

# 組み込みマニュアル

## A D X再生ライブラリ



AD~~X~~

1998年12月10日	Ver5.42
1999年 3月15日	Ver5.51
1999年 6月11日	Ver5.58
1999年12月22日	Ver5.73
2000年 2月25日	Ver5.76
2000年 3月 7日	Ver5.78

## 変 更 履 歴

年月日	バージョン	変 更 内 容
1998.12.07	5.42	・新規作成。
1999. 3.15	5.51	・ポーズ機能の追加 ・他のミドルウェアとのマルチストリーム再生をサポート
1999.6.11	5.58	・ライブラリのバージョンアップに伴いバージョン番号を更新。
1999.12.22	5.73	・記述ミス・付録を追加及び修正。
2000.02.25	5.76	・シームレス連続再生機能の追加。
2000.03.07	5.78	・記述ミスを修正。

## 目 次

1.	はじめに .....	1
2.	ADX再生システム .....	2
2.1	システム構成 .....	2
2.2	ADXファイルシステム .....	4
2.3	モジュール構成 .....	5
2.4	アプリケーションインタフェース (API) .....	5
3.	ADXデータの作成方法 .....	6
4.	ライブラリ関数の使用方法 .....	8
4.1	ライブラリの初期化と終了 .....	8
4.2	ADXT ハンドルの生成 .....	8
4.3	メモリ上のADXデータの再生 (メモリ再生) .....	9
4.4	メモリ上のACXデータの再生 (メモリインデックス再生) .....	9
4.5	GD上のADXデータの再生 (GDストリーム再生) .....	10
4.6	GD上のAFSデータの再生 (GDインデックス再生) .....	11
4.7	再生状態の取得 .....	12
5.	ライブラリ関数一覧 .....	13
6.	ライブラリ関数の詳細 .....	14
付録 1.	ADXライブラリ使用時に必要なファイル .....	38
付録 2.	サウンドデータ作成上の注意 .....	39
付録 3.	データ読み込みの高速化について .....	40
付録 4.	シーク音の軽減方法 .....	41
付録 5.	MPEG Sofdec との並行再生について .....	42
付録 6.	GDFS・Ninja ライブラリとの併用について .....	43
付録 7.	ADX マルチストリームシステムの仕組み .....	44
付録 8.	SofdecF/X ムービーと音声のマルチストリーミング .....	45
付録 9.	ループ再生 .....	46
付録 10.	音声出力までのタイムラグをなくす方法 .....	47
付録 11.	クロスフェード .....	49
付録 12.	サンプルプログラムについて .....	50

## 1. はじめに

A D X再生ライブラリは、A D X圧縮された音声データを再生するためのライブラリです。本ライブラリ関数を使用することにより、圧縮された音声を簡単に再生することができます。本ライブラリは、Dreamcast 上に構成される「A D X再生システム」を制御します。

本ライブラリの特長は以下の通りです。

- ( 1 ) メインC P Uにほとんど負荷をかけずに、8 声まで再生できます。

44.1KHz の音声データデコード処理のC P U負荷

1 声	8 声
約 0.6 %	約 5 %

- ( 2 ) B G M、セリフ、効果音などの様々な音声データを再生できます。

本ライブラリで扱うデータ

データ種別	内 容
A D Xデータ	音楽やセリフなどの単一の音声ファイルを圧縮したデータ。
A C Xデータ ( A D X効果音データ )	効果音やセリフなど、複数のA D Xデータを連結して1ファイルにしたデータ。メモリインデックス再生に使用。
A F Sデータ ( A D Xファイルシステム パーティションデータ )	A D Xデータやグラフィックデータなど複数のファイルを連結したデータ。G Dインデックス再生に使用。

- ( 3 ) 様々な再生方式をサポートしています。

本ライブラリによる再生方式

再生方式	内 容
メモリ再生	メモリ上のA D Xデータを再生
メモリインデックス再生	メモリ上のA C Xデータを再生
G Dストリーム再生	G D上のA D Xデータを再生
G Dインデックス再生	G D上のA F Sデータを再生

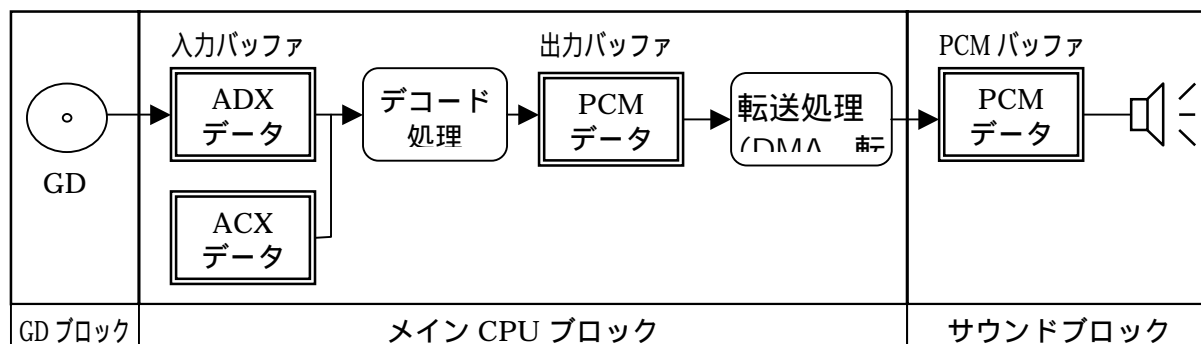
- ( 4 ) サウンドドライバと同時に使用することができます。
- ( 5 ) 複数のG Dストリーム再生、G Dインデックス再生を同時に行うことができます。
- ( 6 ) G Dからシームレスループ再生を行うことができます。
- ( 7 ) A D Xファイルシステムライブラリ ( A F S ) により、G Dストリーム再生中に、さらにG Dからゲームデータなどの別ファイルを読み込むことができます。
- ( 8 ) A F Sにより、大量のファイルを簡便に扱えます。  
( 1 ファイルあたり約 2byte で管理 )
- ( 9 ) 16ビットの非圧縮 / 4ビットADPCM圧縮形式のWAVデータファイルを再生できます。

## 2. A D X再生システム

このライブラリが制御する「A D X再生システム」について述べます。

### 2.1 システム構成

A D Xデータは、メインC P UでワークR A M上にデコードされた後、サウンドR A Mへ転送されて音声として再生されます。



#### 入力バッファ

: G Dから読み込んだA D Xデータを格納します。  
ワークR A M上にサンプリング周波数と並行再生ストリーム数に応じたサイズ分の領域を必要とします。モノラル・44KHzを3ストリーム並行再生する場合、約100Kbyteとなります。メモリから再生する場合は、必要としません。

#### 出力バッファ

: デコードされた16ビットP C Mデータを格納します。  
ワークR A M上に1チャンネル当たり(4096+32)サンプル分(4040Hbyte)の領域を必要とします。入力バッファと出力バッファの合計サイズをADXT\_CALC\_WORKマクロによって計算できます。

#### P C Mバッファ

: サウンドR A M上のP C Mデータを格納します。  
サウンドR A Mの最後から再生されるストリーム数分、サウンドライブラリによって確保されます。サウンドクリエイターは、この領域と重ならないようにマルチユニットファイルを作成する必要があります。  
サイズは、4040Hバイト/チャンネルです。

#### A D Xデータ

: A D X方式により圧縮されたデータです。  
1つの音声ファイルが1つのA D Xデータとなります。

#### A C Xデータ (A D X効果音データ)

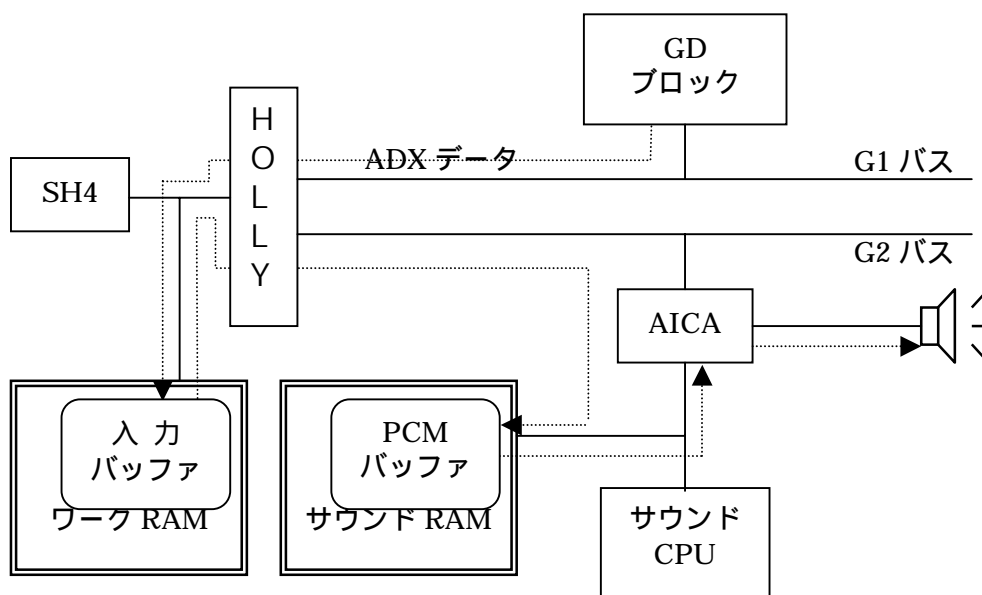
: 複数のA D Xデータを1つにまとめたデータです。  
結合時に指定したファイルリストの順番で、音声データ番号が0から割り当てられます。

#### A F Sデータ (A D Xファイルシステム パーティションデータ)

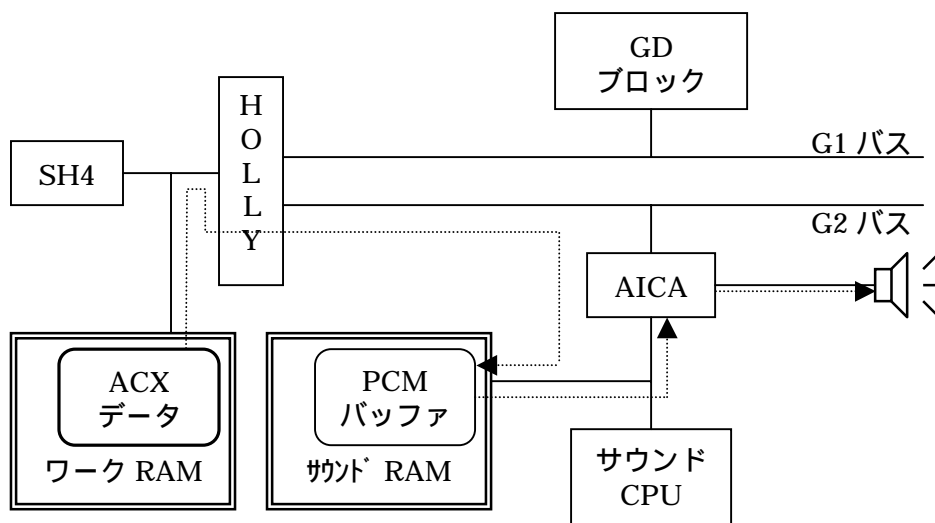
: 複数のA D Xデータやゲームデータを1つにまとめたデータです。結合時に指定したファイルリストの順番で、ファイルI Dが0から順に割り当てられます。A C Xデータとの違いはセクタバウンダリに配置されていることです。

以下に A D X データの流れを示します。

( 1 ) G D 上 の A D X データの再生 ( G D ストリーム再生、 G D インデックス再生 )



