Kapitel 6)

uMessage	dwParam1	dwParam2
WODM_GETROUTING	Pointer auf 32 Bit Wert. Die Interpretation dieses Wertes ist gerätespezifisch: Prodif's, Marc 8 Midi, Marc 2, Marc 2 PRO, Marc 4 Midi, Marc 4 Digi	unbenutzt

4.3.3 WODM_SETROUTING

Verfügbarkeit: Alle MARIAN Wiedergabegeräte außer ARC 88, nicht anwendbar auf die ADAT Wiedergabegeräte der Prodif 96 PRO und Prodif PLUS

Über diese Funktion legen Sie die Quelle des Signals fest, die am entsprechenden Ausgang des Wiedergabegerätes ausgegeben wird. Ggf. legen Sie hier auch die automatischen Routingmodi fest. (siehe „Monitoring“ im Kapitel 6)

uMessage	dwParam1	dwParam2
WODM_GETROUTING	32 Bit Wert. Die Interpretation dieses Wertes ist gerätespezifisch: Prodif's, Marc 8 Midi, Marc 2, Marc 2 PRO, Marc 4 Midi, Marc 4 Digi	unbenutzt

4.4 Wiedergabeformat

4.4.1 WODM_GETSAMPLERATE

Verfügbarkeit: Alle MARIAN Wiedergabegeräte

Diese Funktion erlaubt Ihnen das Abfragen der aktuellen Samplerate, mit der das Wiedergabegerät läuft. Normalerweise ist dies die Samplerate, mit der das Wiedergabegerät durch *waveOutOpen* geöffnet wurde. Wird das Wiedergabegerät jedoch durch eine andere als die interne Clock gesteuert (z.B. externe Wordclock oder Clock des Digital Input) so wird die Samplerate dieser Clock zurückgegeben.

uMessage	dwParam1	dwParam2
WODM_GETSAMPLERATE	Pointer auf 32 Bit Wert der mit der aktuellen Samplerate gefüllt wird	unbenutzt

4.4.2 WODM_GETFORMAT

Verfügbarkeit: Digital Wiedergabegeräte Prodif 96 PRO und Prodif PLUS; außer ADAT Wiedergabegeräte

Über diese Funktion erhalten Sie gerätespezifische Formatinformationen. Im Falle von Stereo-Digital Wiedergabegeräten sind dies Informationen des Digital Datenformats.

uMessage	dwParam1	dwParam2
WODM_GETFORMAT	Pointer auf 32 Bit Wert der mit den Formatinformationen gefüllt wird Prodif 96 PRO: 0x00000000 S/PDIF Audio 0x00000010 AES/EBU Audio Prodif PLUS: 0x00000020 S/PDIF Audio 0x00000088 AES/EBU Audio 0x00000080 AES/EBU Non Audio	unbenutzt

4.4.3 WODM_SETFORMAT

Verfügbarkeit: Digital Wiedergabegeräte Prodif 96 PRO und Prodif PLUS außer ADAT Wiedergabegeräte

Mit dieser Funktion legen Sie gerätespezifische Formate fest. Im Falle von Stereo-Digital Wiedergabegeräten ist dies das Digital Datenformat.

uMessage	dwParam1	dwParam2
WODM_SETFORMAT	32 Bit Wert mit den Formatinformationen: Prodif 96 PRO: 0x00000000 S/PDIF Audio 0x00000010 AES/EBU Audio Prodif PLUS: 0x00000020 S/PDIF Audio 0x00000088 AES/EBU Audio 0x00000080 AES/EBU Non Audio	unbenutzt

5. Erweiterte Funktionen für Aufnahmegeräte

Zur Funktionserweiterung wurden für die Funktion *WaveInMessage* folgende Botschaften eingeführt, die dem Parameter `uMessage` dieser Funktionen übergeben werden müssen. Diese Botschaften wurden auch im beiliegenden Headerfile implementiert.

```
DRVM_USER           = 4000h
WIDM_GETVOLUME      = DRVM_USER + 50
WIDM_SETVOLUME      = DRVM_USER + 51
WIDM_GETROUTING     = DRVM_USER + 52
WIDM_SETROUTING     = DRVM_USER + 53
WIDM_GETPITCH       = DRVM_USER + 54
WIDM_SETPITCH       = DRVM_USER + 55
WIDM_ENABLELEVEL    = DRVM_USER + 56
WIDM_DISABLELEVEL   = DRVM_USER + 57
WIDM_GETLEVEL       = DRVM_USER + 58
WIDM_MON_GETVOLUME  = DRVM_USER + 59
WIDM_MON_SETVOLUME  = DRVM_USER + 60
WIDM_MON_GETMUTE    = DRVM_USER + 61
WIDM_MON_SETMUTE    = DRVM_USER + 62
WIDM_GETSAMPLERATE  = DRVM_USER + 63
WIDM_GETFORMAT      = DRVM_USER + 64
WIDM_SETFORMAT      = DRVM_USER + 65
WIDM_GETMUTE        = DRVM_USER + 66
WIDM_SETMUTE        = DRVM_USER + 67
```

