

Dreamcast
ミドルウェアマニュアル

ミドルウェアレコーダ外部仕様書

1999年12月22日

Ver . 1.01

変 更 履 歴

年月日	バージョン	変 更 内 容
1999.12.22	1.01	新規作成。

目 次

1. 概 要	1
1.1 目 的	1
1.2 モジュール構成	1
2. ミドルウェアライブラリの内部構成	2
3. ミドルウェアの使用方法	3
3.1 ミドルウェアライブラリの初期化	3
3.2 ミドルウェア録音ハンドル	3
3.3 ミドルウェアライブラリの使用方法	3
3.4 ミドルウェア録音ハンドルの動作状態	4
3.5 DualSpeech の録音	5
4. データ仕様	8
4.1 定 数	9
4.2 データ型	10
5. 関数仕様	11
5.1 DualSpeech 用関数	12
5.2 共通関数	14
6. 付録	18
6.1 システム構成	18
6.2 内部構成	18
6.3 SIP ライブラリとの関係	19
6.4 SIP のサンプリング周波数	19

1. 概 要

1.1 目 的

本ライブラリは、音声を容易に録音するためのライブラリである。本ライブラリにより録音された音声データは、指定のコーデックで圧縮されている。現在、対応しているコーデックは DualSpeech のみである。

1.2 モジュール構成

ミドルウェアライブラリのモジュール構成を以下に示す。

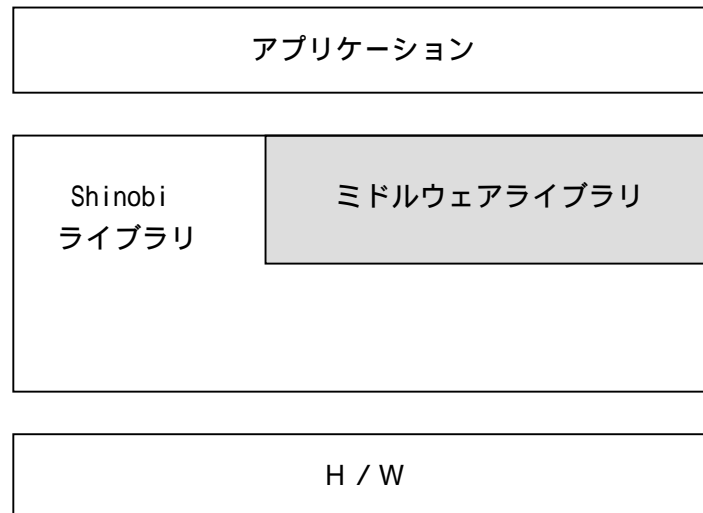


図 1.2 - 1 モジュール構成

2. ミドルウェアライブラリの内部構成

ミドルウェアライブラリは、以下のモジュールから構成されている。アプリケーションは、ミドルウェア API を用いることによって、容易に音声を録音できる。

表 2 - 1 ミドルウェアライブラリの内部モジュール

モジュール名	説 明
モジュールマネージャ	エンコーダに CPU タイムを割り当てる。
オーディオキャプチャ	音声データを読み込む。
オーディオエンコーダ	音声データを圧縮する。