( 2 ) メモリ上の A C X データの再生 ( メモリ再生、メモリインデックス再生 )



## 2.2 A D Xファイルシステム

A D Xファイルシステムによって、大量のファイルを簡便に扱うことができます。複数のファイルをあらかじめ結合しておき、A D Xファイルシステムパーティションファイル(以下、A F Sファイル)としてG D上に格納します。A D Xファイルシステムは、結合されたA F Sファイル内から、任意のファイルを読み出すことができます。この仕組みにより、大量のセリフデータなどを簡単に再生することができます。

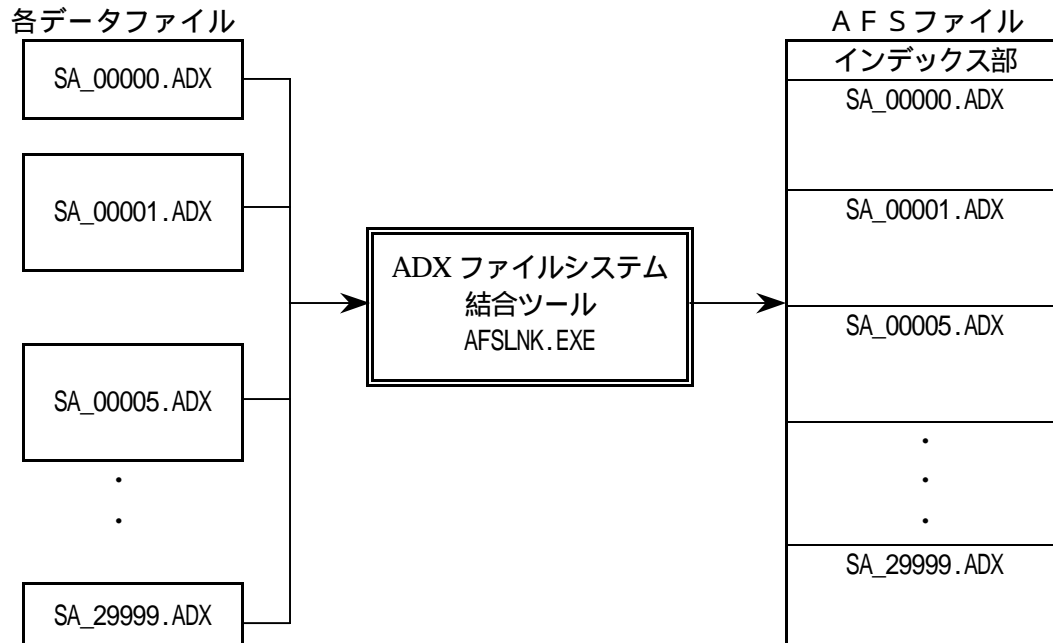


図 A F S ファイルの作成

A D Xファイルシステムのアクセスは、以下の2つのキーにより行います。

(1) パーティション I D

ユーザの決定する任意の数値(0~255)。

ADXF\_LoadPartition 関数によって、パーティション情報を読み込み、パーティション I DとA F Sファイルと関連づけます。

(2) ファイル I D

A F Sファイル内の順番号。

結合ツール(AFSLNK.EXE)は、ファイル I Dをヘッダファイルとして出力します。

A D Xファイルシステムへのアクセスは、以下の手順により行います。

(1) パーティション情報の読み込み

ADXF\_LoadPartition 関数により、パーティション情報を読み込みます。

パーティション情報は、1ファイルあたり約2byteの領域が必要です。

(2) A F Sファイル内のA D Xデータの再生

ADXT\_StartAfs 関数により、パーティション I Dとファイル I Dを指定して再生します。

(3) A F Sファイル内のデータの読み出し

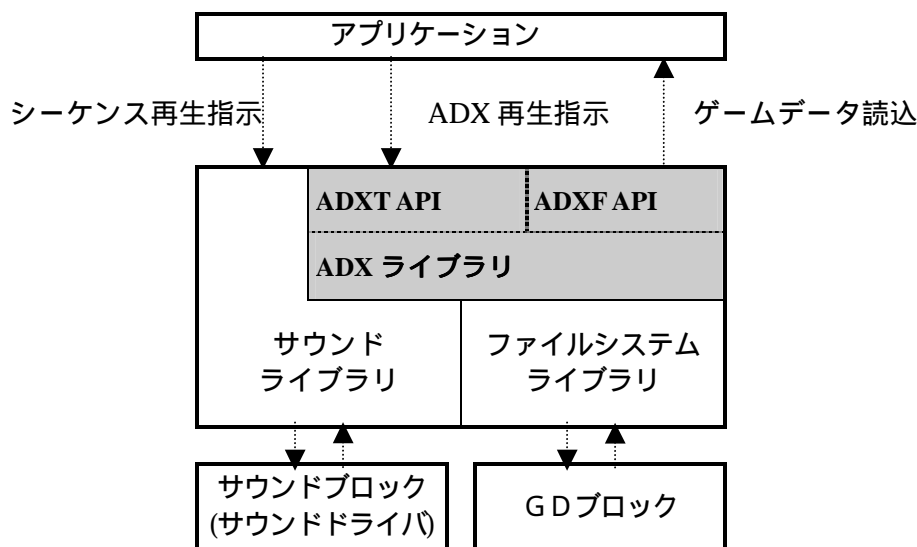
ADXF\_OpenAfs 関数により、ファイルハンドルをオープンし、ゲームデータなど読み出します。

A D Xファイルシステムは、G Dストリーム再生中でも、G D上のデータを読み出すことができます。

## 2.3 モジュール構成

A D X再生システムは、サウンドライブラリ・サウンドドライバと共存することができます。従って、プログラマは、サウンドライブラリとA D Xライブラリを同時に使用することができます。例えば、A D Xによりストリーム再生しながら、MIDI のシーケンスを再生することもできます。

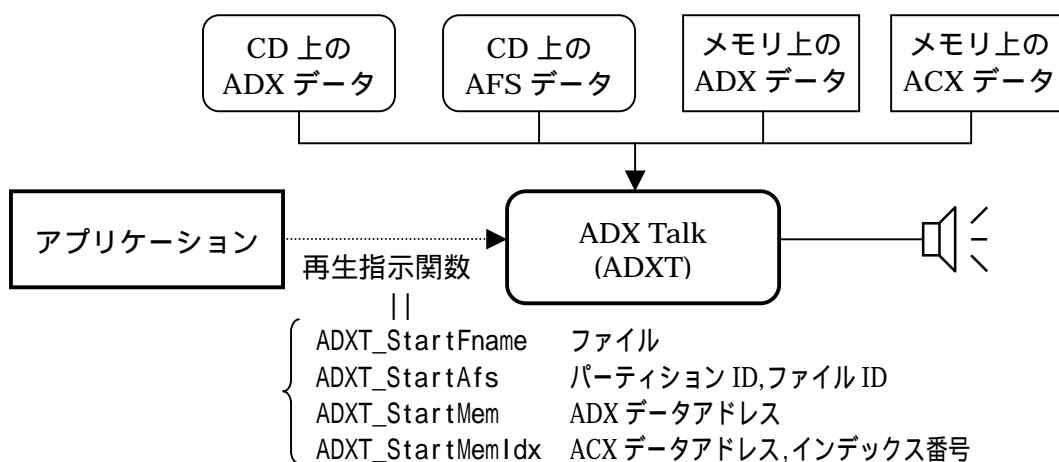
本ライブラリのモジュール構成を以下に示します。



モジュール構成図

## 2.4 アプリケーションインタフェース ( A P I )

A D XライブラリのA P Iは、オブジェクト指向的なインタフェースとなっています。まず、A D Xを再生するためのハンドルである 'ADX Talk' (ADXT) を生成します。そして、このハンドルに再生の指示を与えるだけで、G Dやメモリ上のA D Xデータを簡単に再生することができます。ハンドル1個につき、1つのA D Xデータを再生します。A D Xデータは、モノラルでもステレオでも再生することが可能です。複数のハンドルを生成することにより、同時に複数のA D Xデータを再生できます。





### 3. ADXデータの作成方法

ADXデータ、ACXデータおよびAFSデータの作成方法を示します。

#### (1) ADXデータ作成方法

ADXデータは、PCM音声データをADX圧縮したデータです。WAVまたはAIFFフォーマットの原音ファイルを、'adxencd.exe'プログラムにより圧縮します。操作方法は、以下のように引数に原音ファイル名を引数として、コマンドプロンプトより実行して下さい。カレントディレクトリに、拡張子が'ADX'になったファイルが生成されます。

```
C:¥TEMP>adxencd sample.wav 'sample.wav' が 'sample.adx' に圧縮される。
```

原音と異なるサンプリング周波数のADXデータを作成する場合は、'sf'オプションを指定して下さい。指定できるサンプリング周波数は、原音の整数分の1となります。サンプリング周波数の変換を行う場合、変換後のサンプリング周波数の0.45倍をカットオフ周波数とする、バターワースフィルタで高域を減衰させています。高域の不足を感じる場合は、事前に市販のツールでサンプリング周波数を変換してください。

```
C:¥TEMP>adxencd sample.wav -sf=22050
```

'sample.wav' をサンプリング周波数が22050Hzの'sample.adx'に圧縮する。

また、ループ再生するためには、以下のようにループスタートポイントとループエンドポイントをサンプル単位で指定します。

```
C:¥TEMP>adxencd sample.wav -lps1500 -lpe200000
```

'sample.wav' が 'sample.adx' に圧縮される。  
1500 サンプルから 199999 サンプルまでの間でループ再生する。

ループを指定する場合は、サンプリング周波数の指定は行わないで下さい。ループ時にノイズが発生する場合があります。サンプリング周波数の変換は、事前に別のツールで行ってください。

さらに、lpa オプションによってファイル全体をループ再生することができます。

```
C:¥TEMP>adxencd sample.wav -lpa
```

#### <入力音声ファイルの形式>

ファイルフォーマット	WAVまたはAIFFフォーマット
量子化ビット数	16ビットまたは8ビット
サンプリング周波数	任意

## ( 2 ) A D X データの一括作成方法

ax.exe により大量の音声データを一括エンコードできます。dir/b コマンドによりファイルリスト作成し、そのリストファイルにより音声データファイルを一括エンコードします。

```
C:¥TEMP>dir /b *.wav > wavlist.flis    WAV ファイルのリストファイルの作成  
C:¥TEMP>ax "adxencd %s" wavlist.flis    WAV ファイルを一括エンコード
```

## ( 3 ) A C X データ作成方法

A C X データは、上記の方法によって作成した複数の A D X データ ( 効果音やセリフなど ) を連結したデータです。 'adxcat.exe' プログラムにより結合します。以下のようにファイルリストを引数として与えます。

```
C:¥TEMP>dir /b *.adx > se.flis          カレントディレクトリ内の A D X ファイルを結合  
C:¥TEMP>adxcat se.flis                  した、メモリインデックス再生用 'se.acx' が  
                                         生成される。
```

## ( 4 ) A F S データ作成方法

A F S データは、A D X データやグラフィックデータなどを連結したデータです。 'adxlnk.exe' プログラムにより結合します。以下のようにファイルリストを引数として与えます。

```
C:¥TEMP>dir /b *.adx > pat01.als        カレントディレクトリ内のすべてのファイルを結合  
C:¥TEMP>afslnk pat01.als                した、'pat01.afs' が生成される。
```

カレントディレクトリ以外のディレクトリに格納されているデータを結合する場合は、以下のように指定します。

```
C:¥TEMP>dir /b pat01¥*.adx > pat01.als   pat01 ディレクトリ内のすべてのファイル  
C:¥TEMP>afslnk pat01.als -dir=pat01     を結合した、'pat01.afs' が生成される。
```

