T-Engine Forum
Specification

TEG040-S213-01.00.00/ja
2003.9.29

標準オーディオデバイスドライバ仕様書

Number: TEF040-S213-01.00.00/ja
Title: 標準オーディオデバイスドライバ仕様書
Status: [ ] Working Draft, [ ] Final Draft for Voting, [*] Standard
Date: 2003/7/22 First Edited （作成 株式会社フェイス 相田剛志）
2003/8/5 Updated to 00.00.02 （2.7 節の説明および図を改訂）
2003/9/29 Voted

目次

1　はじめに

本仕様書では T-Engine で音声入出力を行うデバイスドライバの標準 API を定義する。

Driver は audio codec の制御を行う。Driver は最小構成として、1系統のオーディオ入力、1系統のオーディオ出力のどちらか、あるいは両方が実装されたハードウェアへの対応が想定されているが、デジタル入出力インタフェイスとアナログ音量制御ハードウェアのための拡張性も考慮されている。

ハードウェアの拡張に関しては、audio codec の増設については driver のユニットを増やすことでの対応を、また DAC, ADC を複数個実装する場合については driver のサブユニットを増やすことでの対応をそれぞれ想定している。

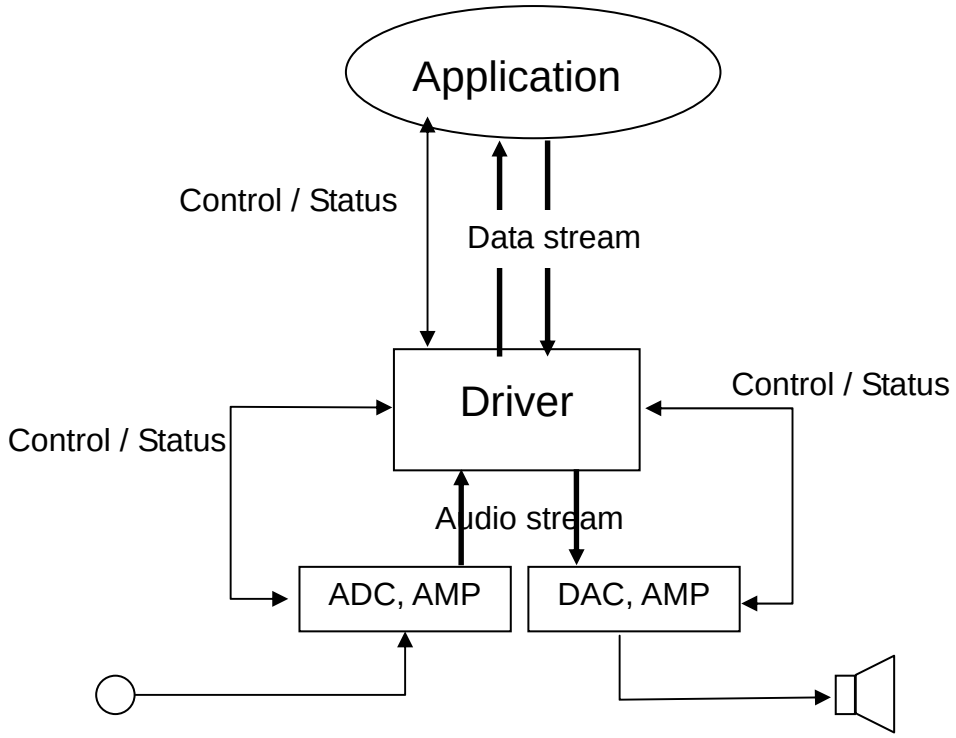

図 1　概念図

2 driver

2.1 実装上の要求事項

- オーディオデバイスの固有データのブロックサイズは AUDIO_DEVBLKSIZE[byte]と定義される。
- サブユニット ID はデバイスに実装された DAC, ADC に割り当てられる。サブユニット番号は 0 から始まる。
- 固有データについて、ドライバは必ず非同期の読み書きメカニズム tk_wri_dev(), tk_rea_dev(), tk_wai_dev() を実装する。
- 固有データへの非同期アクセス要求は、read, write **それぞれについて** AUDIO_MAXREQQ 回までキューイング可能とする。すなわち、たとえば AUDIO_MAXREQQ=2 のとき、図 2 のように write 要求がいっぱいまでキューイングされている状態でも、read 要求が発行された場合、ドライバは read 要求を待たせることなく受け付けねばならない。ただし、ハードウェアが全二重に対応していない場合はこの限りではない。