WaveInMessage ist wie folgt deklariert:

```
DWORD waveInMessage(
  HWAVEIN hwi,
  UINT uMsg,
  DWORD dwParam1,
  DWORD dwParam2
);
```

Parameter:

`hwi`	Handle des Aufnahmegerätes (Rückgabewert von waveInOpen)
`uMsg`	Eine der neu definierten WIDM Botschaften
`dwParam1`	botschaftsabhängiger Parameter 1
`dwParam2`	botschaftsabhängiger Parameter 1

Return: *WaveInMessage* gibt sinnentsprechend die Standard `mmsyserr_xxx` Codes zurück. *WaveInMessage* gibt den allgemeinen Fehlercode `mmsyserr_error` zurück, wenn eine Funktion auf Grund von hardwarspezifischen Restriktionen nicht ausgeführt werden kann.

Headerfile: mmsystem.h

WIN32 Anwendungen unter Windows95/98™

Für verschiedene Funktionen wird als Parameter eine Pointer erwartet. Da ein Wave-Treiber unter Windows 95™ eine 16 Bit Protect Mode Architektur aufzuweisen hat, werden diese Pointer im Format SEG:OFS interpretiert.
Nutzen Sie folgende Funktionen der WOW32.DLL (Bestandteil von Windows95™), um derartige Pointer sowohl in Ihrer WIN32 Applikation als auch dem Treiber zugänglich zu machen:

```
DWORD WINAPI WOWGlobalAllocLock16(
    WORD wFlags,         // object allocation flags
    DWORD cb,            // number of bytes to allocate
    LPWORD phMem         // handle to global memory object
   );
LPVOID WINAPI WOWGetVDMPointer(
    DWORD vp,            // 16:16 address
    DWORD dwBytes,       // size of vp block
    BOOL fProtectedMode // protected mode flag
   );
```

Weitere Informationen zu diesen Funktionen und deren Umfeld erhalten Sie im „Microsoft SDK“ unter „Programming and Tools Guides“ | „Programming Techniques“ | „Generic Thunks“.

5.1 Eingangspegel/Monitoring

5.1.1 WIDM_GETVOLUME

Verfügbarkeit: Alle analogen MARIAN Aufnahmegeräte außer Prodif PLUS und Marc 2 PRO.

Mit dieser Funktion können Sie die eingestellte Pegelanhebung der zwei Eingangskanäle abrufen. Diese Funktion entspricht in ihrem Verhalten der von *WaveOutGetVolume*, bezieht sich aber auf og. Pegelanhebung.

ARC88: Beachten Sie, dass jede routebare Eingangsquelle (siehe WIDM_SETROUTING Kapitel 5.3.3) seinen eigenen Pegelwert hat. Das heißt: nach einer Veränderung des Signalroutings mittels *WIDM_SETROUTING* wird durch *WIDM_GETVOLUME* ein anderer, der neuen Eingangsquelle zugeordneter Wert zurückgegeben wird.

uMessage	dwParam1	dwParam2
WIDM_GETVOLUME	Pointer auf 32bit Wert. Das niederwertige WORD des 32bit Wertes wird mit dem Pegelwert des linken Kanals gefüllt, das höherwertige WORD mit dem Pegelwert des rechten Kanals. ARC88: 0x0000 0 dB 0xFFFF +22.5 dB Prodif 96 PRO: 0x0000 -95.5 dB 0xFFFF +31.5 dB Marc 8 Midi, Marc 2, Marc 4 Midi/Digi: 0x0000 -72 dB 0xFFFF +18 dB	unbenutzt

5.1.2 WIDM_SETVOLUME

Verfügbarkeit: Alle analogen MARIAN Aufnahmegeräte außer Prodif PLUS und Marc 2 PRO

Mit dieser Funktion setzen Sie die Pegelanhebung der zwei Eingangskanäle. Diese Funktion entspricht in ihrem Verhalten der von *WaveOutSetVolume*, bezieht sich aber auf og. Pegelanhebung.