#### 4. ライブラリ関数の使用方法

初期化・メモリ上の効果音の再生・G DからのA D Xデータファイルの再生について以下に説明します。

##### 4.1 ライブラリの初期化と終了

ライブラリ全体の初期化を行うために ADXT\_Init 関数を、アプリケーションの起動時に実行してください。基本的にはゲームアプリケーションの開始時に 1 回だけ行ってください。ADXT\_Finish 関数により A D X ライブラリを終了できます。

```
/* アプリケーション */
void main()
{
    SoundInit();
    /* 転送モードを DMA モードにする */
    sdMemBlkSetTransferMode(SDE_MEMBLK_TRANSFER_MODE_DMA);
    ADXT_Init();                                /* ライブラリの初期化 */
    :
    ADXT_Finish();                              /* ライブラリの終了 */
}
```

##### 4.2 ADXT ハンドルの生成

初期化した後、作業用領域を確保し、以下のように ADXT ハンドルを生成してください。この関数によってサウンドライブラリの P C M ストリームポートのオープンなどのリソースの確保を行います。従って、シーンの最初に必要なハンドルを確保し、シーンの途中での動的な生成と消去は行わない方が C P U 負荷が安定します。

```
/* 作業用領域の大きさを決定するためのパラメータ */
#define MAX_CH          (2)          /* ステレオ (モノラルも可能) */
#define GDSTM           (1)          /* G D ストリーム再生 */
#define MAX_SFREQ       (44100)      /* サンプリング周波数 44100 Hz */

/* 作業領域サイズ */
#define WORKSIZE (ADXT_CALC_WORK(MAX_CH, ADXT_PLY_STM, MAX_GDSTM, MAX_SFREQ))
/* 作業領域 */
static char work[WORKSIZE];

void adx_play_gdstm(void)
{
    ADXT adxt;                                /* ADXT ハンドル */

    adxt = ADXT_Create(MAX_CH, work, WORKSIZE); /* ハンドルの生成 */
    :
    :
    (ゲームの 1 シーンの処理)
    :
    :
    ADXT_Destroy(adxt);                      /* ハンドルの消去 */
}
```

#### 4.3 メモリ上のADXデータの再生（メモリ再生）

メモリ上のADXデータは、ADXT\_StartMem 関数によって再生できます。ADXT ハンドルに対してADXデータのアドレスを指定して、ADXT\_StartMem 関数を実行します。メモリから再生する場合、作業領域は入力バッファを必要としないため、約 16Kbbyte×チャンネル数となります。

```
/* 作業用領域の大きさを決定するためのパラメータ */
#define MAX_CH      (2)          /* ステレオ（モノラルも可能） */
#define MAX_SFREQ   (22050)     /* サンプリング周波数 22050 Hz */

/* 作業領域サイズ */
#define WORKSIZE (ADXT_CALC_WORK(MAX_CH, ADXT_PLY_MEM, 0, MAX_SFREQ))
/* 作業領域 */
static char work[WORKSIZE];

void adx_play_mem(void)
{
    ADXT adxt;          /* ADXT ハンドル */
    void *adx=(void *)0x8c100000; /* ADX データのアドレス */

    adxt = ADXT_Create(MAX_CH, work, WORKSIZE); /* ハンドルの生成 */
    :
    ADXT_StartMem(adxt, adx); /* adx データを再生 */
    :
}
```

#### 4.4 メモリ上のACXデータの再生（メモリインデックス再生）

メモリ上のACXデータは、ADXT\_StartMemIdx 関数によって再生できます。ADXT ハンドルに対してACXデータのアドレスとインデックス番号を指定して、ADXT\_StartMemIdx 関数を実行します。

```
/* 作業用領域の大きさを決定するためのパラメータ */
#define MAX_CH      (1)          /* モノラルのみ */
#define MAX_SFREQ   (22050)     /* サンプリング周波数 22050 Hz */

/* 作業領域サイズ */
#define WORKSIZE (ADXT_CALC_WORK(MAX_CH, ADXT_PLY_MEM, 0, MAX_SFREQ))
/* 作業領域 */
static char work[WORKSIZE];

void adx_play_memidx(void)
{
    ADXT adxt;          /* ADXT ハンドル */
    void *acx=(void *)0x8c100000; /* ACX データのアドレス */

    adxt = ADXT_Create(MAX_CH, work, WORKSIZE); /* ハンドルの生成 */
    :
    ADXT_StartMemIdx(adxt, acx, no) /* acx データ内の no 番目を再生 */
    :
}
```

#### 4.5 G D上のA D Xデータの再生 ( G Dストリーム再生 )

G D上のA D Xデータは、ADXT\_StartFname 関数によって再生できます。ADXT ハンドルに対してファイル名を指定して、ADXT\_StartFname 関数を実行します。

```
/* 作業用領域の大きさを決定するためのパラメータ */
#define MAX_CH      (2)          /* ステレオ (モノラルも可能) */
#define MAX_GDSTM   (3)          /* G Dストリーム並行再生数 (3) */
#define MAX_SFREQ   (44100)      /* サンプリング周波数 44100 Hz */

/* 作業領域サイズ */
#define WORKSIZE (ADXT_CALC_WORK(MAX_CH, ADXT_PLY_STM, MAX_GDSTM, MAX_SFREQ))
/* 作業領域 */
static char work[WORKSIZE];

void adx_play_gdstm(void)
{
    ADXT adxt;                  /* ADXT ハンドル */

    adxt = ADXT_Create(MAX_CH, work, WORKSIZE); /* ハンドルの生成 */
    :
    ADXT_StartFname(adxt, "SE007.ADX"); /* "SE007.ADX" を再生 */
    :
    ADXT_StartFname(adxt, "SA019.WAV"); /* "SA019.WAV" を再生 */
    :
}
```

#### 4.6 G D上のA F Sデータの再生 ( G Dインデックス再生 )

G D上のA D Xデータは、以下の手順により再生できます。

- ( 1 ) 初期化時に ADXF\_LoadPartition 関数により、パーティション情報を読み込みます。  
この時、引数として関連づけるパーティション I D と A F S ファイル名、  
パーティション情報領域と最大ファイル数を指定します。
- ( 2 ) ADXT ハンドルに対してパーティション I D とファイル I D を指定して、  
ADXT\_StartAfs 関数を実行します。パーティション I D は ( 1 ) で定義した値、ファ  
イル I D は結合したときのファイルリスト内の順番号となります。ファイル I D の指  
定に AFSLNK.EXE の出力するヘッダファイルを利用することもできます。

```
#include "pat01.h"                /* ファイル I D の定義したヘッダファイル */
                                  /* AFSLNK.EXE が出力 */

/* パーティション I D */
#define PAT01                      (0)
/* パーティション内の最大ファイル数 */
#define MAX_PAT_NFILES             (10000)

/* 作業用領域の大きさを決定するためのパラメータ */
#define MAX_CH                     (2)          /* ステレオ (モノラルも可能) */
#define MAX_GDSTM                  (3)          /* G D ストリーム並行再生数 ( 3 ) */
#define MAX_SFREQ                   (44100)     /* サンプリング周波数 44100 Hz */
/* 作業領域サイズ */
#define WORKSIZE (ADXT_CALC_WORK(MAX_CH, ADXT_PLY_STM, MAX_GDSTM, MAX_SFREQ))

/* パーティション情報領域 */
static char ptinfo[ADXF_CALC_PTINFO_SIZE(MAX_PAT_NFILES)];
/* 作業領域 */
static char work[WORKSIZE];

void main(void)
{
    /* パーティション情報の読み込み */
    ADXF_LoadPartition(PAT01, "pat01.afs", ptinfo, MAX_PAT_NFILES);
    adx_play_afs();
}

void adx_play_afs(void)
{
    ADXT adxt;                    /* ADXT ハンドル */

    adxt = ADXT_Create(MAX_CH, work, WORKSIZE); /* ハンドルの生成 */
    :
    ADXT_StartAfs(adxt, PAT01, SE007_ADX); /* "SE007.ADX" を再生 */
                                          /* SE007_ADX は "pat01.h" 内に定義 */
}
```

#### 4.7 再生状態の取得

A D Xハンドルの状態を ADXT\_GetStat 関数によって取得することができます。この状態を監視することにより、A D Xデータの再生が終了しているか否かを確認することができます。

ADXT\_STAT\_PLAYING と ADXT\_STAT\_DECEND の時に実際に音声出力されます。

ADXT\_STAT\_PLAYEND になると再生が完了します。自動エラー回復モードが ADXT\_RMODE\_STOP の場合、GD-ROM のリードエラーなどのエラーが発生すると、ADXT\_STAT\_STOP になり音声の再生が停止します。

以下に ADXT ハンドルの状態を示します。

定 数	値	状 態	音声出力
ADXT_STAT_STOP	0	停止中	停止
ADXT_STAT_DECINFO	1	A D X のヘッダ情報取得中	停止
ADXT_STAT_PREP	2	再生準備中	停止
ADXT_STAT_PLAYING	3	デコード&再生中	発音
ADXT_STAT_DECEND	4	デコード終了	発音
ADXT_STAT_PLAYEND	5	再生終了	停止

#### < 再生状態を取得する >

```
long stat;

adxt = ADXT_Create(MAX_CH, work, WORKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartFname(adxt, "SE007.ADX"); /* "SE007.ADX" を再生 */

for (;;) {
    stat = ADXT_GetStat(adxt); /* 再生状態の取得 */
    if ( stat == ADXT_STAT_PLAYING || stat == ADXT_STAT_DECEND ) {
        /* 音声出力中の処理 */
        .
        .
    } else if ( stat == ADXT_STAT_PLAYEND ) {
        /* 再生終了時の処理 */
        .
        .
        break;
    } else if ( stat == ADXT_STAT_STOP ) {
        /* エラーにより停止 */
        .
        .
        break;
    }
}
```

## 5. ライブラリ関数一覧

A D Xライブラリの関数一覧を以下に示します。

機 能	関 数 名	番 号
ADX Talk 関数（高レベル関数）		
A D Xライブラリの初期化	ADXT_Init	6.1
A D Xライブラリの終了処理	ADXT_Finish	6.2
作業領域サイズの計算	ADXT_CALC_WORK	6.3
ADXT ハンドルの生成	ADXT_Create	6.4
ADXT ハンドルの消去	ADXT_Destroy	6.5
ファイル名指定による再生の開始	ADXT_StartFname	6.6
メモリ再生の開始	ADXT_StartMem	6.7
メモリインデックス再生の開始	ADXT_StartMemIdx	6.8
G Dインデックス再生の開始	ADXT_StartAfs	6.9
再生の停止	ADXT_Stop	6.10
状態の取得	ADXT_GetStat	6.11
サンプル単位での再生時刻の取得	ADXT_GetTime	6.12
実時間で再生時刻の取得	ADXT_GetTimeReal	6.13
総サンプル数の取得	ADXT_GetNumSmpl	6.14
サンプリング周波数の取得	ADXT_GetSfreq	6.15
出力ボリュームの設定	ADXT_SetOutVol	6.16
出力ボリュームの取得	ADXT_GetOutVol	6.17
パンポットの設定	ADXT_SetOutPan	6.18
パンポットの取得	ADXT_GetOutPan	6.19
サーバ関数の呼び出し頻度の設定	ADXT_SetSvrFreq	6.20
サーバ関数（内部状態の更新）	ADXT_ExecServer	6.21
エラーコードの取得	ADXT_GetErrCode	6.22
エラーコードのクリア	ADXT_ClearErrCode	6.23
自動エラー回復機能の設定	ADXT_SetAutoRcvr	6.24
エラーコールバック関数の設定	ADXT_EntryErrFunc	6.25
再生の一時停止・再開	ADXT_Pause	6.26
シームレス連続再生ファイルの登録	ADXT_EntryFname	6.27
シームレス連続再生の開始	ADXT_StartSeamless	6.28
シームレスループ再生の設定	ADXT_SetSeamlessLp	6.29
シームレスループ再生の開始	ADXT_StartFnameLp	6.30
シームレス連続再生の解除	ADXT_ReleaseSeamless	6.31
登録ファイル数の取得	ADXT_GetNumFiles	6.32