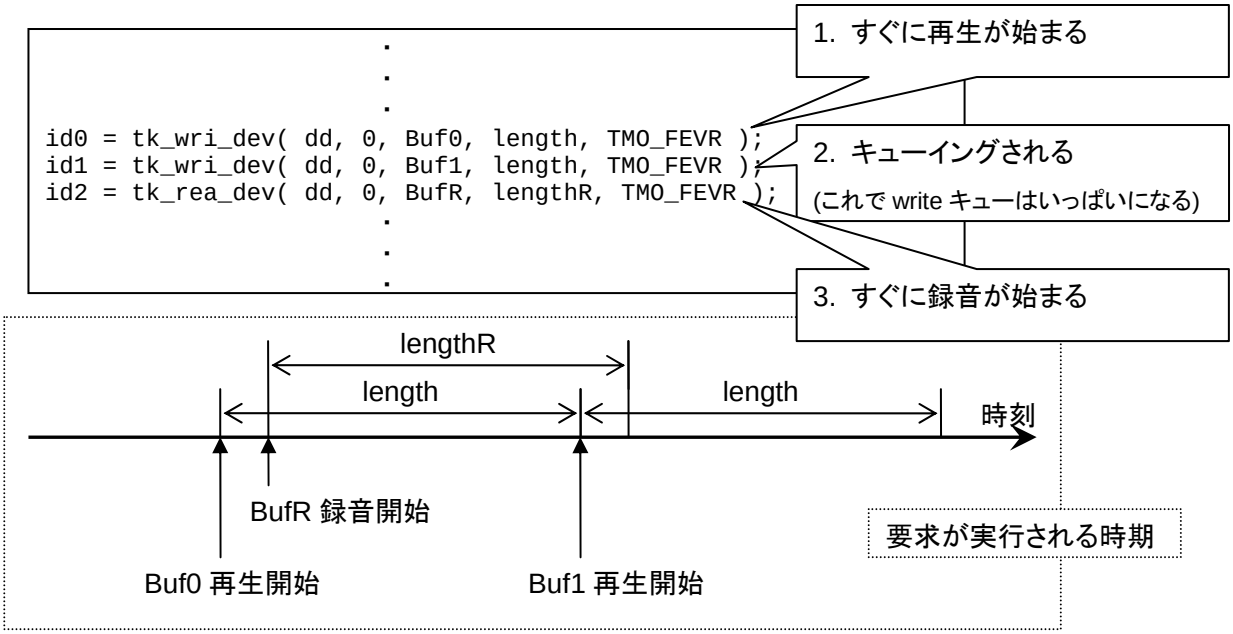

図 2 非同期アクセス要求とキューイング

- 属性データへのアクセス要求は、すべて即時に実行されなければならない。また、**属性データへのアクセスは、固有データアクセス要求のキューイング状態とは無関係に行われる。**すなわち、たとえば AUDIO_MAXREQQ=2 のとき、図 3 のように固有データの write 要求がいっぱいまでキューイングされている状態でも、属性データ DN_AUDIO_MIXERSETOUTPUTVOL の write 要求が発行された場合、ドライバは DN_AUDIO_MIXERSETOUTPUTVOL の write 要求を待たせることなく受け付けねばならない。