ARC88: Beachten Sie, dass jede routebare Eingangsquelle (siehe WIDM_SETROUTING Kapitel 5.3.3) seinen eigenen Pegelwert hat. Das heißt, dass Sie durch *WIDM_SETVOLUME* den Pegelwert der gerade gerouteten Eingangsquelle verändern.

uMessage	dwParam1	dwParam2
WIDM_SETVOLUME	32bit Wert. Das niederwertige WORD des 32bit Wertes entspricht dem zu setzenden Pegelwert des linken Kanals, das höherwertige WORD dem zu setzenden Pegelwert des rechten Kanals. ARC88: 0x0000 0 dB 0xFFFF +22.5 dB Prodif 96 PRO: 0x0000 -95.5 dB 0xFFFF +31.5 dB Marc 8 Midi, Marc 2, Marc 4 Midi/Digi: 0x0000 -72 dB 0xFFFF +18 dB	unbenutzt

5.1.3 WIDM_GETMUTE

Verfügbarkeit: Marc 8 Midi, Marc 4 Midi und analoge Aufnahmegeräte Marc 2, Marc 4 Digi

Diese Funktion ermittelt den Mutestatus der zwei Eingangskanäle.

uMessage	dwParam1	dwParam2
WIDM_GETMUTE	Pointer auf 32bit Wert, der wie folgt gefüllt wird: Lower WORD = 0 = Linker Kanal An Lower WORD = 1 = Linker Kanal Aus Higher WORD = 0 = Rechter Kanal An Higher WORD = 1 = Rechter Kanal Aus	unbenutzt

5.1.4 WIDM_SETMUTE

Verfügbarkeit: Marc 8 Midi, Marc 4 Midi und analoge Aufnahmegeräte Marc 2, Marc 4 Digi

Mit diese Funktion setzen Sie den Mutestatus der zwei Eingangskanäle.

uMessage	dwParam1	dwParam2
WIDM_SETMUTE	32bit Wert mit den Mute-Informationen: Lower WORD = 0 = Linker Kanal An Lower WORD = 1 = Linker Kanal Aus Higher WORD = 0 = Rechter Kanal An Higher WORD = 1 = Rechter Kanal Aus	unbenutzt

5.1.5 WIDM_MON_GETVOLUME

Verfügbarkeit: ARC88 Analog

Mit dieser Funktion können Sie die eingestellte Lautstärke, mit der zwei Eingangskanäle zu den entsprechenden Ausgangskanälen gemischt werden, abholen. Diese Funktion entspricht in ihrem Verhalten WODM_MON_GETVOLUME.

Beachten Sie, dass jede routebare Eingangsquelle (siehe WIDM_SETROUTING im Kapitel 5.3.3) seinen eigenen Volumenwert hat. Das heißt: nach einer Veränderung des Signalroutings mittels *WIDM_SETROUTING* wird durch *WIDM_MON_GETVOLUME* ein anderer, der neuen Eingangsquelle zugeordneter Wert zurückgegeben wird.

uMessage	dwParam1	dwParam2
WIDM_MON_GETVOLUME	Pointer auf 32bit Wert. Das niederwertige WORD des 32bit Wertes wird mit dem Volumenwert des linken Kanals gefüllt, das höherwertige WORD mit dem Volumenwert des rechten Kanals. 0000h entspricht -34.5 dB FFFFh entspricht +12 dB	unbenutzt

5.1.6 WIDM_MON_SETVOLUME

Verfügbarkeit: ARC88 Analog

Mit dieser Funktion setzen Sie die Lautstärke, mit der zwei Eingangskanäle zu den entsprechenden Ausgangskanälen gemischt werden. Diese Funktion entspricht in ihrem Verhalten WODM_MON_SETVOLUME.

Beachten Sie, dass jede routebare Eingangsquelle (siehe WIDM_SETROUTING im Kapitel 5.3.3) seinen eigenen Volumenwert hat. Das heißt, dass Sie durch *WIDM_MON_SETVOLUME* den Volumenwert der gerade gerouteten Eingangsquelle verändern.

uMessage	dwParam1	dwParam2
WIDM_MON_SETVOLUME	32bit Wert. Das niederwertige WORD des 32bit Wertes entspricht dem zu setzenden Volumenwert des linken Kanals, das höherwertige WORD dem zu setzenden Volumenwert des rechten Kanals. 0000h entspricht -34.5 dB FFFFh entspricht +12 dB	unbenutzt

5.1.7 WIDM_MON_GETMUTE

Verfügbarkeit: Alle MARIAN Aufnahmegeräte außer die ADAT Aufnahmegeräte der Prodif's

Durch diese Funktion ermitteln Sie, ob das Eingangssignal des Aufnahmegerätes am festgelegten Ausgang zu hören ist. Den festgelegten Monitorausgang ermitteln Sie hardwarespezifisch durch die Funktionen WIDM_GETROUTING oder WODM_GETROUTING.