## 6. ライブラリ関数の詳細

Title	Function	Function Name	No
関数仕様	A D Xライブラリの初期化	ADXT_Init	6.1

[書 式] void ADXT\_Init(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] A D Xライブラリを初期化します。  
V-Sync 割り込みに、サーバ関数を登録します。

[備 考] Shinobi 環境において、サウンドライブラリに対し DMA 転送モードを指定することにより、転送負荷を軽減できます。設定方法は以下の通りです。

```
/* 通常のサウンドライブラリ・ドライバの初期化 */
SoundInit(snddrv);
/* DMA 転送モード */
sdMemBlkSetTransferMode(SDE_MEMBLK_TRANSFER_MODE_DMA);
```

[用 例]

```
SoundInit(snddrv);
sdMemBlkSetTransferMode(SDE_MEMBLK_TRANSFER_MODE_DMA);
/* A D Xライブラリの初期化 */
ADXT_Init();
```

Title	Function	Function Name	No
関数仕様	A D Xライブラリの終了処理	ADXT_Finish	6.2

[書 式] void ADXT\_Finish(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] A D Xライブラリの終了処理を行います。  
生成されているハンドルを消去し、サーバ関数を割り込み処理から削除します。

[用 例]

```
/* A D Xライブラリの初期化 */
ADXT_Init();
.
.
.
/* A D Xライブラリの終了処理 */
ADXT_Finish();
```

Title	Function	Function Name	No
マクロ仕様	作業領域の計算	ADXT_CALC_WORK	6.3

[書 式] int ADXT\_CALC\_WORK(nch, stmflg, nstm, sfreq);

[入 力] nch : 最大再生チャンネル数 (モノラル: 1, ステレオ: 2)

stmflg : 再生モード

ADXT\_PLY\_MEM: メモリ再生

ADXT\_PLY\_STM: ストリーム再生

nstm : 最大G Dストリーム数

sfreq : 最高再生サンプリング周波数

[出 力] なし

[関数値] 作業領域の大きさ [byte]

[機 能] ADXT ハンドル1つあたりの作業領域の大きさを計算します。

最大G Dストリーム数は、同時にG Dから読み込むストリーム数を指定します。

通常、同時に使用する ADXT ハンドル数と ADXF ハンドル数の総和となります。

[備 考] 作業領域サイズの計算式は、以下の通り。

作業領域 = 入力バッファサイズ + 出力バッファサイズ

< サンプリング周波数 44100Hz、モノラル >

	メモリ再生	G Dストリーム再生
入 力 バ ッ フ ア サ イ ズ	0 byte	25Kbyte +25Kbyte* 並行再生ストリーム 数
出 力 バ ッ フ ア サ イ ズ	16Kbyte	16Kbyte

1 ステレオ再生する場合は、2 倍の領域が必要。

2 入力バッファは、サンプリング周波数に比例して減らすことが可能。

Title	Function	Function Name	No
関数仕様	ADXT ハンドルの生成	ADXT_Create	6.4

[書 式] ADXT ADXT\_Create(long maxnch, void \*work, long worksize);

[入 力] maxnch : 最大再生チャンネル数 (モノラル: 1, ステレオ: 2)

work : 作業領域

worksize : 作業領域サイズ

[出 力] なし

[関数値] 生成された ADXT の構造体へのポインタ (ADXT ハンドル)

[機 能] ADXT ハンドルを生成します。

[備 考] maxnch はモノラル再生のみの場合は 1、ステレオ再生の場合は 2 を指定します。

maxnch に 2 が指定されても、モノラルも再生できます。

作業領域は、グローバル変数が syMalloc 関数によって確保します。

Title	Function	Function Name	No
関数仕様	ADXT ハンドルの消去	ADXT_Destroy	6.5

[書 式] void ADXT\_Destroy(ADXT adxt);

[入 力] adxt : ADXT ハンドル

[出 力] なし

[関数値] なし

[機 能] 指定された ADXT ハンドルを消去する。

[用 例]

```

/* 作業領域サイズ */
#define WKSIZE      ADXT_CALC_WORK(2, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE];          /* 作業領域 */
ADXT adxt;                   /* ADXT ハンドル */

adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
.
.
.
ADXT_Destroy(adxt);          /* ADXT ハンドルの消去 */

```

Title	Function	Function Name	No
関数仕様	G Dストリーム再生の開始	ADXT_StartFname	6.6

[書 式] void ADXT\_StartFname(ADXT adxt, char \*fname);

[入 力] adxt : ADXT ハンドル  
fnmae : 音声ファイル名

[出 力] なし

[関数値] なし

[機 能] fname で指定された音声ファイルの再生を開始します。すでに再生されているハンドルに対しこの関数を実行すると、再生中の音声が停止し、指定された音声ファイルが再生されます。音声ファイルとして、ADX ファイルと WAV ファイルを指定できます。異なるハンドルに対しこの関数を実行すると、同時に複数の音声を再生することができます。

[用 例]

```

/* 44KHz のステレオデータを 1 ストリームだけ再生する場合 */
#define WKSIZE ADXT_CALC_WORK(2, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE];          /* 作業領域 */
ADXT adxt;                   /* ADXT ハンドル */

adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartFname(adxt, "MUSIC.ADX"); /* 再生開始 */
.
.
ADXT_StartFname(adxt, "SE2.WAV"); /* 再生開始 */
.
.
```

Title	Function	Function Name	No
関数仕様	メモリ再生の開始	ADXT_StartMem	6.7

[書 式] void ADXT\_StartMem(ADXT adxt, void \*adxdat);

[入 力] adxt : ADXT ハンドル

adxdat : 音声データのアドレス

[出 力] なし

[関数値] なし

[機 能] adxdat で指定されたメモリ上の音声データを再生します。G Dストリーム再生よりもレスポンス良く、音声を再生できます。遅延時間は、約 50msec です。

[用 例]

```

/* モノラルデータを再生する場合 */
#define WKSIZE      ADXT_CALC_WORK(1, ADXT_PLY_MEM, 0, 0)

char *work[WKSIZE];          /* 作業領域 */
char adxdat[ADXDAT_SIZE];    /* データ領域 */
ADXT adxt;                   /* ADXT ハンドル */

load_data(adxdat);            /* 音声データの読み込み */
adxt = ADXT_Create(1, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartMem(adxt, adxdat);  /* 再生開始 */
.
.

```

Title	Function	Function Name	No
関数仕様	メモリインデックス再生の開始	ADXT_StartMem	6.8

[書 式] void ADXT\_StartMemIdx(ADXT adxt, void \*acxd, long no);

[入 力] adxt : ADXT ハンドル  
acxd : ACX データのアドレス  
no : インデックス番号

[出 力] なし

[関数値] なし

[機 能] acxd で指定された A C X データ内の no 番目のデータを再生します。

[用 例]

```

/* モノラルデータを再生する場合 */
#define WKSZ ADXT_CALC_WORK(1, ADXT_PLY_MEM, 0, 0)

char *work[WKSZ]; /* 作業領域 */
char acxd[ADXDAT_SIZE]; /* データ領域 */
ADXT adxt; /* ADXT ハンドル */

load_data(acxd); /* 音声データの読み込み */
adxt = ADXT_Create(1, work, WKSZ); /* ADXT ハンドルの生成 */
ADXT_StartMem(adxt, acxd, 5); /* 再生開始 */
.
.
ADXT_StartMem(adxt, acxd, 8); /* 再生開始 */
.
.

```

Title	Function	Function Name	No
関数仕様	G Dインデックス再生の開始	ADXT_StartAfs	6.9

[書 式] void ADXT\_StartAfs(ADXT adxt, long pat\_id, long file\_id);

[入 力] adxt : ADXT ハンドル  
pat\_id : パーティション I D  
file\_id : ファイル I D

[出 力] なし

[関数値] なし

[機 能] G D上の A F S ファイル内のデータを再生します。

ADXF\_LoadPartition 関数により、par\_id と AFS ファイル名を関連づけて

おきます。file\_id は、A F S ファイル内の順番号となります。  
AFSLNK.EXE はファイル I D を定義したヘッダファイルを出力するので、  
ファイル名に対応する定義を指定することもできます。

[用 例]

```

/* 44KHz ステレオと 22KHz モノラルを同時に再生する場合 */
/* 44KHz ステレオ再生用作業領域 */
#define WKSIZ44 ADXT_CALC_WORK(2, ADXT_PLY_STM, 2, 44100)
/* 22KHz モノラル再生用作業領域 */
#define WKSIZ22 ADXT_CALC_WORK(1, ADXT_PLY_STM, 2, 44100)
/* パーティション内の最大ファイル数 */
#define MAX_NFILES 10000

/* パーティション情報領域 */
char ptinfo[ADXF_CALC_PTINFO_SIZE(MAX_NFILES)];
char *work1[WKSIZ44]; /* 作業領域 */
char *work2[WKSIZ22]; /* 作業領域 */
ADXT adxt1, adxt2; /* ADXT ハンドル */

/* パーティション情報の読み込み */
ADXF_LoadPartition(PATID=0, "pat01.afs", ptinfo, MAX_NFILES);

adxt1 = ADXT_Create(2, work1, WKSIZ44); /* ADXT ハンドルの生成 */
adxt2 = ADXT_Create(1, work2, WKSIZ22); /* ADXT ハンドルの生成 */
ADXT_StartAfs(adxt1, PATID, BGM); /* 再生開始 */
.
.
ADXT_StartAfs(adxt1, PATID, SRF137); /* 再生開始 */
.
.
```

Title	Function	Function Name	No
関数仕様	再生の停止	ADXT_Stop	6.10

[書 式] void ADXT\_Stop(ADXT adxt);

[入 力] adxt : ADXT ハンドル

[出 力] なし

[関数値] なし

[機 能] A D X の再生を停止する。

[用 例]

```

/* 44KHz のステレオデータを 1 ストリームだけ再生する場合 */
#define WKSIZE ADXT_CALC_WORK(2, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE]; /* 作業領域 */
ADXT adxt; /* ADXT ハンドル */

adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartFname(adxt, "MUSIC.ADX"); /* 再生開始 */
.
.
/* B ボタンが押されたら停止 */
if ( per->press & PDD_DGT_TB )
    ADXT_Stop(adxt); /* 再生停止 */

```



Title	Function	Function Name	No
関数仕様	状態の取得	ADXT_GetStat	6.11

[書 式] long ADXT\_GetStat(ADXT adxt);

[入 力] adxt : ADXT ハンドル

[出 力] なし

[関数値] 現在の ADXT ハンドルの動作状態

[機 能] 現在の adxt の動作状態を取得する。  
動作状態は以下の通り。

定 義 名	定 数 値	動 作 状 態
ADXT_STAT_STOP	0	停止中
ADXT_STAT_DECINFO	1	A D X のヘッダ情報取得中
ADXT_STAT_PREP	2	再生準備中
ADXT_STAT_PLAYING	3	デコード&再生中
ADXT_STAT_DECEND	4	デコード終了
ADXT_STAT_PLAYEND	5	再生終了