以下に、ミドルウェアライブラリの内部構成を示す。

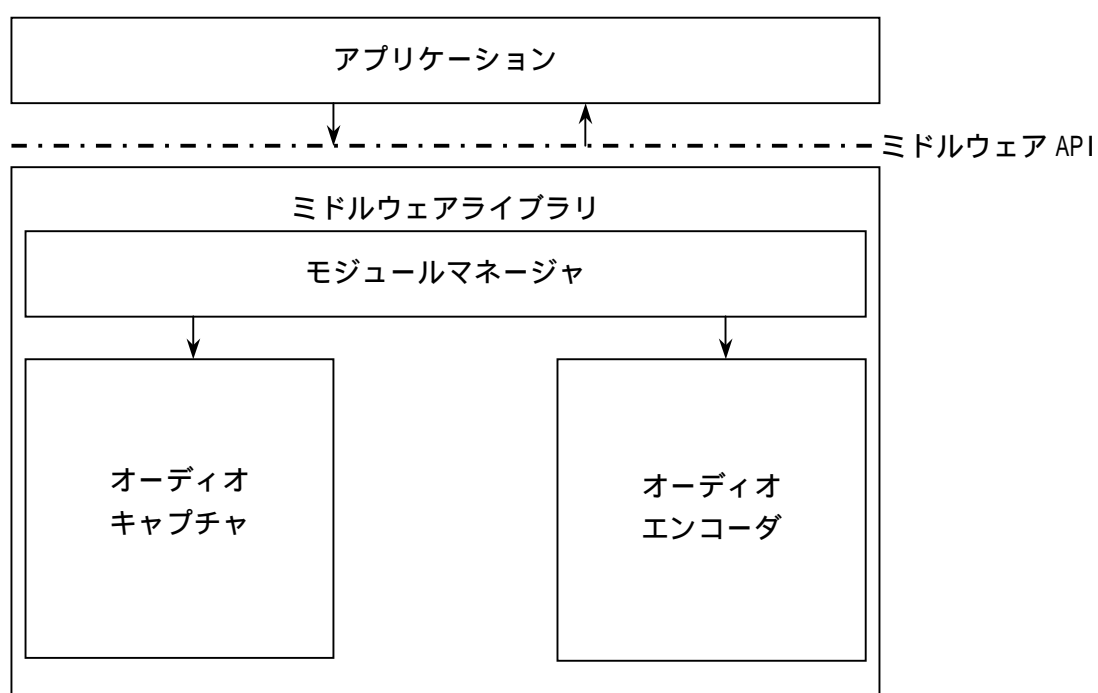


図 2 - 1 ミドルウェアライブラリの内部構成

3. ミドルウェアの使用方法

3.1 ミドルウェアライブラリの初期化

ミドルウェアライブラリの初期化は、各コーデックに対応した初期化関数を呼び出さなければならない。

<DualSpeech の場合>

```
sbInitSystem(NJD_RESOLUTION_640x480_NTSC1, NJD_FRAMEBUFFER_MODE_ARGB8888, 1);  
mwRecInitDlse(MWD_REC_SVR_VSYNC);
```

3.2 ミドルウェア録音ハンドル

各コーデックによって取り込まれた音声データは、ミドルウェア録音ハンドルを用いて録音する。まず、各コーデック用のミドルウェア録音ハンドルを生成する。このハンドルの生成には、使用するコーデックに対応した API を用いる。

```
MWREC          recdlse;  
MWS_REC_CPRM   cprm_dlse;
```

```
recdlse = mwRecCreateDlse(&cprm_dlse); /* DualSpeech データ録音用ハンドルの生成 */
```

ハンドルの生成には、異なる API を使用するが、録音指示にはコーデックによらず同じ API を使用することができる。

ストリームジョイントで録音する場合、mwRecStartSj 関数により音声の録音開始を制御する。

```
mwRecStartSj(recdlse, sj); /* DualSpeech データの録音開始 */
```

メモリ上にデータを録音する場合、mwRecStartMem 関数により音声の録音開始を制御する。

```
mwRecStartMem(recdlse, addr, len); /* DualSpeech データの録音開始 */
```

録音方法に関わらず、mwRecStop 関数により音声の録音停止を制御する。

3.3 ミドルウェアライブラリの使用方法

ミドルウェアライブラリの内部状態は、メイン処理用サーバ関数 (mwExecMainServer) を njWaitVSync 関数の前に呼び出すことで更新される。

```
while (1) {  
    /* ペリフェラル情報の取得処理 */  
    :  
    mwExecMainServer(); // エンコード処理など  
    /* 描画処理 */  
    :  
    njWaitVSync();  
}
```

3.4 ミドルウェア録音ハンドルの動作状態

ミドルウェア録音ハンドルの動作状態を下記に示す。生成した直後は、STOP 状態となる。録音開始後、状態は PREP -> RECORD -> RECEND と遷移する。マイクデバイスが接続されていない場合、ERROR に遷移する。