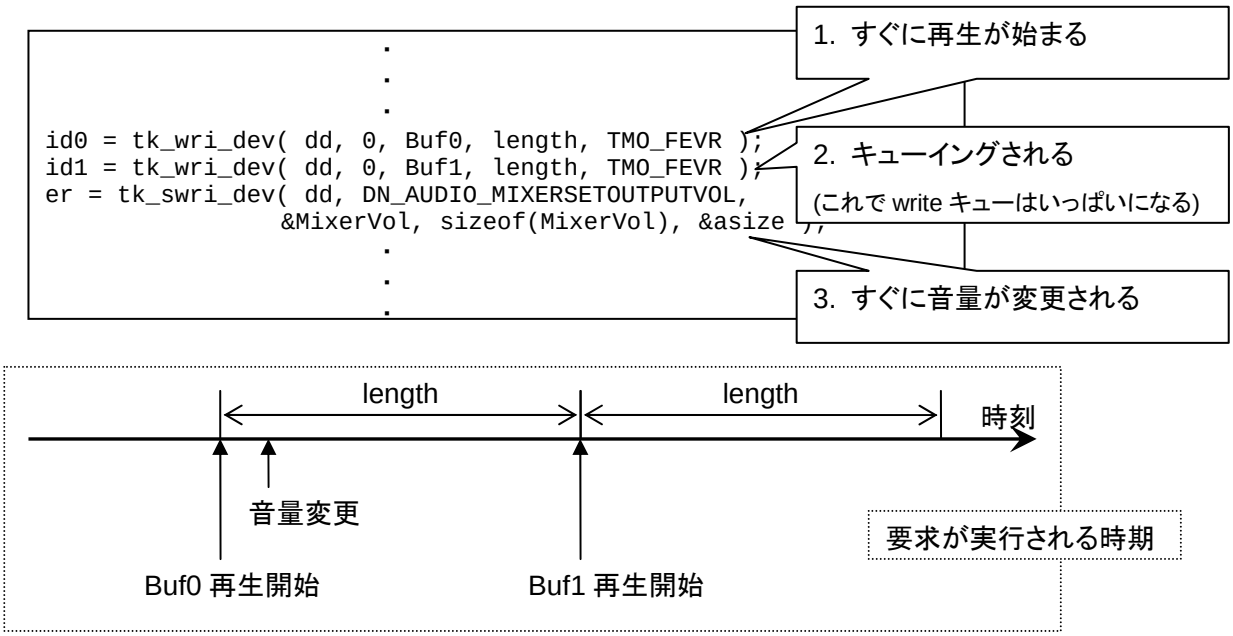

図 3 非同期アクセス要求と属性データへのアクセス

- 本文は DMA 転送を前提とした解説になっているが、DMA 転送が利用できないプラットホームである場合、転送周りの実装はプラットホーム依存とする。
- 特に記述の無い事項については、T-Engine 標準の規定に準じる。

2.2 ID tk_opn_dev(UB *devnm, UINT omode)

デバイスを確保し、待機状態にする。

引数

devnm

デバイス名文字列へのポインタ。
デバイス名は種別を”audio”とし、この後ろにユニットを表すアルファベット”a”～”z”と、サブユニットを表す数字がつくものとする。最初のユニットの、サブユニット 0 なら、デバイス名は”audioa0”である。

omode

オプション指定。

TD_WRITE	音声再生処理を許可
TD_READ	音声録音処理を許可
TD_NOLOCK	固有データバッファをドライバでロックしない

TD_WRITE と TD_READ が同時に設定できる(全二重である)かどうかはハードウェアに依存する。ただし、属性データ(`start < 0`)は omode の指定にかかわらず read, write することができる。
ドライバは tk_wri_dev()や tk_rea_dev()で固有データへとアクセスする前後に対象バッファをロック／アンロック(常駐化／非常駐化)するが、TD_NOLOCK をここで指定すると、これを行わない。

2.3 ER tk_cls_dev(ID dd, UINT option)

デバイスを停止し、解放する。
進行中の read, write は中断され、キューイングされた要求は全てキャンセルされる。
DN_AUDIO_REGISTERMSGBUF によってドライバメッセージバッファ(2.5.2 参照)が登録されていれば登録解除される。

2.4 ID tk_wai_dev(ID dd, ID reqid, INT *asize, ER *ioer, TMO tmout)

reqid で示される read, write 要求の完了を待つ。
エラーで終了した場合、*asize は不定となる。

2.5 ID tk_wri_dev(ID dd, INT start, VP buf, INT size, TMO tmout)
ER tk_swri_dev(ID dd, INT start, VP buf, INT size, INT *asize)

2.5.1 音声再生

再生するデータの先頭番地とブロック数を指定し、再生を開始する。
非同期処理で要求がキューイングされる場合、現在の要求が完了した後、ドライバは音声を途切れさせることなく次の要求の処理に入らねばならない。
いちど write 要求を発行したバッファについては、その要求の完了まではバッファの解放等をしてはならない。
DN_AUDIO_REGISTERMSGBUF によってドライバメッセージバッファ(2.5.2 参照)が登録されていれば、バッファの再生が始まる直前と、再生が終了した直後に、ドライバは通知メッセージを発行する。

データ番号(start)
DN_AUDIO_PLAYAUDIO (=0)