#### [用 例]

```

/* 44KHz のモノラルデータを 1 ストリームだけ再生する場合 */
#define WKSIZE ADXT_CALC_WORK(1, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE];          /* 作業領域 */
ADXT adxt;                   /* ADXT ハンドル */
long stat;                    /* 動作状態 */

adxt = ADXT_Create(1, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartAfs(adxt, pat_id, 0);      /* 再生開始 */
for (;;) {
    /* 音が発生している間、アニメーション */
    stat = ADXT_GetStat(adxt);
    if ( stat == ADXT_STAT_PLAYING && stat == ADXT_STAT_DECEND )
        user_animate_obj();

    /* 再生が終了したら、次の音声を再生 */
    if ( stat == ADXT_STAT_PLAYEND && stat == ADXT_STAT_STOP )
        ADXT_StartAfs(adxt, pat_id, 1);
}

```

再生終了は、状態が PLAYEND または STOP になったときと  
して判断してください。G D リードエラーが発生すると状態は STOP に  
なります。

Title	Function	Function Name	No
関数仕様	サンプル単位での再生時刻の取得	ADXT_GetTime	6.12

[書 式] void ADXT\_GetTime(ADXT adxt, long \*ncount, long \*tscale);

[入 力] adxt : ADXT ハンドル

[出 力] ncount : 再生サンプル数

tscale : サンプリング周波数 [Hz]

[関数値] なし

[機 能] サンプル単位での再生時刻を取得します。

[用 例]

```

/* 44KHz のステレオデータを 1 ストリームだけ再生する場合 */
#define WKSIZE ADXT_CALC_WORK(2, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE];          /* 作業領域 */
ADXT adxt;                   /* ADXT ハンドル */
long nsmpl, sfreq;            /* 再生時刻 */
long nfrm, hh, mm, ss, ff;    /* 時刻 */

adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartFname(adxt, "music.adx"); /* 再生開始 */
for (;;) {
    /* 再生時刻の表示 (フレームの単位は 1/60) */
    ADXT_GetTime(adxt, &nsmpl, &sfreq);
    nfrm = nsmpl*60/sfreq; /* 総フレーム数 */
    hh = nfrm / (60*60*60); /* 時 */
    mm = (nfrm - hh*60*60*60) / (60*60); /* 分 */
    ss = (nfrm - (hh*60+mm)*60*60) / 60; /* 秒 */
    ff = nfrm % 1000; /* フレーム */
    disp_time(hh, mm, ss, ff);
}

```

44KHz の音声データを上記の例で扱う場合、nfrm を求める時に 60 を乗じているため、再生時刻の最大値は 811 秒となります。ADXT\_GetTime 関数は、最大で約 13.5 時間ですが、このように単位を変換するときは、注意が必要です。

Title	Function	Function Name	No
関数仕様	実時間での再生時刻の取得	ADXT_GetTimeReal	6.13

[書 式] long ADXT\_GetTimeReal(ADXT adxt);

[入 力] adxt : ADXT ハンドル

[出 力] なし

[関数値] 再生時刻 [1/100sec]

[機 能] 実時間で再生時刻を取得します。単位は 1/100 秒です。

[用 例]

```

/* 44KHz のステレオデータを 1 ストリームだけ再生する場合 */
#define WKSIZE ADXT_CALC_WORK(2, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE];          /* 作業領域 */
ADXT adxt;                   /* ADXT ハンドル */
long time;                    /* 再生時刻 */
long hh, mm, ss, ff;         /* 時刻 */

adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartFname(adxt, "music.adx"); /* 再生開始 */
.
for (;;) {
    /* 再生時刻の表示 (フレームの単位は 1/100) */
    time = ADXT_GetTime(adxt, &nsmpl, &sfreq);
    hh = time / (60*60*100); /* 時 */
    mm = (time - hh*60*60*100)/(60*100); /* 分 */
    ss = (time - (hh*60+mm)*60*100)/100; /* 秒 */
    ff = time % 100; /* フレーム */
    disp_time(hh, mm, ss, ff);
}

```

Title	Function	Function Name	No
関数仕様	総サンプル数の取得	ADXT_GetNumSmpl	6.14

[書 式] long ADXT\_GetNumSmpl( ADXT adxt );

[入 力] adxt : ADXT ハンドル

[出 力] なし

[関数値] 音声データの総サンプル数

[機 能] 再生中の音声データの総サンプル数を取得します。

[備 考] 再生状態が、再生準備中(ADXT\_STAT\_PREP=2)から再生終了(ADXT\_STAT\_PLAYEND=5)までの間に取得できます。

[用 例]

```

/* 44KHz のステレオデータを 1 ストリームだけ再生する場合 */
#define WKSIZE ADXT_CALC_WORK(2, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE];          /* 作業領域 */
ADXT adxt;                   /* ADXT ハンドル */
long total_nsmp;              /* 総サンプル数 */
long nsmp, sfreq;             /* 再生時刻 */
long percent;                 /* パーセント */

adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartFname(adxt, "music.adx"); /* 再生開始 */
for (;;) {
    /* 再生の進行状況の表示 (パーセント) */
    if ( ADXT_GetStat(adxt) >= ADXT_STAT_PREP ) {
        total_nsmp = ADXT_GetNumSmpl(adxt);
        ADXT_GetTime(adxt, &nsmp, &sfreq);
        percent = nsmp * 100 / total_nsmp;
    } else {
        percent = 0;
    }
    disp_progress(percent);
}

```

Title	Function	Function Name	No
関数仕様	サンプリング周波数の取得	ADXT_GetSfreq	6.15

[書 式] long ADXT\_GetSfreq(ADXT adxt);

[入 力] adxt : ADXT ハンドル

[出 力] なし

[関数値] 音声データのサンプリング周波数 [Hz]

[機 能] 再生中の音声データのサンプリング周波数を取得します。

[備 考] 再生状態が、再生準備中(ADXT\_STAT\_PREP=2)から再生終了(ADXT\_STAT\_PLAYEND=5)までの間に取得できます。

[用 例]

```

/* 44KHz のステレオデータを 1 ストリームだけ再生する場合 */
#define WKSIZE ADXT_CALC_WORK(2, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE];          /* 作業領域 */
ADXT adxt;                   /* ADXT ハンドル */
long sfreq;                   /* サンプリング周波数 */

adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartFname(adxt, "music.adx"); /* 再生開始 */
for (;;) {
    /* 再生周波数の表示 (Hz) */
    if ( ADXT_GetStat(adxt) >= ADXT_STAT_PREP ) {
        sfreq = ADXT_GetSfreq(adxt);
        disp_sampling_frequecy(sfreq);
    }
}

```

Title	Function	Function Name	No
関数仕様	出力ボリュームの設定	ADXT_SetOutVol	6.16

[書 式] void ADXT\_SetOutVol(ADXT adxt, long vol);

[入 力] adxt : ADXT ハンドル  
vol : 減衰レベル (0:-0dB ~ -999:-99.9dB)

[出 力] なし

[関数値] なし

[機 能] 出力ボリュームを設定します。再生前でも再生中でも設定することができます。

vol の設定値 0 : -0dB 減衰なし (最大)  
-30 : -3dB 約 70%  
-60 : -6dB 約 50%  
-999 : -99.9dB ミュート

Title	Function	Function Name	No
関数仕様	出力ボリュームの取得	ADXT_GetOutVol	6.17

[書 式] void ADXT\_GetOutVol(ADXT adxt);

[入 力] adxt : ADXT ハンドル

[出 力] なし

[関数値] 出力ボリュームの設定値 (0:-0dB ~ -999:-99.9dB)

[機 能] 出力ボリュームを取得します。

[用 例]

```

/* 44KHz のステレオデータを 1 ストリームだけ再生する場合 */
#define WKSIZE ADXT_CALC_WORK(2, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE]; /* 作業領域 */
ADXT adxt; /* ADXT ハンドル */
long vol; /* 現在のボリューム */
long fade_vol; /* フェードの変化量 */

fade_vol = 1000/180; /* 3 秒でフェード */
adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartFname(adxt, "music.adx"); /* 再生開始 */
for (;;) {
    /* フェードイン */
    if ( per->on & PDD_DGT_TA ) {
        vol = ADXT_GetOutVol(adxt);
        ADXT_SetOutVol(adxt, vol+fade_vol);
    }
    /* フェードアウト */
    if ( per->on & PDD_DGT_TB ) {
        vol = ADXT_GetOutVol(adxt);
        ADXT_SetOutVol(adxt, vol-fade_vol);
    }
}

```

Title	Function	Function Name	No
関数仕様	パンポットの設定	ADXT_SetOutPan	6.18

[書 式] void ADXT\_SetOutPan(ADXT adxt, long ch, long pan);

[入 力] adxt : ADXT ハンドル  
ch : チャンネル番号 (0, 1)  
ADXT\_CH\_L(0):左チャンネル, ADXT\_CH\_R(1):右チャンネル  
pan : パンポット設定値 (-15~+15, -128)  
ADXT\_PAN\_LEFT =-15, ADXT\_PAN\_CENTER= 0  
ADXT\_PAN\_RIGHT= 15, ADXT\_PAN\_AUTO =-128

[出 力] なし

[関数値] なし

[機 能] 出力パンポットを設定します。  
AUTO の場合は、音声データがモノラルかステレオかによって自動的にパンが設定されます。音声を再生中でも設定することができます。

Title	Function	Function Name	No
関数仕様	パンポットの取得	ADXT_GetOutPan	6.19

[書 式] long ADXT\_GetOutPan(ADXT adxt, long ch);

[入 力] adxt : ADXT ハンドル  
ch : チャンネル番号  
ADXT\_CH\_L(0) : 左チャンネル, ADXT\_CH\_R(1) : 右チャンネル

[出 力] なし

[関数値] パンポットの設定値 (-15~+15, -128)

[機 能] 出力パンポットを取得します。

[用 例]

```

/* 44KHz のモノラルデータを1ストリームだけ再生する場合 */
#define WKSIZE ADXT_CALC_WORK(1, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE]; /* 作業領域 */
ADXT adxt; /* ADXT ハンドル */
long pan; /* 現在のパンポット値 */

adxt = ADXT_Create(1, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartFname(adxt, "srf01.adx"); /* 再生開始 */
for (;;) {
    /* パンを右へ移動 */
    if (per->on & PDD_DGT_KR ) {
        pan = ADXT_GetOutPan(adxt, 0);
        pan = min(ADXT_PAN_RIGHT, pan+1);
        ADXT_SetOutPan(adxt, 0, pan);
    }
    /* パンを左へ移動 */
    if (per->on & PDD_DGT_KL ) {
        pan = ADXT_GetOutPan(adxt, 0);
        pan = max(ADXT_PAN_LEFT, pan-1);
        ADXT_SetOutPan(adxt, 0, pan);
    }
}

```

Title	Function	Function Name	No
関数仕様	サーバ関数の呼び出し頻度の設定	ADXT_SetSvrFreq	6.20

[書 式] void ADXT\_SetSvrFreq(ADXT adxt, long freq);

[入 力] adxt : ADXT ハンドル

freq : サーバ関数の呼び出し頻度 ( 1 秒当たりの呼び出し回数)

[出 力] なし

[関数値] なし

[機 能] サーバ関数 (ADXT\_ExecServer 関数) の呼び出し頻度の設定します。  
この関数によりサーバ関数が一回にデコードする量を制御できます。  
デフォルトでは、60 (V-Sync) が設定されています。