ARC88: Beachten Sie, dass jede routebare Eingangsquelle (siehe WIDM_SETROUTING im Kapitel 5.3.3) seinen eigenen Mutestatus hat. Das heißt, dass nach einer Veränderung des Signalroutings mittels *WIDM_SETROUTING* durch *WIDM_MON_GETMUTE* ein anderer, der neuen Eingangsquelle zugeordneter Mutestatus zurückgegeben wird.

uMessage	dwParam1	dwParam2
WIDM_MON_GETMUTE	Pointer auf 32bit Wert, der wie folgt gefüllt wird: Lower WORD = 0 = Linker Kanal An Lower WORD = 1 = Linker Kanal Aus Higher WORD = 0 = Rechter Kanal An Higher WORD = 1 = Rechter Kanal Aus	unbenutzt

5.1.8 WIDM_MON_SETMUTE

Verfügbarkeit: Alle MARIAN Aufnahmegeräte außer die ADAT Aufnahmegeräte der Prodif's

Mit dieser Funktion entscheiden Sie, ob das Eingangssignal des Aufnahmegerätes am festgelegten Ausgang zu hören ist. Da diese Funktion auch zum Signalisieren einer PunchIn/PunchOut Aufnahme und dem damit verbundenen automatischen Monitoring genutzt wird, muss diese Funktion zunächst im Monitormodus freigeschaltet werden.
Den Monitorausgang und den Monitormodus legen Sie hardwarespezifisch durch die Funktion WIDM_SETROUTING oder WODM_SETROUTING fest.

Hinweis: Die ARC88 ist die einzige Karte, bei der das Eingangssignal zum Ausgangssignal gemischt wird. Bei allen anderen Karten wird der Ausgang zum Eingang geschaltet, d.h. ein ggf. vorhandenes Wiedergabesignal ist nicht zu hören. Nur bei der ARC88 kann der linke Kanal getrennt vom rechten Kanal am Ausgang abgehört werden, alle anderen Karten können dies nur als Stereopaar, wobei die Mute-Information für den linken Kanal auch für den rechten Kanal herangezogen wird.

ARC88: Beachten Sie, dass jede routebare Eingangsquelle (siehe WIDM_SETROUTING im Kapitel 5.3.3) seinen eigenen Mutestatus hat. Das heißt, dass Sie durch *WIDM_MON_SETMUTE* den Mutestatus der gerade gerouteten Eingangsquelle verändern.

uMessage	dwParam1	dwParam2
WIDM_MON_SETMUTE	32bit Wert, dessen Inhalt wie folgt ausgewertet wird: Lower WORD = 0 = Linker Kanal An Lower WORD = 1 = Linker Kanal Aus Higher WORD = 0 = Rechter Kanal An Higher WORD = 1 = Rechter Kanal Aus	unbenutzt

5.2 Levelerfassung

5.2.1 WIDM_ENABLELEVEL

Verfügbarkeit: Alle MARIAN Aufnahmegeräte

Mit dieser Funktion können Sie das Erfassen und Auswerten der Levelinformationen von Audiodaten, die an dem Eingang eines Aufnahmegerätes anliegen, einschalten. Das Einschalten dieser Funktion hat allerdings eine Herabsetzung der Systemperformance zu Folge.

uMessage	dwParam1	dwParam2
WIDM_ENABLELEVEL	Unbenutzt	unbenutzt

5.2.2 WIDM_DISABLELEVEL

Verfügbarkeit: Alle MARIAN Aufnahmegeräte

Mit dieser Funktion können Sie das Erfassen und Auswerten der Levelinformationen von Audiodaten, die an dem Eingang eines Aufnahmegerätes anliegen, abschalten.

uMessage	dwParam1	dwParam2
WIDM_DISABLELEVEL	Unbenutzt	unbenutzt

5.2.3 WIDM_GETLEVEL

Verfügbarkeit: Alle MARIAN Aufnahmegeräte

Mittels dieser Funktion holen Sie sich den zuletzt gespeicherten Spitzenwert der Signalamplitude ab. Danach setzt der Treiber diesen internen Wert auf 0 und setzt die Erfassung von Spitzenwerten fort. Sie müssen zuerst die Funktion WIDM_ENABLELEVEL aufrufen, um diese Funktion nutzen zu können.

uMessage	dwParam1	dwParam2
WIDM_GETLEVEL	Pointer auf Speicherbereich, der 2 aufeinanderfolgende 32 Bit Werte, speichern kann und wie folgt gefüllt wird: Lower DWORD = PEAK linker Kanal Higher DWORD = PEAK rechter Kanal	unbenutzt

5.3 Routing

5.3.1 INPROUTING Formate

Die Funktionen WIDM_GETROUTING und WIDM_SETROUTING (Kapitel 0 und 5.3.3) erwarten einen DWORD Parameter, der das entsprechende Eingangs-Routing festlegt. Dieser Parameter ist jedoch gerätespezifisch und soll im Folgenden erläutert werden.