表 3 - 1 ハンドルの状態

状 態	説 明
STOP	録音が停止している状態
PREP	録音の準備をしている状態
RECORD	録音中の状態
RECEND	録音が終了した状態
ERROR	エラーが発生した状態

状態遷移図を以下に示す。

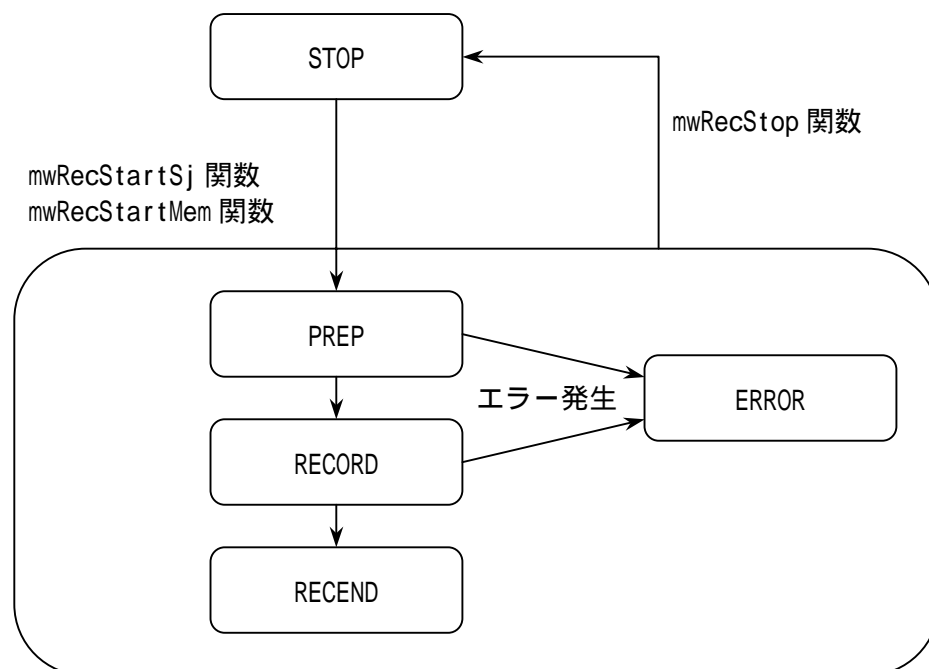


図 3.4 - 1 状態遷移図

3.5 DualSpeech の録音

以下に、DualSpeech を録音するサンプルプログラムを示す。

< ストリームジョイントを使用して録音 >

```
#define NUM_HNDL    (1)                /* 生成するハンドル数          */
#define DATA_SIZE  (2048)             /* 録音データバッファのサイズ  */

/* アプリケーションメイン関数 */
void main(void)
{
    MWREC          rec;                /* ミドルウェア録音ハンドル    */
    MWE_REC_STAT   stat;              /* ハンドルの状態              */
    MWS_REC_CPRM   cprm;              /* ハンドル生成のパラメータ構造体 */
    SJ             sj;               /* ストリームジョイントハンドル */
    SJCK           ck, ck2;          /* チャンク                    */
    Sint8          *buf;             /* ストリームジョイント作業領域 */
    Sint8          *data;            /* 録音データへのポインタ      */
    Sint32         pos;              /* 録音データ位置              */
    Sint32         len;              /* データ転送量                */

    /*
     * 画面やサウンドの初期化
     */
    mwRecInitDlse(MWD_REC_SVR_VSYNC); /* ライブラリの初期化          */
    memset(&cprm, 0, sizeof(cprm));
    cprm.size = MWD_REC_CALC_AWORK(MWD_REC_MIN_AWORK);
    cprm.buf = syMalloc(cprm.size);
    cprm.maxch = 1;
    cprm.libwork = syMalloc(MWD_DLSE_CALC_WORK(NUM_HNDL));
    cprm.extsize[MWD_CH_L] = MWD_DLSE_EXTRA_SIZE;
    cprm.extsize[MWD_CH_R] = MWD_REC_EXTRA_SIZE;
    cprm.devno = MWD_REC_DEV_A1;
    cprm.sfreq = MWD_REC_SFREQ_8KHZ;
    cprm.bps = MWD_REC_BPS_16BIT;
    cprm.gain = (MWD_REC_GAIN_MAX - MWD_REC_GAIN_MIN) / 2;
    cprm.limit = MWD_DLSE_MEMORY_LIMIT;
    rec = mwRecCreateDlse(&cprm);      /* ハンドルの生成              */
    mwRecStartSj(rec, sj);            /* 録音開始                    */
    pos = 0;
```



```

for (;;) {
    If (pos < DATA_SIZE) {
        ndata = SJ_GetNumData(sj, SJ_LIN_DATA);
        SJ_GetChunk(sj, SJ_LIN_DATA, ndata, &ck); /* データチャンクを取得 */
        len = MIN(ck.len, (size - pos));
        memcpy(data[pos], &ck.data, len);
        SJ_SplitChunk(&ck, len, &ck, &ck2);
        SJ_PutChunk(sj, SJ_LIN_DATA, &ck); /* データチャンクを渡す */
        SJ_UngetChunk(sj, SJ_LIN_FREE, &ck2); /* 空きチャンクを戻す */