引数
buf
再生するデータが格納されるバッファの先頭番地。
ハードウェアプラットホームによっては次のような制限が生じる場合がある。
(制限の例)
- メモリページ単位でアライメントをとらねばならない
- タスク固有空間であってはならない

size
バッファのブロック数。

2.5.2 ドライバメッセージバッファ登録

ドライバ固有のメッセージバッファを登録する。
すでにメッセージバッファが登録済みである場合、メッセージバッファは登録されず、登録済みのメッセージバッファの ID が返る。
非同期に発行した read, write 要求は、それが実際に処理されるタイミングを呼び出し元が知ることは難しいが、ここで登録したメッセージバッファを介すれば write, read の開始および完了の通知をドライバから受けとることができる。
メッセージの発行タイミングは、なるべく再生や録音の実時間と一致していることが望ましい。
このメッセージバッファは、デフォルトでは登録されていない。
ドライバからのメッセージは AudioMsgPacket 構造体で送られる。

```
typedef struct {
   ID id;
   VP buf;
   SYSTIM otm;
} AudioMsgPacket;
#define AUDIO_MSGPKTID_WRITESTART    0x0000
#define AUDIO_MSGPKTID_WRITECOMPLETE 0x0001
#define AUDIO_MSGPKTID_READSTART     0x0002
#define AUDIO_MSGPKTID_READCOMPLETE  0x0003
```

id にはメッセージ種別が入る。Id は以下の 4 つのいずれかである。

- AUDIO_MSGPKTID_WRITESTART　　メッセージは再生開始通知である
- AUDIO_MSGPKTID_WRITECOMPLETE　　メッセージは再生完了通知である

・ AUDIO_MSGPKTID_READSTART　　メッセージは録音開始通知である
・ AUDIO_MSGPKTID_READCOMPLETE　　メッセージは録音完了通知である

buf にはメッセージ元の read, write 要求におけるバッファの先頭番地(tk_wri_dev(), tk_rea_dev() の buf)が入る。
otm にはメッセージが発行された時点のシステム稼働時間(tk_get_otm() で取得できるものと同じもの)が入る。

メッセージバッファが溢れた等の通知を行うために、ドライバには内部ステータスが存在する。
内部ステータスは「内部ドライバステータス参照(DN_AUDIO_SETSTATUS)」、「内部ドライバステータス取得(DN_AUDIO_GETSTATUS)」で読み書きできる。
メッセージバッファが一杯である場合、メッセージは発行されず、内部ステータスに AUDIO_STATUS_MBFFLOW ビットがセットされる。

データ番号(start)
DN_AUDIO_REGISTERMSGBUF

引数
buf
登録する作成済みメッセージバッファの ID を格納した ID 型へのポインタ。

size
必ず sizeof(ID)とする。

返り値
登録されたドライバメッセージバッファの ID。

2.5.3 ドライバメッセージバッファ登録解除

ドライバ固有のメッセージバッファの登録を解除する。
メッセージバッファが登録されていない場合、E_OBJ が返る。
buf および size は参照されない。

データ番号(start)
DN_AUDIO_UNREGISTERMSGBUF

返り値
登録解除されたドライバメッセージバッファの ID。

2.5.4 内部ドライバステータス設定

ドライバ固有の内部ステータスを設定する。
buf で示される UW 型変数の内容が、内部ドライバステータスにコピーされる。

```
#define AUDIO_STATUS_MBFFLOW    0x0001  // ドライバメッセージバッファが溢れた
```

データ番号(start)
DN_AUDIO_SETSTATUS

引数
buf
新しい内部ドライバステータスが格納された UW 型変数へのポインタ。

size
必ず sizeof(UW)とする。

2.5.5 出力データフォーマット指定

write するブロックのデータフォーマットを指定する。
(注 サポート可能なフォーマットは実装されるハードウェアに依存する)

データ番号(start)
DN_AUDIO_SETOUTPUTFMT

引数
buf
以下の AudioDriverDataFormat 構造体へのポインタ。

```
struct {
  W    nSize;
  W    nFormatTag;
  W    nFS;
  W    nChannels;
  W    nInterleaveSample;
 } AudioDriverDataFormat;
```

nSize
AudioDriverDataFormat 構造体のサイズ[byte]

nFormatTag
サンプルのフォーマット種別(rawPCM ならビット数、バイトオーダ、符号のあるなし)を指定する。
それぞれの組み合わせごとに特定の値を指定する。
(例)