5.3.1.1 INPROUTING ARC 88

```
typedef struct {
    BYTE     routeLeft;
    BOOLEAN  MuteWaveOnMon;
    BYTE     routeRight;
    BYTE     MonMode;
} INPROUTING, *PINPROUTING;
```

routeLeft,
routeRight

legen die Signalquelle für den linken bzw. rechten Kanal des Aufnahmegerätes fest.
Wertebereich:

```
#define iroute_Line 0   // Eingangsquelle ist Cinch mit Linepegel
#define iroute_Mic  1   // Eingangsquelle ist Cinch mit Mikrofonpegel
#define iroute_CD   2   // Eingangsquelle ist internal CD connector
                        // Nur gültig für Aufnahmegerät 3-4
```

MuteWaveOnMon

bestimmt, ob im Falle von Monitoring (siehe WIDM_MON_SETMUTE im Kapitel 5.1.8) das Eingangssignal zum Ausgang gemischt wird oder der Ausgang ausschließlich das Eingangssignal wiedergibt.
Wertebereich:

```
TRUE                // Wiedergabesignal wird beim Monitoring abgeschaltet
FALSE               // Monitorsignal wird zum Wiedergabesignal gemischt.
```

MonMode

legt fest, wann das Monitoring des Eingangs am Ausgang stattfindet bzw. zugelassen wird.
Wertebereich:

```
#define monNever   0 // Sämtliches Monitoring deaktiviert
#define monOnRec   1 // Automatisches Monitoring bei Aufnahme aktiviert
#define monOnPunch 2 // Monitoring bei PunchIn/PunchOut zugelassen
```

5.3.1.2 INPROUTING Prodif 96 PRO, Prodif PLUS, MARC 2 und Marc 2 PRO

```
typedef struct {
  BYTE     Route;
  BYTE     MonRoute;
  BYTE     MonMode;
  BYTE     reserved;
} INPROUTING, *PINPROUTING;
```

Route

legt die Signalquelle für das digitale Aufnahmegerät fest. Es wird bei Anwendung auf das analoge Aufnahmegerät ignoriert.
Wertebereich:

```
#define iroute_Nothing    0   // Eingangsquelle abgeschaltet
#define iroute_Optical    1   // Eingangsquelle ist optischer Eingang
#define iroute_Electrical 2   // Eingansquelle ist elektrischer Eingang
#define iroute_CD         3   // Eingangsquelle ist internal CD connector
                              // Nicht anwendbar auf Prodif PLUS
```

Route Marc 2 PRO

```
#define iroute_Nothing    0   // Eingangsquelle abgeschaltet
#define iroute_Optical    1   // Eingangsquelle ist optischer Eingang
#define iroute_AesEbu     2   // Eingansquelle ist XLR AES/EBU
#define iroute_CD         3   // Eingangsquelle ist internal CD connector
#define iroute_Spdif      4   // Eingangsquelle ist Cinch S/PDIF Eingang
```

MonRoute

legt den Monitorausgang für das Aufnahmegerät fest.
Wertebereich:

```
#define mroute_Nothing    0   // Kein Monitorausgang
#define mroute_AnaOut     1   // Analog Out ist Monitorausgang
#define mroute_DigOut     2   // DigitalOut ist Monitorausgang
```

MonMode

legt fest, wann das Monitoring des Eingangs am Ausgang stattfindet bzw. zugelassen wird.

```
#define monNever    0   // Sämtliches Monitoring deaktiviert
#define monOnRec    1   // Automatisches Monitoring bei Aufnahme aktiviert
#define monOnPunch  2   // Monitoring bei PunchIn/PunchOut zugelassen
```

5.3.1.3 INPROUTING Marc 4 Digi

```
typedef struct {
  BYTE     Route;
  BYTE     MonRoute;
  BYTE     MonMode;
  BYTE     reserved;
} INPROUTING, *PINPROUTING;
```

Route

legt die Signalquelle für die Aufnahmegeräte fest.
Wertebereich Digital Eingang:

```
#define iroute_Nothing    0   // Eingangsquelle abgeschaltet
#define iroute_Optical    1   // Eingangsquelle ist optischer Eingang
#define iroute_AesEbu     2   // Eingansquelle ist XLR AES/EBU
#define iroute_DigCD      3   // Eingangsquelle ist internal Digital CD
#define iroute_Spdif      4   // Eingangsquelle ist Cinch S/PDIF Eingang
```

Wertebereich Analog Eingang:

```
#define iroute_Nothing    0   // Eingangsquelle abgeschaltet
#define iroute_AnaCD      5   // Eingangsquelle ist internal Analog CD
#define iroute_Jack       6   // Eingangsquelle ist Line-Klinken-Eingang
```

MonRoute

legt den Monitorausgang für das Aufnahmegerät fest.
Wertebereich:

```
#define mroute_Nothing    0   // Kein Monitorausgang
#define mroute_AnaOut12   1   // Analog Out 1-2 ist Monitorausgang
#define mroute_AnaOut34   2   // Analog Out 3-4 ist Monitorausgang
#define mroute_DigOut     3   // Digital Out ist Monitorausgang
```

MonMode

legt fest, wann das Monitoring des Eingangs am Ausgang stattfindet bzw. zugelassen wird.

```
#define monNever    0   // Sämtliches Monitoring deaktiviert
#define monOnRec    1   // Automatisches Monitoring bei Aufnahme aktiviert
#define monOnPunch  2   // Monitoring bei PunchIn/PunchOut zugelassen
```