        pos += len;
    }
    mwExecMainServer(); /* ミドルウェアライブラリの実行 */
    stat = mwRecGetStat(rec); /* ハンドルの状態の取得 */
    if ( stat == MWE_REC_STAT_RECEND ) {
        break;
    }
    njWaitVSync(); /* V-Sync 待ち */
}
mwRecStop(rec); /* 録音停止 */
mwRecDestroy(rec); /* ハンドルの消去 */
syFree(buf);
SJ_Destroy(sj);
syFree(cprm.libwork);
syFree(cprm.buf);
mwRecFinishDlse(); /* ライブラリの終了 */
}

```

<メモリへの録音>

```
#define NUM_HNDL    (1)                /* 生成するハンドル数          */
#define DATA_SIZE  (2048)             /* 録音データバッファのサイズ  */
/* アプリケーションメイン関数 */
void main(void)
{
    MWREC          rec;                /* ミドルウェア録音ハンドル    */
    MWE_REC_STAT    stat;              /* ハンドルの状態              */
    MWS_REC_CPRM    cprm;              /* ハンドル生成のパラメータ構造体 */
    Sint8           *data;             /* 録音データへのポインタ      */

    /*
     * 画面やサウンドの初期化
     */
    mwRecInitDlse(MWD_REC_SVR_VSYNC);  /* ライブラリの初期化          */
    memset(&cprm, 0, sizeof(cprm));
    cprm.size = MWD_REC_CALC_AWORK(MWD_REC_MIN_AWORK);
    cprm.buf = syMalloc(cprm.size);
    cprm.maxch = 1;
    cprm.libwork = syMalloc(MWD_DLSE_CALC_WORK(NUM_HNDL));
    cprm.extsize[MWD_CH_L] = MWD_DLSE_EXTRA_SIZE;
    cprm.extsize[MWD_CH_R] = MWD_REC_EXTRA_SIZE;
    cprm.devno = MWD_REC_DEV_A1;
    cprm.sfreq = MWD_REC_SFREQ_8KHZ;
    cprm.bps = MWD_REC_BPS_16BIT;
    cprm.gain = (MWD_REC_GAIN_MAX - MWD_REC_GAIN_MIN) / 2;
    cprm.limit = MWD_DLSE_MEMORY_LIMIT;
    rec = mwRecCreateDlse(&cprm);        /* ハンドルの生成              */
    mwRecStartMem(rec, (void *)data, DATA_SIZE); /* 録音開始                    */
    for (;;) {
        mwExecMainServer();            /* ミドルウェアライブラリの実行 */
        stat = mwRecGetStat(rec);      /* ハンドルの状態の取得        */
        if ( stat == MWE_REC_STAT_RECEND ) {
            break;
        }
        njWaitVSync();                /* V-Sync 待ち                  */
    }
    mwRecStop(rec);                    /* 録音停止                      */
    mwRecDestroy(rec);                /* ハンドルの消去              */
    syFree(cprm.libwork);
    syFree(cprm.buf);
    mwRecFinishDlse();                /* ライブラリの終了            */
}
```