[備 考] 通常、サーバ関数は V-Sync 割り込みで呼び出されるので、この関数を使用する必要はありません。

Title	Function	Function Name	No
関数仕様	サーバ関数 ( 内部状態の更新)	ADXT_ExecServer	6.21

[書 式] void ADXT\_ExecServer(void);

[入 力] なし

[出 力] なし

[関数値] なし

[機 能] ライブラリの内部状態を更新し、デコード処理を行います。

[備 考] V-Sync 毎に呼び出さなければなりません。

ADXT\_Init 関数で V-Sync 割り込み処理に登録されるので、ユーザがこの関数を使用することはありません。



Title	Function	Function Name	No
関数仕様	エラーコードの取得	ADXT_GetErrCode	6.22

[書 式] long ADXT\_GetErrCode(ADXT adxt);

[入 力] adxt : ADXT ハンドル

[出 力] なし

[関数値] エラーコード

[機 能] エラーコードを取得する。

この値は、ADXT\_ClearErrCode 関数でクリアするまで保持されます。

#### エラーコード表

定 義 名	定数値	エ ラ ー 状 態
ADXT_ERR_OK	0	正常
ADXT_ERR_SHRTBUF	1	入力バッファにデータが供給されない。 G D リードエラー時に発生。
ADXT_ERR_SNDBLK	2	サウンドブロックエラー。 サウンドブロックが停止している。

Title	Function	Function Name	No
関数仕様	エラーコードのクリア	ADXT_ClearErrCode	6.23

[書 式] void ADXT\_ClearErrCode(ADXT adxt);

[入 力] adxt : ADXT ハンドル

[出 力] なし

[関数値] なし

[機 能] エラーコードをクリアする。

[用 例]

```

/* 44KHz のモノラルデータを 1 ストリームだけ再生する場合 */
#define WKSIZE ADXT_CALC_WORK(1, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE]; /* 作業領域 */
ADXT adxt; /* ADXT ハンドル */
long errcode; /* エラーコード */

adxt = ADXT_Create(1, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_StartFname(adxt, "srf01.adx"); /* 再生開始 */
for (;;) {
    errcode = ADXT_GetErrCode(adxt);
    if ( errcode != ADXT_ERR_OK ) {
        ADXT_ClearErrCode(adxt);
        ADXT_Stop(adxt);
    }
}

```

Title	Function	Function Name	No
関数仕様	自動エラー回復機能の設定	ADXT_SetAutoRcvr	6.24

[書 式] void ADXT\_SetAutoRcvr(ADXT adxt, long mode);

[入 力] adxt : ADXT ハンドル

mode : エラー回復モード

(ADXT\_RMODE\_NOACT, ADXT\_RMODE\_STOP, ADXT\_RMODE\_REPLAY)

[出 力] なし

[関数値] なし

[機 能] 自動エラー回復モードを設定します。

自動エラー回復機能が有効の場合、エラーが発生すると約 1 秒後に以下の動作を行います。

[備 考] デフォルトでは、ADXT\_RMODE\_STOP に設定されています。

定 義 名	定数値	エラー回復処理
ADXT_RMODE_NOACT	0	エラーリカバーしない。
ADXT_RMODE_STOP	1	自動的に停止し、動作状態が ADXT_STAT_STOP になる。
ADXT_RMODE_REPLAY	2	G D からのデータの供給が途切れたときに、自動的にファイルの先頭から再生する。 その他のエラーの場合には、自動的に停止する。

[用 例]

```

/* 44KHz のステレオデータと 22KHz のモノラルデータを再生する場合 */
#define WKSIZ44 ADXT_CALC_WORK(2, ADXT_PLY_STM, 2, 44100)
#define WKSIZ22 ADXT_CALC_WORK(1, ADXT_PLY_STM, 2, 22050)

char *work44[WKSIZ44]; /* 作業領域 (BGM用) */
char *work22[WKSIZ22]; /* 作業領域 (セリフ用) */
ADXT adxt; /* ADXT ハンドル (BGM用) */
ADXT adxt2; /* ADXT ハンドル (セリフ用) */
long pan; /* 現在のパンポット値 */

adxt = ADXT_Create(2, work44, WKSIZ44); /* BGM 用 ADXT ハンドル */
ADXT_SetAutoRcvr(adxt, ADXT_RMODE_REPLAY); /* オートリピートモード */
adxt = ADXT_Create(1, work22, WKSIZ22); /* セリフ用 ADXT ハンドル */
ADXT_SetAutoRcvr(adxt, ADXT_RMODE_STOP); /* オートストップモード */

.
.
ADXT_StartFname(adxt, "BGM.ADX"); /* BGM再生開始 */
.
.
ADXT_StartFname(adxt2, "SRF01.ADX"); /* セリフ再生開始 */
.
.

```

Title	Function	Function Name	No
関数仕様	エラーコールバック関数の設定	ADXT_EntryErrFunc	6.25

[書 式] void ADXT\_EntryErrFunc(void (\*func)(void \*obj, char \*msg), void \*obj);

[入 力] adxt : ADXT ハンドル  
func : ユーザのコールバック関数  
obj : コールバック関数の第 1 引数

[出 力] なし

[関数値] なし

[機 能] エラーコールバック関数を登録します。  
エラーが発生すると登録されたコールバック関数が以下の形式で呼び出されます。

```
(*func)(*obj, char *msg);
```

この関数の第 1 引数 obj は、ADXT\_EntryErrFunc 関数の第 2 引数となります。。また、第 2 引数 msg は、エラーメッセージです。この関数は、デバッグ用の関数ですので、アプリケーションのマスターアップ時には、何もしない関数に置き換えてください。

[用 例]

```
/* エラーコールバック関数 */
void user_adx_error(void *obj, char *msg)
{
    /* エラーが発生するとこの関数が呼び出されます。 */
    /* msg にエラーメッセージが渡されます。 */
    /* msg は、R5 レジスタに格納されているので、R5 レジスタのアドレスを */
    /* ダンプすることによってエラーメッセージを知ることができます。 */
    /* この関数は、V-Sync 割り込みから呼び出されることがありますので、 */
    /* Ninja の関数は使用しないでください。 */
    /* また、アプリケーションをリリースするときは、何もせずに戻るような */
    /* 関数にしてください。 */
    .
    for (;;) /* リリース時には無くす */
    .
}

/* メイン関数 */
void main(void)
{
    /* A D X ライブラリの初期化 */
    ADXT_Init();
    /* エラーコールバック関数の登録 */
    ADXT_EntryErrFunc(user_adx_error, NULL);
    .
}
}
```

Title	Function	Function Name	No
関数仕様	再生の一時停止・再開	ADXT_Pause	6.26

[書 式] void ADXT\_Pause(ADXT adxt, long sw);

[入 力] adxt : ADXT ハンドル

sw : ポーズスイッチ (1=一時停止,0=再開)

[出 力] なし

[関数値] なし

[機 能] 再生の一時停止・再開を行います。sw に 1 を設定すると一時停止、sw に 0 を設定すると再開します。再生状態が STOP のときに一時停止を設定すると、音声の再生を開始する関数を実行しても音声は再生されず、PLAYING の状態で一時停止します。この状態から一時停止を解除することによって、即座に音声を再生できます。

[用 例]

( 1 ) 通常の一時停止

```

/* 44KHz のステレオデータを 1 ストリームだけ再生する場合 */
#define WKSIZE ADXT_CALC_WORK(2, ADXT_PLY_STM, 1, 44100)

char *work[WKSIZE];          /* 作業領域 */
ADXT adxt;                   /* ADXT ハンドル */
long pause_flag;             /* ポーズフラグ */

adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
pause_flag = 0;
ADXT_StartFname(adxt, "music.adx"); /* 再生開始 */
for (;;) {
    if ( per->on & PDD_DGT_TA ) {
        if ( pause_flag == 0 ) /* 一時停止 */
            ADXT_Pause(adxt, pause_flag=1);
        else /* 再生再開 */
            ADXT_Pause(adxt, pause_flag=0);
    }
}

```

( 2 ) G D 上の音声データを即時に再生する

```

adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_Pause(adxt, 1); /* 一時停止 */
ADXT_StartFname(adxt, "music.adx"); /* 再生開始 */
/* 音声は再生されず、状態が PLAYING になる */
for (;;) {
    /* A ボタンが押されると即時に音声は出力される */
    if ( per->on & PDD_DGT_TA ) {
        ADXT_Pause(adxt, 0);
    }
}

```

Title	Function	Function Name	No
関数仕様	シームレス連続再生ファイルの登録	ADXT_EntryFname	6.27

[書 式] void ADXT\_EntryFname(ADXT adxt, char \*fname);  
 [入 力] adxt : ADXT ハンドル  
           fname : 音声ファイル名  
 [出 力] なし  
 [関数値] なし  
 [機 能] fname に指定された音声ファイルをシームレス連続再生用のファイルとして登録します。ループ設定のない ADX データのみ登録できます。

Title	Function	Function Name	No
関数仕様	シームレス連続再生の開始	ADXT_StartSeamless	6.28

[書 式] void ADXT\_StartSeamless(ADXT adxt);  
 [入 力] adxt : ADXT ハンドル  
 [出 力] なし  
 [関数値] なし  
 [機 能] シームレス連続再生を開始します。

Title	Function	Function Name	No
関数仕様	シームレスループ再生の設定	ADXT_SetSeamlessLp	6.29

[書 式] void ADXT\_SetSeamlessLp(ADXT adxt, long flg);  
 [入 力] adxt : ADXT ハンドル  
           flg : 0-ループ再生しない、1-ループ再生する。  
 [出 力] なし  
 [関数値] なし  
 [機 能] シームレスループ再生の設定を行います。

Title	Function	Function Name	No
関数仕様	シームレスループ再生の開始	ADXT_StartFnameLp	6.30

[書 式] void ADXT\_StartFnameLp(ADXT adxt, char \*fname);  
 [入 力] adxt : ADXT ハンドル  
           fname : 音声ファイル名  
 [出 力] なし  
 [関数値] なし  
 [機 能] 指定された音声ファイルを繰り返し再生します。

Title	Function	Function Name	No
関数仕様	シームレス連続再生の解除	ADXT_ReleaseSeamless	6.31

[書 式] void ADXT\_ReleaseSeamless(ADXT adxt);

[入 力] adxt : ADXT ハンドル

[出 力] なし

[関数値] なし

[機 能] シームレス連続再生を解除します。

Title	Function	Function Name	No
関数仕様	登録ファイル数の取得	ADXT_GetNumFiles	6.32

[書 式] void ADXT\_GetNumFiles(ADXT adxt);

[入 力] adxt : ADXT ハンドル

[出 力] なし

[関数値] なし

[機 能] 登録されているファイルを取得します。





## 付録 1 . A D Xライブラリ使用時に必要なファイル

A D Xライブラリを使用するためには、以下のファイルが必要です。

### < A D Xライブラリ本体 >

CRI_ADXT.H	ADX 再生ライブラリヘッダファイル
CRI_ADXF.H	ADX ファイルシステムライブラリヘッダファイル
CRI_ADXS.LIB	SHINOBI 版ライブラリファイル

### < A D X ツール >

ADXENCD.EXE	A D Xエンコーダ
ADXCAT.EXE	音声データ結合ツール
AFSLNK.EXE	A D Xファイルシステム結合ツール
AX.EXE	一括エンコード用ユーティリティプログラム