5.3.2 WIDM_GETROUTING

Verfügbarkeit: ARC88 Analog, Prodif's außer ADAT Inputs, MARC 2, MARC 2 PRO, MARC 4 Digi

Durch diese Funktion erhalten Sie die eingestellte Signalquelle für das Aufnahmegerät.

uMessage	dwParam1	dwParam2
WIDM_GETROUTING	Pointer auf 32bit Wert der hardwarespezifisch interpretiert werden muss: ARC88, Prodif's und MARC 2/PRO, MARC 4 Digi	unbenutzt

5.3.3 WIDM_SETROUTING

Verfügbarkeit: ARC88 Analog, Prodif's außer ADAT Inputs, MARC 2, MARC 2 PRO, MARC 4 Digi

Durch diese Funktion bestimmen Sie die Signalquelle für das Aufnahmegerät.

uMessage	dwParam1	dwParam2
WIDM_SETROUTING	32bit Wert der hardwarespezifisch interpretiert wird. ARC88, Prodif's und MARC 2/PRO, MARC 4 Digi	unbenutzt

5.4 Aufnahmeformat

5.4.1 WIDM_GETSAMPLERATE

Verfügbarkeit: Alle MARIAN Aufnahmegeräte

Diese Funktion erlaubt Ihnen das Abfragen der aktuellen Samplerate, mit der das Aufnahmegerät läuft. Normalerweise ist dies die Samplerate, mit der das Aufnahmegerät durch *waveInOpen* geöffnet wurde. Wird das Aufnahmegerät jedoch durch eine andere als die interne Clock gesteuert (z.B. externe Wordclock oder Clock des Digital Input) so wird die Samplerate dieser Clock zurückgegeben.

uMessage	dwParam1	dwParam2
WIDM_GETSAMPLERATE	Pointer auf 32 Bit Wert der mit der aktuellen Samplerate gefüllt wird	unbenutzt

5.4.2 WIDM_GETFORMAT

Nicht dokumentiert.

5.4.3 WIDM_SETFORMAT

Nicht dokumentiert.

5.5 Pitch

Mit den Pitchfunktionen kann auf die Samplerate der Geräte Einfluss genommen werden. Der Pitchwert dient als Multiplikator für die bei *WaveInOpen* oder *WaveOutOpen* eingestellte Samplerate. Beachten Sie, dass sich die Veränderung des Pitchwertes durch *WIDM_SETPITCH* oder w*aveOutSetPitch* immer auf alle Geräte auswirkt, die mit ein und dem selben Clock verbunden sind. Variable Clock-Setups stehen für die Prodif's, die MARC 2, die MARC 2 PRO und die MARC 4 Digi zur Verfügung und werden durch die entsprechenden Treiberoberflächen implementiert. Die ARC88, die Marc 8 Midi und die MARC 4 Midi verfügen nur über eine interne Clock, welche die Samplerate aller Geräte bestimmt.

5.5.1 WIDM_GETPITCH

Diese Funktion dient zum Abholen des eingestellten Pitchwertes und stellt das Pendant zur Standard-MME-Funktion W*aveOutGetPitch* dar. Das Format des Pitchwertes entspricht dem des Pitchwertes der Funktion *WaveOutGetPitch*.

uMessage	dwParam1	dwParam2
WIDM_GETPITCH	Pointer auf 32bit Wert der mit Pitchwert gefüllt wird.	unbenutzt

5.5.2 WIDM_SETPITCH

Diese Funktion dient zum Einstellen des Pitchwertes. Das Format des Pitchwertes entspricht dem des Pitchwertes der Funktion *WaveOutSetPitch*.

uMessage	dwParam1	dwParam2
WIDM_SETPITCH	32 bit Wert, der den Pitchwert enthält.	unbenutzt

6. Monitoring

Alle MARIAN Soundsysteme verfügen über eine hardwareunterstützte und damit latenzfreie Abhörmöglichkeit der Eingangskanäle, kurz Monitoring genannt. Die MARIAN MME Erweiterung unterscheidet zwischen drei Modi des Monitoring: dem *Immediate Monitoring*, dem *AutoRecord Monitoring* und dem *Punch Monitoring*.

Immediate Monitoring

Dieser Modus hat die sofortige Wiedergabe einer Signalquelle am gewünschten Ausgang zur Folge. Signalquellen können hardwarespezifisch Ein- oder Ausgänge sein. Realisieren sie diese Funktion über WODM_SETROUTING, beschrieben im Kapitel 4.3.