4. データ仕様

ライブラリのデータ一覧を以下に示す。

表 4 - 1 データ一覧

データ名		機 能	番号
定 数			
	MWE_REC_STAT_~	ハンドルの状態	1.1
データ型			
	MWREC	ミドルウェア録音ハンドル	2.1
	MWS_REC_CPRM	ハンドル生成のパラメータ構造体 (DualSpeech 用)	2.2

4.1 定 数

Title データ	Data Name MWE_REC_STAT_~	Data ハンドルの状態	No 1.1
--------------	-----------------------------	-----------------	-----------

以下に示す定数は、ハンドルの状態を示す。

定数名	説 明
MWE_REC_STAT_STOP	停止中
MWE_REC_STAT_PREP	準備中
MWE_REC_STAT_RECORD	録音中
MWE_REC_STAT_RECEND	録音終了
MWE_REC_STAT_ERROR	エラー状態

4.2 データ型

Title データ	Data Name MWREC	Data ミドルウェア録音ハンドル	No 2.1
--------------	--------------------	----------------------	-----------

音声の録音を制御するためのハンドル。

Title データ	Data Name MWS_REC_CPRM	Data ハンドル生成のパラメータ構造体	No 2.2
--------------	---------------------------	-------------------------	-----------

音声コーデック（DualSpeech）用ミドルウェア録音ハンドルを生成する時に、設定するパラメータ構造体。メンバーは以下の通り。

メンバー	型 名	説 明
buf	Uint8*	使用するバッファへのポインタ MWD_REC_CALC_AWORK マクロにより計算された大きさのバッファを確保し、このメンバーに設定する。
size	Sint32	使用するバッファのサイズ（単位：バイト） MWD_REC_CALC_AWORK マクロにより計算されたバッファサイズを設定する。MWD_REC_MIN_AWORK=48 は、MWD_REC_CALC_AWORK マクロに設定する最小値で、滞りなく音声を録音するために必要なセクタ数である。
maxch	Sint32	録音する音声のチャンネル数 （DualSpeech の場合は 1 固定）
libwork	Sint32*	エンコーダが使用する作業領域 DualSpeech の録音では、MWD_DLSE_CALC_WORK マクロにより計算されたサイズの領域を確保し、このメンバーに設定する。 MWD_DLSE_CALC_WORK マクロに指定する引数には、生成するハンドルの個数を指定する。
extsize[2]	Sint32	エキストラバッファサイズ（単位：バイト） 内部で使用しているバッファにエキストラ領域を必要とする場合は、このメンバーにサイズを指定する。 （DualSpeech は指定する必要がある。） 各コーデック毎の指定方法は、関数仕様・ハンドルの生成の項を参照のこと。
devno	Sint32	マイクが接続されているデバイス番号
sfreq	Sint32	録音する音声のサンプリング周波数 （DualSpeech の場合は MWD_REC_SFREQ_8KHZ 固定）
bps	Sint32	録音する音声の量子化ビット数 （DualSpeech は MWD_REC_BPS_16BIT 固定）
gain	Sint32	マイクから音声データを取り込む際のゲイン値 下記に示すマクロの範囲のみ有効である。 MWD_REC_GAIN_MAX(16) ~ MWD_REC_GAIN_MIN(-15)
limit	Sint32	録音停止する為の限界値 メモリ上に録音する際、録音を自動停止する為の値である。 DualSpeech は MWD_DLSE_MEMORY_LIMIT を指定する。