FMT_PCM_S16_LE	rawPCM signed 16bit LittleEndian
FMT_PCM_U8	rawPCM unsigned 8bit (offset binary)

nFS
サンプリングレート[Hz]を指定する。

nChannels
データストリームのチャンネル数(>=1)を指定する。

nInterleaveSample
データストリームが nChannels>1 のとき、何サンプルごとにチャンネルが変わるかを指定する。
(例)
nChannels = 2 のとき

```
                              時刻-------------------->
  nInterleaveSample=1    L R L R L R L R L R ...
  nInterleaveSample=4    L L L L R R R R L L ...
```

size
必ず sizeof(AudioDriverDataFormat)とする。

2.5.6 入力データフォーマット指定

read するブロックのデータフォーマットを指定する。

出力データフォーマットと同様の方法で指定する。
(注 サポート可能なフォーマットは実装されるハードウェアに依存する)

データ番号(start)
DN_AUDIO_SETINPUTFMT

引数
buf
AudioDriverDataFormat 構造体へのポインタ。
解説は出力データフォーマットを参照。

size
必ず sizeof(AudioDriverDataFormat)とする。

2.5.7 出力部駆動制御 (オプション)

音声出力ハードウェア(DAC 含む)の動作状態を制御する。
デフォルトでは、動作状態は常に run である。
この機能は、DN_AUDIO_PLAYAUDIO により固有データを書き込んでから実際の発音が始まるまでの遅延時間(レイテンシ)が大きいハードウェアにおいて、最初のデータ書き込みと発音開始タイミングを個別に制御する必要があるような状況を想定している。

データ番号(start)
DN_AUDIO_SETOUTPUTSTATE

引数
buf
UW 型へのポインタ。
buf で示される UW 型では、次のように操作パラメータを規定する。

bit31	0 = stop 1 = run
bit30～16	予約 (必ず 0)
bit15～0	実装依存とする

size
必ず sizeof(UW)とする。

2.5.8 入力部駆動制御 (オプション)

音声入力ハードウェア(ADC 含む)の動作状態を制御する。
デフォルトでは、動作状態は常に run である。
この機能は、DN_AUDIO_RECAUDIO により固有データの読み込みを指示してから実際に録音が始まるまでの遅延時間(レイテンシ)が大きいハードウェアにおいて、最初のデータ読み込み指示と録音開始タイミングを個別に制御する必要があるような状況を想定している。

データ番号(start)
DN_AUDIO_SETINPUTSTATE

引数
buf
UW 型へのポインタ。
buf で示される UW 型では、次のように操作パラメータを規定する。

	bit31	0 = stop　1 = run
	bit30～16	予約 (必ず 0)
	bit15～0	実装依存とする

size

必ず sizeof(UW)とする。

2.5.9　mixer 関連機能 (オプション)

本節では write 側のミキサー関連機能を定義する。

- 本仕様書ではボリューム調節機能およびアナログミキシング機能をまとめてミキサと呼んでいる。(2.7 節参照)
- mixer 関連はサブユニット 0 でのみ制御できる。サブユニット 1 以降ではエラー(E_OBJ)になる。
- アナログミキシングハードウェアを利用できないプラットホームでは、ソフトウェアで音量制御を実装することもできるが、その場合音声出力にレイテンシが生じる場合がある。
- open 直後、mixer の状態はすべて不定である。

2.5.9.1　set output volume

指定ラインにおける出力ボリュームを制御する。

データ番号(start)

DN_AUDIO_MIXERSETOUTPUTVOL

引数

buf

以下の MixerLineVolume 構造体へのポインタ。

```
struct {
  UB    lineId;         // ライン ID
  UB    time;           // ボリューム変更に費やす時間[msec]
  H     vol[0];         // ボリューム値
} MixerLineVolume;
```

ラインは以下のような ID で指定する。

MIXER_LINEID_MASTEROUT	出力マスターボリューム
MIXER_LINEID_PCMOUT	PCM 音量
MIXER_LINEID_MICIN	マイク音量

vol[0]以降に各チャンネルのボリュームを指定する。

(例)

モノラルラインの場合

vol[0] = ボリューム

ステレオ 2 チャンネルラインの場合

vol[0] = L チャンネルのボリューム

vol[1] = R チャンネルのボリューム

ボリューム値は対数指定で、1/256 dB 単位とする。 第 n チャンネルのボリューム値は (vol[n]÷256) dB になる。

ボリューム値は、そのラインにおいて有効な範囲でクリップされる。

time には、指定ボリューム値を反映させるまでに費やす時間を指定する。即時に指定ボリューム値を反映させるときは、time に 0 を指定する。

time>0 のとき、ドライバは現在のボリュームから、vol[0]以降で指定される新しいボリューム値へと、time [msec]かけて徐々にボリュームを変化させる。この機能はクリックノイズ抑止のためのものであり、実装はオプションとする。なお、クリックノイズ抑止機能がハードウェアで