## 付録 2 . サウンドデータ作成上の注意

ADX 再生ライブラリにより音声データを再生中に、サウンドブロック内の効果音や MIDI シーケンスを同時に再生できます。Shinobi 環境において ADX 再生ライブラリは、サウンドライブラリの PCM ストリームの機能を用いて実現されています。この時、PCM バッファは、サウンド RAM 最後から 4040H ずつ割り当てられます。従って、ADXT ハンドルの使用するチャンネル数から PCM バッファのサイズを求め、その領域が重ならないようにマルチユニットファイルを作成する必要があります。

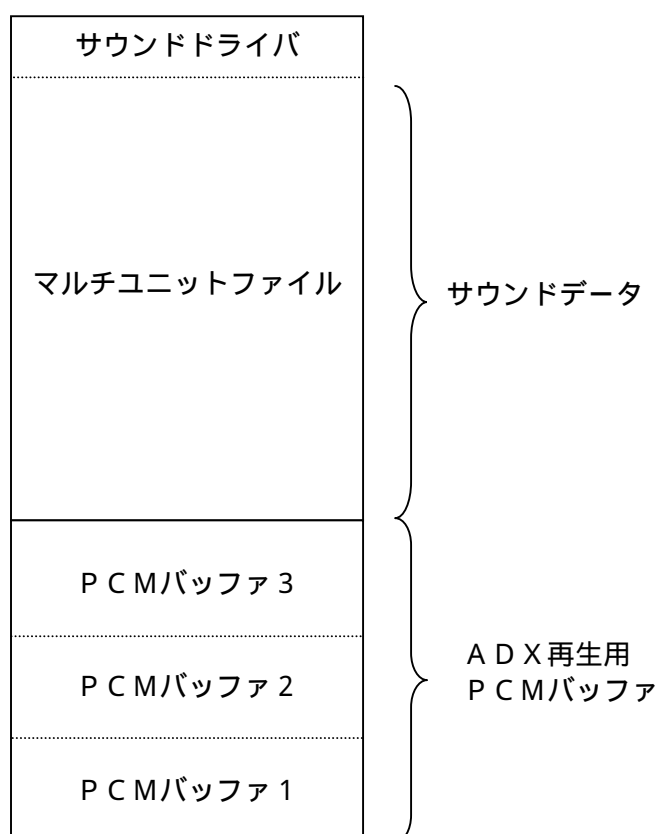


図 サウンド RAM の配置

### 付録 3 . データ読み込みの高速化について

A D X ファイルシステムにより、G D から音声を再生している最中に、データを読み込むことができます。しかしながら、デフォルトの設定では、入力バッファの 8 5 % を割るとデータの再読み込みが始まるため、複数のゲームデータファイルを読み込む場合、ゲームデータと音声データを交互に読み込むことになります。ADXT\_SetReloadSct 関数を使用し音声データの再読み込み開始量を調整することにより、ゲームデータの読み込みを高速化できます。再読み込み開始量は、音声データの 1 秒分を指定します。例えば、44.1KHz のステレオ音声は約 50Kbyte/sec なので、2 5 セクタを指定することになります。また、入力バッファを大きくすることにより、再読み込みの間隔を大きくすることができるので、さらにデータ読み込みを高速化できます。暫定仕様なので、ADXT\_SetReloadSct 関数はリファレンスマニュアルには記載されていません。将来、このような指定をしなくても高速にデータが読み出せるようになります。

```
/* 44KHz のステレオデータを再生する場合 */
#define WKSIZE      ADXT_CALC_WORK(2, ADXT_PLY_STM, 3, 44100)

char work[WKSIZE];          /* 作業領域 */
ADXT adxt;                  /* ADXT ハンドル */
ADXF adxf;                  /* ADXF ハンドル */

adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
ADXT_SetReloadSct(adxt, 25);        /* 再読み込み量の設定 */
.
ADXT_StartFname(adxt, "BGM.ADX");   /* 音声再生開始 */
.
adxf = ADXF_Open("GAMEDAT1.BIN", NULL); /* ファイルオープン */
ADXF_ReadNw32(adxf, nsct, buf);        /* データ読み込み */
while ( ADXF_GetStat(adxf) != ADXF_STAT_READEND )
;
ADXF_Close(adxf);                  /* ファイルクローズ */
.
(GAMEDAT1 が 2 秒以内に読み込めれば音声データの再読み込みは発生しない)
.
adxf = ADXF_Open("GAMEDAT2.BIN", NULL); /* ファイルオープン */
ADXF_ReadNw32(adxf, nsct, buf);        /* データ読み込み */
while ( ADXF_GetStat(adxf) != ADXF_STAT_READEND )
;
ADXF_Close(adxf);                  /* ファイルクローズ */
```

最大 G D ストリーム数(ADXT\_CALC\_WORK の第 3 引数)を増やすことにより、入力バッファを増やすことができます。1 増やすごとに 1 秒分の余裕ができます。また、B G M とセリフを同時に再生するが、データを読み出すときにはセリフが再生されないことが保証されていれば、セリフのためのバッファ量をデータ読み込みに割り当てることができます。

#### 付録 4 . シーク音の軽減方法

A D X再生ライブラリは、デフォルトで反応速度を重視したバッファリング制御を行います。バッファの入力バッファの85%を割るとデータ読み込みが発生します。従って、G D上のデータの配置位置が離れていると頻繁にシーク音が発生します。このシーク音は、ADXT\_SetReloadSct 関数によって軽減することができます。

例えば、B G Mとセリフを再生する場合、セリフの読み込み間隔を大きくすることによりシーク回数を減らすことができます。22KHz モノラルのセリフ再生する場合、ビットレートは、約 12.5Kbyte/sec になります。従って、2ストリームの並行再生の場合、1秒分バッファリングすれば並行再生できるので、セリフ側の再読み込みセクタ数を7セクタとすれば良いことになります。また、セリフ用の作業領域を計算する時に、並行再生ストリーム数を大きく指定することにより、入力バッファを大きくとることができます。これにより、よりシークの間隔を大きくすることができます。

```
/* 44KHz ステレオを2ストリーム並行再生する場合の作業領域 */
#define WKSIZ44S ADXT_CALC_WORK(2, ADXT_PLY_STM, 2, 44100)
/* 22KHz モノラルを3ストリーム並行再生する場合の作業領域 */
#define WKSIZ22M ADXT_CALC_WORK(1, ADXT_PLY_STM, 3, 22050)
/* 本来、2ストリームしか再生しないが、3ストリームとすることにより */
/* 入力バッファを増やすことができる */

char wk44[WKSIZ44S], wk22[WKSIZ22M]; /* 作業領域 */
ADXT adxt_bgm, adxt_srv; /* ADXT ハンドル */

adxt_bgm = ADXT_Create(2, wk44, WKSIZ44S); /* ADXT ハンドルの生成 */
adxt_srf = ADXT_Create(1, wk22, WKSIZ22M); /* ADXT ハンドルの生成 */
ADXT_SetReloadSct(adxt_srf, 7); /* 再読み込み量の設定 */
.
ADXT_StartFname(adxt_bgm, "BGM.ADX"); /* 音声再生開始 */
.
if ( イベント発生 )
    ADXT_StartFname(adxt_srv, "SRF001.ADX");
```

## 付録 5 . MPEG Sofdec との並行再生について

A D X再生ライブラリにより MPEG Sofdec との並行再生ができます。B G MをG Dストリーム再生しながらムービーを非同期に並行再生することにより、ゲームシーンからムービーシーンへのシームレスな移り変わりを実現できます。しかしながら、ムービー用のバッファ容量を増やす必要があります。Sofdec ハンドルを生成する時に、最大ビットストリーム量を実際のストリーム量よりも大きい値を設定してください。目安としては、約 1.5 倍の数値を指定してください。これは、シークタイムを約 1 秒として見込んでいますので、データの配置の仕方によっては、減らせる可能性があります。また、ムービーが途切れる場合は、この値を増やしてください。

```

/* 44KHz ステレオを 2 ストリーム並行再生する場合の作業領域 */
#define WKSIZ44S ADXT_CALC_WORK(2, ADXT_PLY_STM, 2, 44100)
/* 2 ストリームは、B G Mとムービーという意味 */
char wk44[WKSIZ44S]; /* 作業領域 */

MWPLY ply; /* MWPLY ハンドル */
ADXT adxt_bgm; /* ADXT ハンドル */

adxt_bgm = ADXT_Create(2, wk44, WKSIZ44S); /* ADXT ハンドルの生成 */
ADXT_SetReloadSct(adxt_bgm, 25); /* シーク音の軽減のため */

/* MWPLY ハンドルの生成 */
memset(&cprm, 0, sizeof(cprm)); /* 予約メンバのゼロ設定のために必要 */
cprm.opt.first_hpel = ON;
cprm.ftype = MWD_PLY_FTYPE_MPV;
cprm.dtype = MWD_PLY_DTYPE_AUTO;
cprm.max_bps = 900*1024*8; /* 通常 600Kbyte/sec の 1.5 倍を指定 */
cprm.max_width = 320;
cprm.max_height = 240;
cprm.nfrm_pool_wk = 3;
cprm.wksize = mwPlyCalcWorkSofdec(
    cprm.ftype,
    cprm.max_bps,
    cprm.max_width,
    cprm.max_height,
    cprm.nfrm_pool_wk);
cprm.work = syMalloc(cprm.wksize);
ply = mwPlyCreateSofdec(&cprm);

ADXT_StartFname(adxt_bgm, "BGM.ADX"); /* 音声再生開始 */
.
if ( イベント発生 )
    mwPlyStartFname(ply, "SCENE01.M1V"); /* ムービー再生

```

## 付録 6 . GDFS・Ninja ライブラリとの併用について

ADX ライブラリは、Shinobi ライブラリを使用しています。従って、基本的には GDFS との併用することができます。しかしながら、GDFS が GD-ROM のピックアップを占有してしまうと、音声データが読み込めずに音声途切れてしまうことがあります。基本的には、ADX ファイルシステムを使うことをお勧めしますが、njLoadTexture 関数などでどうしても GDFS を使用しなければならない場合は、ADXT\_IsIbufSafety 関数や ADXT\_GetNumSctIbuf 関数を用いて、入力バッファの状態を確認後、GDFS を使用してください。

ADXT\_IsIbufSafety 関数は、再読み込み開始セクタ数以上のデータが入力バッファにある時に 1 を返します。

ADXT\_GetNumSctIbuf 関数は、入力バッファ内のセクタ数を返します。音声ストリームのビットレートから音声途切れるまでの時間を求めて調節してください。

```
/* 44KHz のステレオデータを再生する場合 */
#define WKSIZE      ADXT_CALC_WORK(2, ADXT_PLY_STM, 3, 44100)

char *work[WKSIZE];          /* 作業領域 */
ADXT adxt;                   /* ADXT ハンドル */
ADXF adxf;                   /* ADXF ハンドル */

adxt = ADXT_Create(2, work, WKSIZE); /* ADXT ハンドルの生成 */
.
ADXT_StartFname(adxt, "BGM.ADX"); /* 音声再生開始 */
.
while ( ! ADXT_IsIbufSafty(adxt) )
;
njLoadTextrure(&tlist);
```

#### 付録 7 . ADX マルチストリームシステムの仕組み

ADX マルチストリームシステムによって、GD-ROM 上の別々に格納された音声や動画ファイルを、並行再生することができます。

GD-ROM 上の音声ファイルであれば 4 ストリームまで同時に再生することができます。例えば、44KHz・ステレオの BGM と効果音に、44KHz・モノラルのセリフ 2 つを同時に再生できます。また、MPEG SofdecF/X によって動画ファイルを再生しながら、別の音声ファイルを再生することもできます。