5. 関数仕様

ライブラリの関数一覧を以下に示す。

表 5 - 1 関数一覧

関数名	機 能	番号
DualSpeech 用関数		
mwRecInitDlse	DualSpeech ライブラリの初期化	1.1
mwRecFinishDlse	DualSpeech ライブラリの終了処理	1.2
mwRecCreateDlse	DualSpeech 用ハンドルの生成	1.3
共通関数		
mwRecDestroy	ハンドルの消去	2.1
mwRecStartSj	録音開始(ストリームジョイントからの録音)	2.2
mwRecStartMem	録音開始(メモリからの録音)	2.3
mwRecStop	録音停止	2.4
mwRecGetStat	ハンドルの状態の取得	2.5
mwRecPause	ポーズの設定	2.6
mwRecEntryErrFunc	エラー発生時に呼ばれる関数の登録	2.7
mwExecMainServer	サーバ関数	2.8

5.1 DualSpeech 用関数

DualSpeech の録音に必要な関数である。

Title 関 数	Function Name	Function	No
	mwRecInitDlse	DualSpeech ライブラリの初期化	1.1

[書 式] void mwRecInitDlse(Sint32 mode);

[入 力] mode : サーバ関数呼び出しモード

MWD_REC_SVR_VSYNC : V-Sync 割込み内で、サーバ関数を実行する。

MWD_REC_SVR_MAIN : アプリケーション側で、サーバ関数を実行する。

[出 力] なし

[関数値] なし

[機 能] ライブラリを初期化する。

Title 関 数	Function Name	Function	No
	mwRecFinishDlse	DualSpeech ライブラリの終了処理	1.2

[書 式] void mwRecFinishDlse(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] ライブラリの終了処理をする。

Title 関 数	Function Name mwRecCreateDlse	Function DualSpeech 用ハンドルの生成	No 1.3
--------------	----------------------------------	---------------------------------	-----------

[書 式] MWREC mwRecCreateDlse(MWS_REC_CPRM *cprm);

[入 力] cprm : ハンドル生成のパラメータ

[出 力] なし

[関数値] ミドルウェア録音ハンドル

[機 能] ハンドルを生成する。

cprm.size には、MWD_REC_CALC_AWORK マクロにより計算された値を設定する。

MWD_REC_CALC_AWORK マクロの引数にはセクタ数を設定する。 MWD_REC_MIN_AWORK(=48) は MWD_REC_CALC_AWORK マクロに設定する最小値で、滞りなく音声を録音するために必要なセクタ数である。

cprm.buf には、cprm.size 分の領域を確保して設定する。

cprm.libwork には、MWD_DLSE_CALC_WORK マクロにより計算された値を設定する。

MWD_DLSE_CALC_WORK マクロの引数には、生成するハンドルの個数を指定する。

cprm.devno には、マイクが接続されているデバイス番号 (MWD_REC_DEV_~) を指定する。

cprm.sfreq には、音声のサンプリング周波数 (MWD_REC_SFREQ_8KHZ 固定) を指定する。

cprm.bps には、音声の量子化ビット数 (MWD_REC_BPS_16BIT 固定) を指定する。

[例] 使用例を以下に示す。

```

cprm.size = MWD_REC_CALC_AWORK(MWD_REC_MIN_AWORK);
cprm.buf = syMalloc(cprm.size));
cprm.maxch = 1;
cprm.libwork = syMalloc(MWD_DLSE_CALC_WORK(1));
cprm.extsize[MWD_CH_L] = MWD_DLSE_EXTRA_SIZE;
cprm.extsize[MWD_CH_R] = MWD_REC_EXTRA_SIZE;
cprm.devno = MWD_REC_DEV_A1;
cprm.sfreq = MWD_REC_SFREQ_8KHZ;
cprm.bps = MWD_REC_BPS_16BIT;
cprm.gain = (MWD_REC_GAIN_MAX - MWD_REC_GAIN_MIN) / 2;
cprm.limit = MWD_DLSE_MEMORY_LIMIT;
rec = mwRecCreateDlse(&cprm);