実装されている場合、time は無視される。

size
buf からの有効なバイト長。

2.5.9.2 set input volume

指定ラインにおける入力ボリューム(録音レベル)を制御する。

データ番号(start)
DN_AUDIO_MIXERSETINPUTVOL

引数
buf
MixerLineVolume 構造体へのポインタ。
解説は set output volume を参照。

size
buf からの有効なバイト長。

2.5.9.3 mute line

指定ラインの出力を mute/unmute する。
mute してもボリューム値は保存され、変更も可能であるが、unmute するまでは当該ラインの音は出ない。

データ番号(start)
DN_AUDIO_MIXERMUTELINE

引数
buf
UW 型へのポインタ。
buf で示される UW 型では、次のように操作パラメータを規定する。

bit31	0 = unmute 1 = mute
bit30～16	予約 (必ず 0)
bit15～8	ミュートに費やす時間[msec]
bit7～0	ライン ID

bit15～8 には、ミュートさせるまでに費やす時間[msec]を符号無し 8bit で指定する。即時にミュートさせるときは 0 を指定する。
(bit15～8)>0 のとき、ドライバは現在のボリュームから無音へと徐々にボリュームを変化させる。この機能はクリックノイズ抑止のためのものであり、実装はオプションとする。なお、クリックノイズ抑止機能がハードウェアで実装されている場合、bit15～8 は無視される。

size
必ず sizeof(UW)とする。

2.5.9.4 select recording source

指定ラインを録音ソースに設定する。

データ番号(start)
DN_AUDIO_MIXERSELECTRECSRC

引数

buf

以下の MixerLineRecSrc 構造体へのポインタ。

```
struct {
  W     nLines;
  UB    lineId[0];
} MixerLineRecSrc;
```

nLines
録音ソースに指定するラインの数

lineId[0]～[nLines-1]
録音ソースにするラインのライン ID

size

buf からの有効なバイト長。

2.6 ID tk_rea_dev(ID dd, INT start, VP buf, INT size, TMO tmout) ER tk_srea_dev(ID dd, INT start, VP buf, INT size, INT *asize)

2.6.1 音声録音

録音するバッファの先頭番地とブロック数を指定し、録音を開始する。
非同期処理で要求がキューイングされる場合、現在の要求が完了した後、ドライバは録音を途切れさせることなく次の要求の処理に入らねばならない。
いちど read 要求を発行したバッファについては、その要求の完了まではバッファの解放等をしてはならない。
DN_AUDIO_REGISTERMSGBUF によってドライバメッセージバッファ(2.5.2 参照)が登録されていれば、バッファへの録音が始まる直前と、録音が終了した直後に、ドライバは通知メッセージを発行する。

データ番号(start)
DN_AUDIO_RECAUDIO (=0)

引数
buf
録音するデータが格納されるバッファの先頭番地。
ハードウェアプラットホームによっては次のような制限が生じる場合がある。
(制限の例)
- メモリページ単位でアライメントをとらねばならない
- タスク固有空間であってはならない

size
バッファのブロック数。

2.6.2 サポートされているデータフォーマットの取得

デバイスユニットがサポートするデータフォーマットを列挙する。
データフォーマットの記述規則は別途定義する。
用意した配列の大きさが足りない場合、E_PAR が返る。

データ番号(start)
DN_AUDIO_GETAVAILABLEFMTS

引数
buf
B 型配列へのポインタ。
ここで示される配列に、データフォーマットを記述した半角英数字(ASCII)文字列が格納される。

size
buf で示される配列の大きさ[byte]。

2.6.3 内部ドライバステータス取得

ドライバ固有の内部ステータスを取得する。
内部ドライバステータスが、buf で示される UW 型変数にコピーされる。

内部ドライバステータスはビットごとに意味を持つので、条件判定するときは if (status & AUDIO_STATUS_MBFFLOW) などのようにビットごとに論理積をとるべきである。

```
#define AUDIO_STATUS_MBFFLOW    0x0001  // ドライバメッセージバッファが溢れた
```