ADX ファイルシステムを併用することにより、GD-ROM 上の音声データを再生しながら、グラフィックデータなどを読み込むことができます。  
以下に、ADX マルチストリームの仕組みを示します。

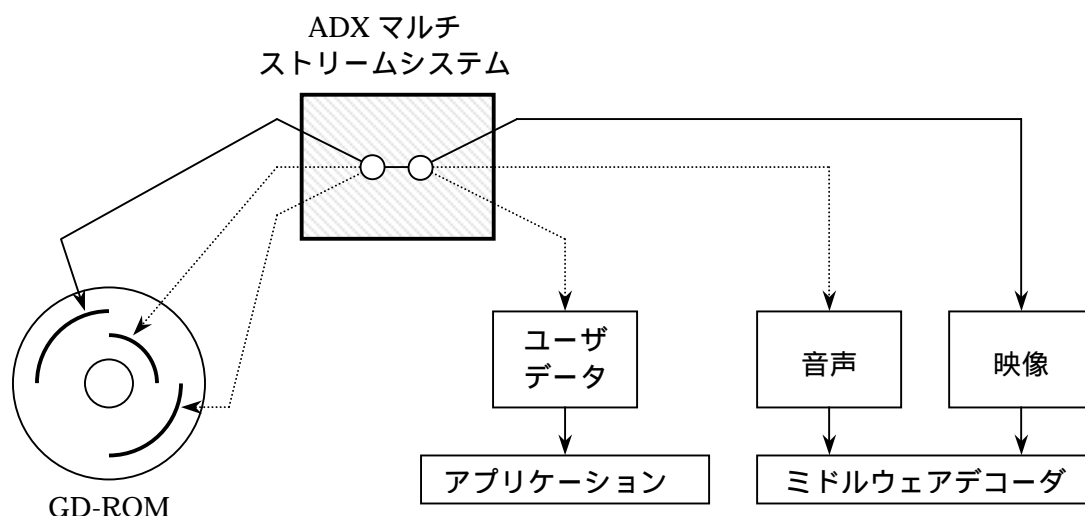


図 ADX マルチストリームの仕組み

ADX マルチストリームシステムは、データの流量管理を行います。各ストリームごとにデータバッファを持ち、データ量が再読込開始セクタ数を下回るとデータを読み込みます。デフォルトの再読込開始セクタ数は、バッファの 85% に設定されています。

以下に、ADX のデータ流量管理の仕組みを示します。

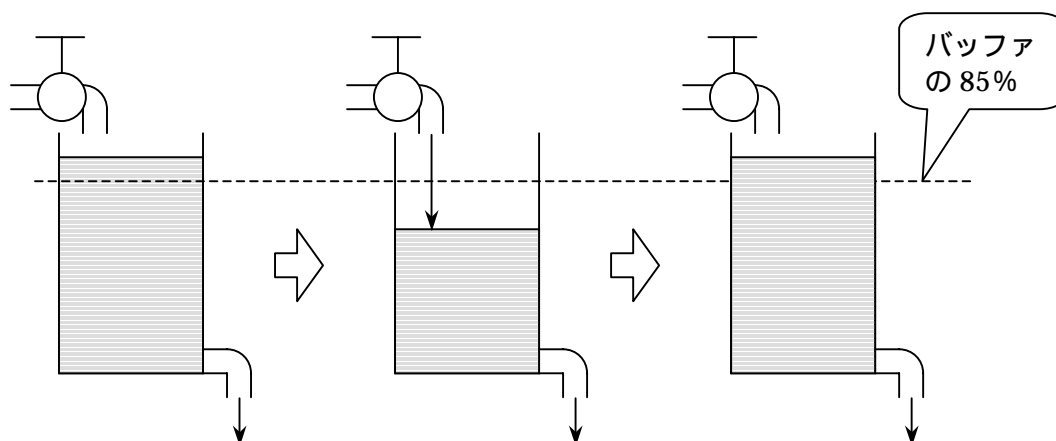


図 データ流量管理の仕組み

## 付録 8 . SofdecF/X ムービーと音声のマルチストリーミング

SofdecF/X ムービーと音声をマルチストリーミングする場合、下記のようにバッファを設定すると作業領域を削減できます。

```
#define NADX          (3)    // 並行再生する音声のストリーム数
#define NSFD          (2)    // 並行再生するムービーのストリーム数

#define ADX_NSTM      (NADX*( (NSFD != 0) ? (NSFD+1) : 1))
#define SFD_NSTM      (NSFD+( (NADX != 0) ? 1 : 0))
#define MOVIE_BPS      (300*1024*8)

// ムービー再生ハンドルの生成
cprm.max_bps = MOVIE_BPS * SFD_NSTM;           // 最大ビットレート
for (i=0; i<NSFD; i++)
    ply[i] = mwPlyCreateSofdec(&cprm);
// ADX 再生ハンドルの生成
wksize=ADXT_CALK_WORK(2, ADXT_PLY_STM, ADX_NSTM, 44100));
for (i=0; i<NADX; i++) {
    wk = syMalloc(wksize);
    adxt[i] = ADXT_Create(2, wk, wksize);
}
// 音声再生開始
ADXT_StartFname(adxt[0], "music.adx");
for (;;) {
    if ( Aボタンが押された )
        mwPlyStartFname(ply[0], "sample.sfd");    // ムービー再生開始
}
```



## 付録 9 . ループ再生

ADX は、音声データの一部をシームレスにループ再生できます。ADXT\_SetLpFlg 関数により、ループするか否かを設定できます。再生中にループを解除すると、その音声データの最後まで再生し終了します。再生中は、ループ解除のみ設定できます。

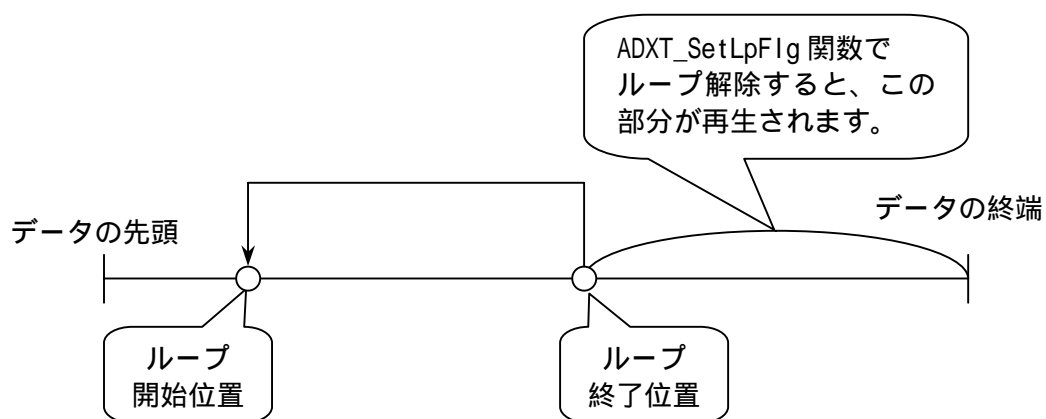


図 ループ再生

#### 付録 10 . シームレス連続再生

ADX は、複数の音声データをシームレスに再生することができます。下記の例では、音声データ smpsl1.adx、smpls1.adx、smpls2.adx、smpls2.adx、smpls3.adx の順に再生します。

```
// ADX ハンドルの生成
adxt = ADXT_Create(2, adxt_work, ADXT_WKSIZE);

// 再生する音声データの登録
ADXT_EntryFname(adxt, "smpls1.adx");
ADXT_EntryFname(adxt, "smpls1.adx");
ADXT_EntryFname(adxt, "smpls2.adx");
ADXT_EntryFname(adxt, "smpls2.adx");
ADXT_EntryFname(adxt, "smpls3.adx");

// シームレスループ再生の設定
ADXT_SetSeamlessLp(adxt, 1);

// シームレス連続再生の開始
ADXT_StartSeamless(adxt);
```

付録 11 . 音声出力までのタイムラグをなくす方法

以下に、音声出力されるまでの流れを示します。

( 1 ) GD-ROM 上から音声データを読み込む。

( 2 ) 音声データのヘッダを解析する。

( 3 ) 音声データのデコード。

( 4 ) サウンド RAM 上の PCM ストリームバッファにデータを溜める。

( 5 ) 音声出力開始。

従って、ADXT\_StartFname 関数を実行しても、すぐに音声は出力されません。そこで、以下の方法により、音声出力までのタイムラグをなくすことができます。

```

        :
        :
ADXT_Pause(adxt, ON);                                /* 一時停止          */
ADXT_StartFname(adxt, "sample.adx");                  /* 再生開始          */

for (;;) {
    :
    if ( 音声を出力するタイミング ) {
        ADXT_Pause(adxt, OFF);                        /* 一時停止の解除    */
    }
    :
    :
```

## 付録 12 . クロスフェード

クロスフェードすることにより、ゲーム中の音楽を途切れさせることなく、自然にシーンを切り替える事ができます。

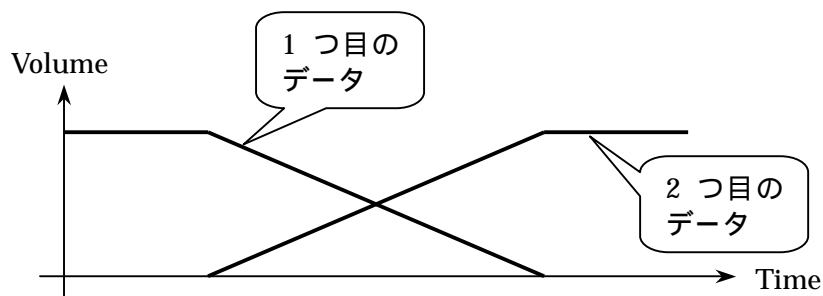


図 クロスフェードしているボリュームの時間変化

```
vol1 = 0;
vol2 = -1000;
for (;;) {
    if ( 1 つ目のデータ再生を開始するタイミング ) {
        ADXT_SetOutVol(adxt1, vol1);
        ADXT_StartFname(adxt1, "sample1.adx");
    }
    if ( 2 つ目のデータ再生を開始するタイミング ) {
        ADXT_SetOutVol(adxt2, vol2);
        ADXT_StartFname(adxt2, "sample2.adx");
    }
    if ( ADXT_GetStat(adxt2) == ADXT_STAT_PLAYING ) {
        ADXT_SetOutVol(adxt1, vol1-=10;);
        ADXT_SetOutVol(adxt2, vol2+=10;);
    }
}
```

### 付録 13 . サンプルプログラムについて

本ライブラリに添付しているサンプルプログラムの主なプログラムについて説明します。

adxsm01: 単純な G D ストリーム再生

'smpadx01.c' は、G D 上の音声データファイルを再生する簡単なサンプルプログラムです。'sample.adx'を再生します。

adxsm02: メモリ再生&メモリインデックス再生

'smpadx02.c' は、G D 上の'sample.adx'と'smpl\_mem.acx'をワーク R A M 上に読み込み後、パッドによりメモリ再生とメモリインデックス再生を行うプログラムです。

A ボタン : メモリ再生

A ボタン以外 : メモリインデックス再生

adxsm03: G D インデックス再生

'smpadx03.c' は、G D 上の'pat01.afs'のパーティション情報を読み込み後、パッドにより、G D インデックス再生を行います。同時に 4 ストリームの再生を行います。

adxsm04: ADX ファイルシステム

'smpadx04.c' は、G D 上の'pat02.afs'のパーティション情報を読み込み後、パッドにより、パーティション内のファイルを読み込みます。