```


5.2 共通関数

コーデックの種類に関わらず共通な関数群である。

Title 関 数	Function Name	Function	No
	mwRecDestroy	ハンドルの消去	2.1

[書 式] void mwRecDestroy(MWREC rec);
[入 力] rec : ミドルウェア録音ハンドル
[出 力] なし
[関数値] なし
[機 能] ハンドルを消去する。

Title 関 数	Function Name	Function	No
	mwRecStartSj	録音開始(ストリームジョイントへの録音)	2.2

[書 式] void mwRecStartSj(MWREC rec, SJ sj);
[入 力] rec : ミドルウェア録音ハンドル
 sj : ストリームジョイントハンドル
[出 力] なし
[関数値] なし
[機 能] 外部から取り込まれた音声データをストリームジョイントへ録音を開始する。

Title 関 数	Function Name	Function	No
	mwRecStartMem	録音開始(メモリへの録音)	2.3

[書 式] void mwRecStartMem(MWREC rec, void *addr, Sint32 len);

[入 力] rec : ミドルウェア録音ハンドル

addr : 音声データを録音するバッファへのポインタ

len : バッファの長さ

[出 力] なし

[関数値] なし

[機 能] 外部から取り込まれた音声データを指定されたバッファへ録音を開始する。

Title 関 数	Function Name	Function	No
	mwRecStop	録音停止	2.4

[書 式] void mwRecStop(MWREC rec);

[入 力] rec : ミドルウェア録音ハンドル

[出 力] なし

[関数値] なし

[機 能] 音声の録音を停止する。

Title 関 数	Function Name mwRecGetStat	Function ハンドルの状態の取得	No 2.5
--------------	-------------------------------	------------------------	-----------

[書 式] MWE_REC_STAT mwRecGetStat(MWREC rec);
[入 力] rec : ミドルウェア録音ハンドル
[出 力] なし
[関数値] ハンドルの内部状態
[機 能] ミドルウェア録音ハンドルの内部状態を取得する。

定数名	説 明
MWE_REC_STAT_STOP	停止中
MWE_REC_STAT_PREP	準備中
MWE_REC_STAT_RECORD	録音中
MWE_REC_STAT_RECEND	録音終了
MWE_REC_STAT_ERROR	エラー状態

Title 関 数	Function Name mwRecPause	Function ポーズの設定	No 2.6
--------------	-----------------------------	--------------------	-----------

[書 式] void mwRecPause (MWREC rec, Sint32 sw);
[入 力] rec : ミドルウェア録音ハンドル
sw : ポーズスイッチ (0: ポーズ解除、1: ポーズ)
[出 力] なし
[関数値] なし
[機 能] ポーズの切り換えを行う。

Title 関 数	Function Name mwExecMainServer	Function サーバ関数	No 2.7
--------------	-----------------------------------	-------------------	-----------

[書 式] void mwExecMainServer(void);
[入 力] なし
[出 力] なし
[関数値] なし
[機 能] ハンドルの内部状態を更新する。

Title 関 数	Function Name mwRecEntryErrFunc	Function エラー発生時に呼ばれる関数の登録	No 2.8
--------------	------------------------------------	------------------------------	-----------

[書 式] void mwRecEntryErrFunc(MW_REC_ERRFN errfn, void *obj);
[入 力] errfn :エラー関数
obj :エラーオブジェクト
[出 力]
[関数値] なし
[機 能] エラー発生時に呼ばれる関数を登録する。
[備 考] エラー関数を登録すると、エラー発生と共に、登録した関数が呼び出される。
エラー関数は、第2引数に文字列が渡される。この文字列によりエラーの内容が確認できる。

[例] 使用例を以下に示す。

```

/* ユーザエラー関数 */
void user_error(void *user_obj, char *msg)
{
    for (;;) {
        njPrintC(NJM_LOCATION(3, 3), msg);
    }
}

void main(void)
{
    mwRecEntryErrFunc(user_error, user_obj);
}