AutoRecord Monitoring

Dieser Modus hat die Wiedergabe einer Eingangssignalquelle am gewünschten Ausgang zur Folge, wenn für das entsprechende Aufnahmegerät durch *waveInStart* eine Aufnahme gestartet wird. Die Wiedergabe dieser Eingangsquelle wird beendet, wenn die Aufnahme durch *waveInStop* oder *waveInReset* beendet wird. Dieser Monitoring-Modus und der gewünschte Monitorausgang wird hardwarespezifisch durch die Funktionen WODM_SETROUTING oder WIDM_SETROUTING festgelegt.

Punch Monitoring

Dieser Modus hat die Wiedergabe einer Eingangssignalquelle am gewünschten Ausgang zur Folge, wenn für das entsprechende Aufnahmegerät eine PunchIn-Aufnahme gestartet wird. Die Wiedergabe dieser Eingangsquelle wird beendet, wenn die Aufnahme durch PunchOut beendet wird. PunchIn und PunchOut wird dem Aufnahmegerät durch den Aufruf von WIDM_MON_SETMUTE, beschrieben in Kapitel 5.1.8, signalisiert. Dieser Monitoring-Modus und der gewünschte Monitorausgang wird hardwarespezifisch durch die Funktionen WODM_SETROUTING oder WIDM_SETROUTING festgelegt.

7. Start/Stop Synchronisation

Bei der Verwendung von mehreren Aufnahme- und Wiedergabegeräten über MME besteht durch die ausschließlich getrennte Ansprachemöglichkeit das Problem des synchronen Start und Stop aller beteiligten Aufnahme- und Wiedergabegeräte. Zu diesem Zweck können die Geräte über die entsprechenden Treiberoberflächen und den *SyncBus* miteinander verbunden werden, so dass nachfolgend beschriebene Funktionalität erreicht wird. Prinzipiell gilt, dass Start- und Stopbefehle immer ALLE Aufnahme- und Wiedergabegeräte betrifft, die als Verbund konfiguriert wurden. Diese werden im Folgenden als „Verbundene Geräte“ bezeichnet.

7.1 Synchroner Wiedergabestart

Voraussetzung: alle gewünschten, verbundenen Wiedergabegeräte wurden durch *WaveOutPause* in den Pausenstatus versetzt und durch *WaveOutWrite* mit Buffers versorgt

Auszuführende Aktion: Aufruf von *WaveOutRestart* für alle verbundenen, pausierten Geräte

Resultierendes Verhalten: Beim ersten *WaveOutRestart* werden alle verbundenen Wiedergabegeräte synchron gestartet.

7.2 Synchrone Wiedergabepause und Restart

Auszuführende Aktion:

- Für Wiedergabepause: Aufruf von *WaveOutPause* für beliebiges, verbundenes Wiedergabegerät.
- Für Wiedergaberestart: Aufruf von *WaveOutRestart* für dieses Gerät.

Resultierendes Verhalten:
Alle verbundenen Wiedergabegeräte stoppen die Wiedergabe. Die aktuelle Wiedergabeposition wird beibehalten. Nach *WaveOutRestart* starten alle verbundenen Wiedergabegeräte synchron.

7.3 Synchroner Wiedergabestop

Auszuführende Aktion: Aufruf von *WaveOutReset* für alle laufenden, verbundenen Wiedergabegeräte

Resultierendes Verhalten: Alle verbundenen Wiedergabegeräte stoppen die Wiedergabe synchron beim ersten Aufruf von *WaveOutReset*. Die aktuelle Wiedergabeposition wird auf Null gesetzt.

7.4 Synchroner Aufnahmestart

Voraussetzung: alle gewünschten, verbundenen Aufnahmegeräte wurden durch *WaveInAddBuffer* mit Buffers versorgt.

Auszuführende Aktion: Aufruf von *WaveInStart* für alle verbundenen Aufnahmegeräte

Resultierendes Verhalten: Beim ersten *WaveInStart* werden alle verbundenen Aufnahmegeräte synchron gestartet.

7.5 Synchrone Aufnahmepause und Restart

Auszuführende Aktion:

- Für Aufnahmepause: Aufruf von *WaveInStop* für beliebiges, verbundenes Aufnahmegerät.
- Für Aufnahmerestart: Aufruf von *WaveInStart* für dieses Gerät.

Resultierendes Verhalten:
Alle verbundenen Aufnahmegeräte stoppen die Aufnahme. Die aktuelle Aufnahmeposition bleibt erhalten. Nach *WaveInStart* starten alle verbundenen Aufnahmegeräte synchron.

Kommentar: Durch *WaveInStop* wird der aktuelle Aufnahmebuffer an die Applikation zurückgegeben. Ein folgendes *WaveInStart* ist in Sachen Synchronität nur dann erfolgreich, wenn diesem Aufnahmegerät mindestens ein Aufnahmebuffer zur Verfügung steht.

7.6 Synchroner Aufnahmestop

Auszuführende Aktion: Aufruf von *WaveInReset* für alle laufenden, verbundenen Aufnahmegeräte durchführen.