データ番号(start)
DN_AUDIO_GETSTATUS

引数
buf
UW 型変数へのポインタ。

size
必ず sizeof(UW)とする。

2.6.4 現在の書き込み番地取得

録音中、バッファの現在の書き込み番地を取得する。
録音中ではない場合は E_OBJ が返る。

データ番号(start)
DN_AUDIO_GETRECORDINGPOS

引数
buf
VP 型変数へのポインタ。
ここで示される変数に、バッファの現在の書き込み番地が格納される。

size
必ず sizeof(VP)とする。

2.6.5 現在の読み出し番地取得

再生中、バッファの現在の読み出し番地を取得する。
再生中ではない場合は E_OBJ が返る。

データ番号(start)
DN_AUDIO_GETPLAYINGPOS

引数
buf
VP 型変数へのポインタ。
ここで示される変数に、バッファの現在の読み出し番地が格納される。

size
必ず sizeof(VP)とする。

2.6.6 mixer 関連機能 (オプション)

本節では read 側のアナログミキサー関連機能を定義する。
- 本仕様書ではボリューム調節機能およびアナログミキシング機能をまとめてミキサと呼んでいる。(2.7 節参照)
- mixer 関連はサブユニット 0 でのみ制御できる。サブユニット 1 以降ではエラー(E_OBJ)になる。

2.6.6.1 enumerate lines

ミキサーの諸元(サポートするライン番号、最大/最小音量、名称)、および各ラインがサポートするチャンネル数を列挙する。
lineDesc[0]～[nLines-1]が各ラインの記述子である。

データ番号(start)
DN_AUDIO_MIXERENUMLINES

引数
buf
以下の MixerAllLinesDesc 構造体へのポインタ。

```
struct {
  UB    lineId;         // ライン ID
  UB    nChannels;      // チャンネル数
  H     volMax;         // 音量の有効な最大値[1/256 dB]
  H     volMin;         // 音量の有効な最小値[1/256 dB]
  B     LineName[32];   // ラインの名称(半角英数)
} MixerLineDesc;

struct {
  W     nLines;         // ミキサーが持つライン数
  struct MixerLineDesc  LineDesc[0];
} MixerAllLinesDesc;
```

2.7 mixer ハードウェアの例 (オプション)

本仕様では、複数のアナログ入力のミキシング機能をサポートしている codec まで、想定している。その例を図 4 に示す。図において点線で囲まれた部分が mixer である。
また具体例として、ステレオ出力1系統、モノラル入力1系統をサポートする場合の mixer 部分の例を図 5 に示す。
なお、これらはあくまで本仕様のドライバがサポートし得る mixer の一例であり、ドライバの対象となるハードウェアを限定するものではない。