```

6. 付録

6.1 システム構成

以下に、ミドルウェア録音ライブラリのシステム構成を示す。

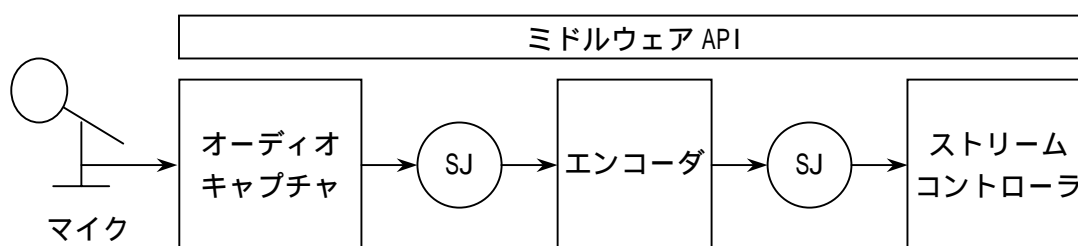


図 6.1-1 ミドルウェア再生ライブラリのシステム構成

6.2 内部構成

以下に、ミドルウェア録音ライブラリの内部構成を示す。

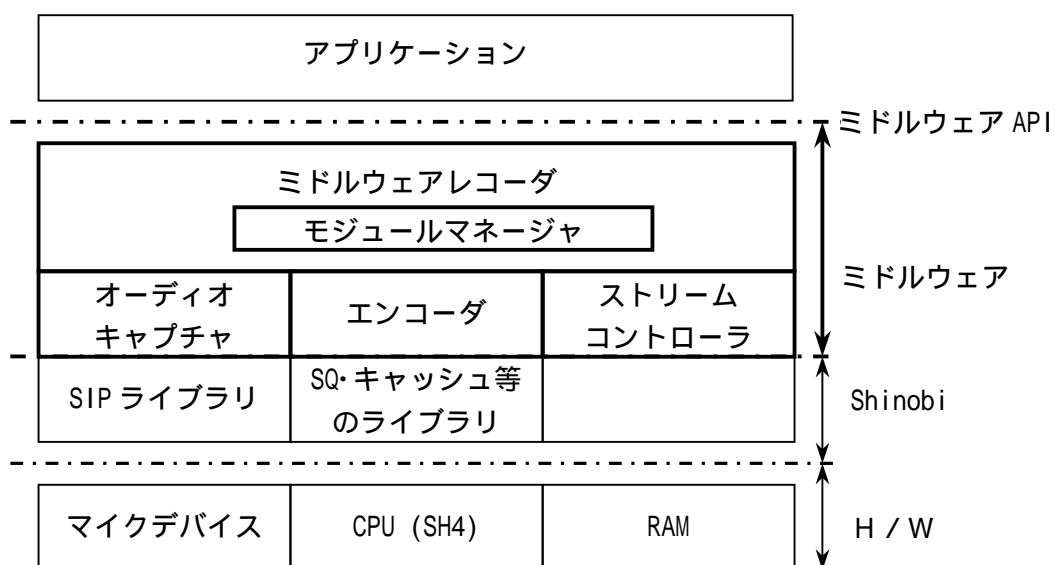


図 6.2-1 ミドルウェア録音ライブラリの内部構成

6.3 SIP ライブラリとの関係

セガ・ライブラリ環境において、ミドルウェア録音ライブラリは SIP(Sound Input Peripheral)ライブラリ上に実装されています。

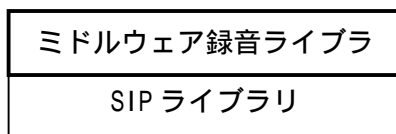


図 6.3 - 1 SIP ライブラリとの関係

6.4 SIP のサンプリング周波数

SIP にはサンプリング周波数のモードとして、8KHz と 11KHz の 2 種類があります。SIP の正確なサンプリング周波数を以下に示します。

- (1) 8KHz モード ... 8085 Hz
- (2) 11KHz モード ... 11025 Hz

SIP は上記サンプリング周波数に対して、 $\pm 0.5\%$ の個体差があります。