Resultierendes Verhalten: Alle verbundenen Aufnahmegeräte stoppen die Wiedergabe synchron beim ersten Aufruf von *WaveInReset*. Die aktuelle Aufnahmeposition wird auf Null gesetzt.

7.7 Synchroner Aufnahme- und Wiedergabestart

Voraussetzung:
Alle gewünschten, verbundenen Wiedergabegeräte wurden durch *WaveOutPause* in den Pausenstatus versetzt und durch *WaveOutWrite* mit Buffers versorgt.
Alle gewünschten, verbundenen Aufnahmegeräte wurden durch *WaveInAddBuffer* mit Buffers versorgt.

Auszuführende Aktion:
- Aufruf von *WaveOutRestart* für alle verbundenen, pausierten Geräte
- Aufruf von *WaveInStart* für alle verbundenen Aufnahmegeräte

Resultierendes Verhalten: Beim ersten Aufruf eines Start-Befehls werden alle verbundenen Geräte synchron gestartet.

Kommentar: Die Reihenfolge, in der die Geräte für den Start präpariert oder gestartet werden ist gleichgültig.

7.8 Synchrone Aufnahme- und Wiedergabepause und Restart

Auszuführende Aktion:
- Aufruf von *WaveOutPause* für beliebiges, verbundenes Wiedergabegerät.
- Aufruf von *WaveOutRestart* für dieses Gerät.

Resultierendes Verhalten:
Alle verbundenen Geräte stoppen. Die aktuelle Aufnahme- und Wiedergabeposition wird beibehalten. Nach *WaveOutRestart* starten alle verbundenen Geräte synchron.

Kommentar: Die Kombination von *WaveInStop* und *WaveInStart* führt nicht zum gleichen Resultat sondern zieht nur die Pause/Restart der verbundenen Aufnahmegeräte nach sich!

7.9 Synchroner Aufnahme- und Wiedergabestop

Auszuführende Aktion:
Aufruf von *WaveOutReset* für alle laufenden, verbundenen Wiedergabegeräte
Aufruf von *WaveInReset* für alle laufenden, verbundenen Aufnahmegeräte.

Resultierendes Verhalten: Alle verbundenen Geräte stoppen synchron beim ersten Aufruf einer der Reset-Funktionen. Die aktuelle Aufnahme- und Wiedergabeposition wird auf Null gesetzt.

Kommentar: Die Reihenfolge, in der die Geräte gestoppt werden, ist gleichgültig.

8. Besonderheiten Prodif 96 PRO und Prodif PLUS

Die Prodif's arbeiten prinzipiell in zwei Betriebsarten – dem *Stereo Digital Modus* und dem *ADAT Modus*. Diese Betriebsarten schließen einander aus. Je nach Anforderungsart an den Treiber schaltet dieser die Karten in die eine oder andere Betriebsart. Die Betriebsart kann nur umgeschaltet werden, wenn keine Audioaktionen auf der Karte laufen. Ist ein Umschalten nicht möglich, so quittiert der Treiber dies beim Aufruf einer MME Funktion, die dieses Umschalten erfordert, durch den Fehlercode *mmsyserr_error*.

Stereo Digital Modus

Dieser Modus ermöglicht neben den analogen Geräten die Benutzung der Aufnahme- und Wiedergabegeräte für S/PDIF und AES/EBU. Dies ist der Standardbetriebsmode nach dem Start des Windows Betriebssystems. Eine Umschaltung von der ADAT Betriebsart in den Stereo Digital Modus erfolgt unter folgenden Bedingungen:

- Ein Stereo Digital Gerät wird durch *WaveOutOpen* oder *WaveInOpen* geöffnet.
- Als Signalquelle für den analogen Ausgang wird der digitale Ein- oder Ausgang festgelegt (siehe WODM_SETROUTING in Kapitel 4.3)
- Ein Gerät wird zum Clock des Digital Input synchronisiert.

ADAT Modus

Dieser Modus ermöglicht neben den analogen Geräten die Benutzung der ADAT Aufnahme- und Wiedergabegeräte. Eine Umschaltung vom Stereo Digital Modus in den ADAT Modus erfolgt unter folgenden Bedingungen:

- Ein ADAT Gerät wird durch *WaveOutOpen* oder *WaveInOpen* geöffnet.
- Als Signalquelle für den analogen Ausgang wird ein ADAT Eingang festgelegt (siehe WODM_SETROUTING in Kapitel 4.3)
- Clockgenerator 2 der Karten wird zum ADAT Eingang synchronisiert.

9. Kontakt zu den Entwicklern

Sollten Sie Fragen oder Probleme zur vorgestellten API haben, so wenden Sie sich bitte schriftlich an:

marian
DIGITAL AUDIO ELECTRONICS

Aribert Laschke & Andreas Hellner OHG
Eisenacher Straße 72
04155 Leipzig

Email: developers@marian.de