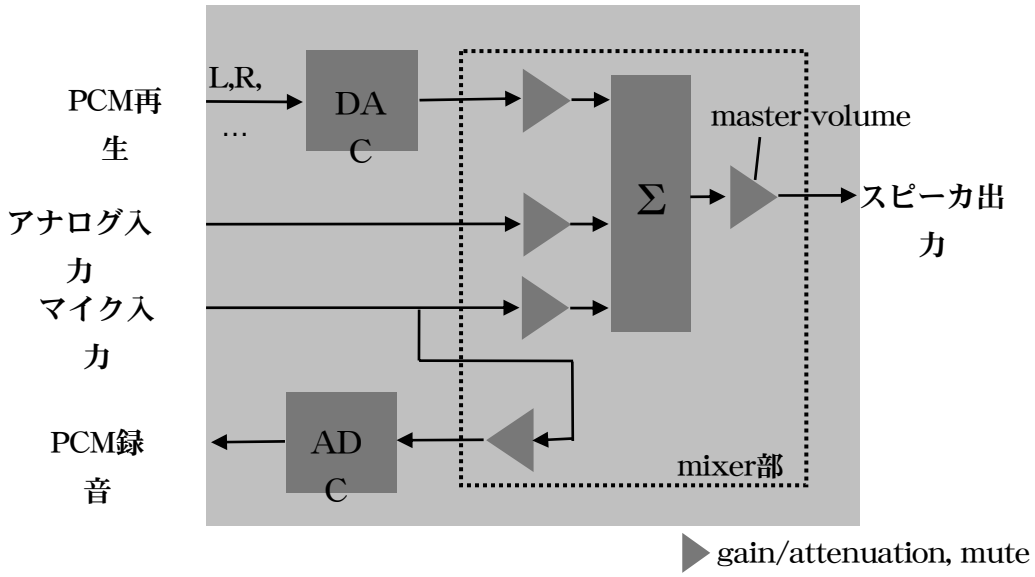

図4 本仕様書が想定している codec 例

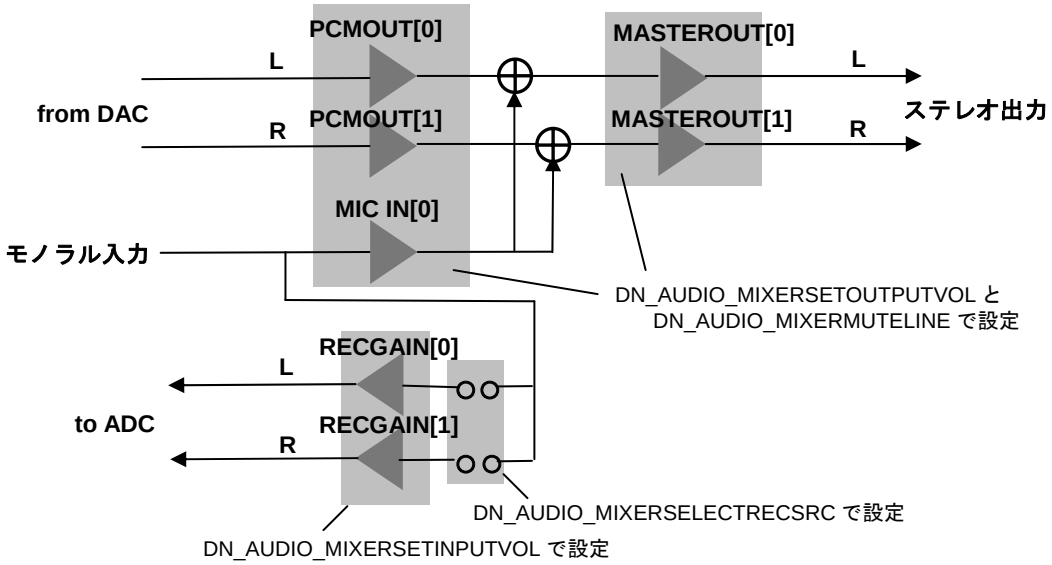

図5 ステレオ出力1系統、モノラル入力1系統の場合の mixer 例

2.8 デバイス制御のコード例(イメージ)

2.8.1 要求をキューイングしない場合

```
        .
        .
        .
// 書き込みモードでデバイスを開く
dd = tk_opn_dev( “audioa0”, TD_WRITE );
// PCM フォーマットを指定
tk_swri_dev( dd, DN_AUDIO_SETOUTPUTFMT, &pcmfmt, sizeof(pcmfmt), &asize );
// ミキサー音量を初期化
//   master volume
tk_swri_dev( dd, DN_AUDIO_MIXERSETOUTPUTVOL, &mastv, sizeof(mastv), &asize );
tk_swri_dev( dd, DN_AUDIO_MIXERMUTELINE, &mastmon, sizeof(mastmon), &asize );
//   pcmout volume
tk_swri_dev( dd, DN_AUDIO_MIXERSETOUTPUTVOL, &pcmv, sizeof(pcmv), &asize );
tk_swri_dev( dd, DN_AUDIO_MIXERMUTELINE, &pcmmon, sizeof(pcmmon), &asize );
// オーディオ再生
//   buf から length[block]再生
tk_swri_dev( dd, 0, buf, length, &asize );
// マスターボリュームをミュート(ノイズ防止)
tk_swri_dev( dd, DN_AUDIO_MIXERMUTELINE, &mastmoff, sizeof(mastmoff), &asize );
// デバイスを閉じる
tk_cls_dev( dd, 0 );
        .
        .
        .
```

2.8.2 要求をキューイングする場合

```
(ここまではキューイングしない場合と同じ)
// オーディオ再生
nBuf = 0;
id[nBuf] = tk_wri_dev( dd, 0, buf_A, length, TMO_FEVR ); // id は ID 型、長さ 2 の配列
// 終了要求があるまで繰り返し
while (!bEnd){
   // サウンドバッファを処理
   audioproc( nBuf?buf_A:buf_B );
   // 次回の要求を発行(キューイングされる)
   id[nBuf^1] = tk_wri_dev( dd, 0, nBuf?buf_A:buf_B, length, TMO_FEVR );
   // 現行の要求完了待ち
   tk_wai_dev( dd, id[nBuf], &asize, &er, TMO_FEVR );
   // バッファをスイッチ
   nBuf ^= 1;
}
tk_wai_dev( dd, id[nBuf], &asize, &er, TMO_FEVR );
        .
        .
        .
```