

サウンドドライバ解説

Ver 1.2.1

任天堂株式会社発行

このドキュメントの内容は、機密情報であるため、厳重な取り扱い、管理を行ってください。

目次

1	導入	5
1.1	はじめに	5
1.2	サウンドドライバの構成	5
1.3	NITRO-Composer	5
2	サウンドハードウェア	6
2.1	サウンド回路概略	6
2.2	チャンネル	6
2.2.1	ADPCM / PCM	7
2.2.2	PSG矩形波	7
2.2.3	ノイズ	7
2.3	サウンドキャプチャ	7
3	ARM7 コマンド処理	8
3.1	コマンド処理の流れ	8
3.2	サウンド関数とコマンド	9
3.3	コマンドパケット	9
3.4	コマンドのフラッシュ	10
3.5	コマンド応答の受信	10
3.6	コマンドタグ	11
3.7	空きコマンドパケット不足時の動作	11
3.8	サウンドフレーム	12
4	サウンド再生	13
4.1	シーケンス再生とチャンネル制御	13
4.2	チャンネル制御	13
4.2.1	チャンネルのロック	13
4.2.2	チャンネルのセットアップ	13
4.2.3	タイマーの開始と停止	13
4.2.4	チャンネルパラメータ	13
5	サウンドキャプチャ	15
5.1	サウンドキャプチャの概要	15
5.2	サウンドキャプチャの使い方	15
5.3	サウンドキャプチャの不具合について	15
6	サウンドアラーム	16
6.1	サウンドアラームの概要	16
6.2	サウンドアラームの使い方	16
7	ドライバ情報取得	17

7.1	ドライバ情報取得の概要	17
7.2	情報構造体の取得	17
7.3	その他の情報取得	17
8	NITRO-Composerと併用する際の注意点	19
8.1	プレイヤーの使用	19
8.2	チャンネルの使用	19
8.3	サウンドキャプチャの使用	19
8.4	サウンドアラームの使用	19

コード

コード 3-1	コマンドのフラッシュとコマンド応答処理	11
コード 7-1	ドライバ情報構造体の取得	17

表

表 2-1	チャンネル番号	6
-------	---------------	---

図

図 2-1	サウンド回路概略	6
図 3-1	コマンド処理の流れ	8
図 3-2	コマンドパケット	9
図 3-3	コマンドパケットの状態遷移	10

改訂履歴

版	改訂日	改訂内容
1.2.1	2008-10-16	1. NITRO-SDK から TWL-SDK に表記変更
1.2.0	2005-07-04	1. コマンドの説明を改訂 「コマンドパケット」の用語を追加し、その他の用語も表記を統一 2. サウンドキャプチャの不具合を記載 3. コード 7-1 の誤記修正
1.1.1	2005-05-10	1. コマンドの状態の表記修正 2. コマンドタグの説明修正 3. 誤記修正
1.1.0	2005-04-26	1. サウンドキャプチャの誤記修正 2. サウンド関数の追加に伴う修正 3. コマンドの説明修正
1.0.0	2005-04-13	初版

1 導入

1.1 はじめに

サウンドドライバ(SND)は、ニンテンドーDS のサウンドハードウェアをアプリケーションから比較的低レベルに扱うことが出来るライブラリです。ニンテンドーDS のサウンド機能は ARM7 に実装されたこのサウンドドライバを通じて使用できます。

本文書では、サウンドドライバの動作の仕組みと主要な関数群について解説します。各関数の詳細な説明は関数リファレンスを参照してください。

1.2 サウンドドライバの構成

サウンドドライバは大別すると次の3つのライブラリ部分に分けることが出来ます。

- ライブラリのインターフェイスを提供する ARM9 側ライブラリ
- ARM7 側のドライバコア部分
- ARM9、ARM7 間のデータのやりとりを受け持つコマンドライブラリ

サウンドドライバは、ARM9 側の関数群を通じて使用しますが、関数を呼び出してからARM7 で実際にサウンドが処理されるまでの動作の流れを理解しておく必要があり、特にARM9, ARM7 間でのコマンドのやりとりのフローが重要になってきます。これは、後述の「3 章ARM7 コマンド処理」で詳しく説明します。

1.3 NITRO-Composer

TWL-System パッケージには、シーケンスの再生やストリーミング再生、サウンドデータ管理などを含めたサウンドライブラリ「NITRO-Composer」が用意されています。このライブラリは、低レベル関数群であるサウンドドライバを使用して設計されていますので、NITRO-Composer を使いながらサウンドドライバの関数群を併用することが可能です。

ただし注意する点がありますので、後述の「8 章 NITRO-Composer と併用する際の注意点」を参照してください。

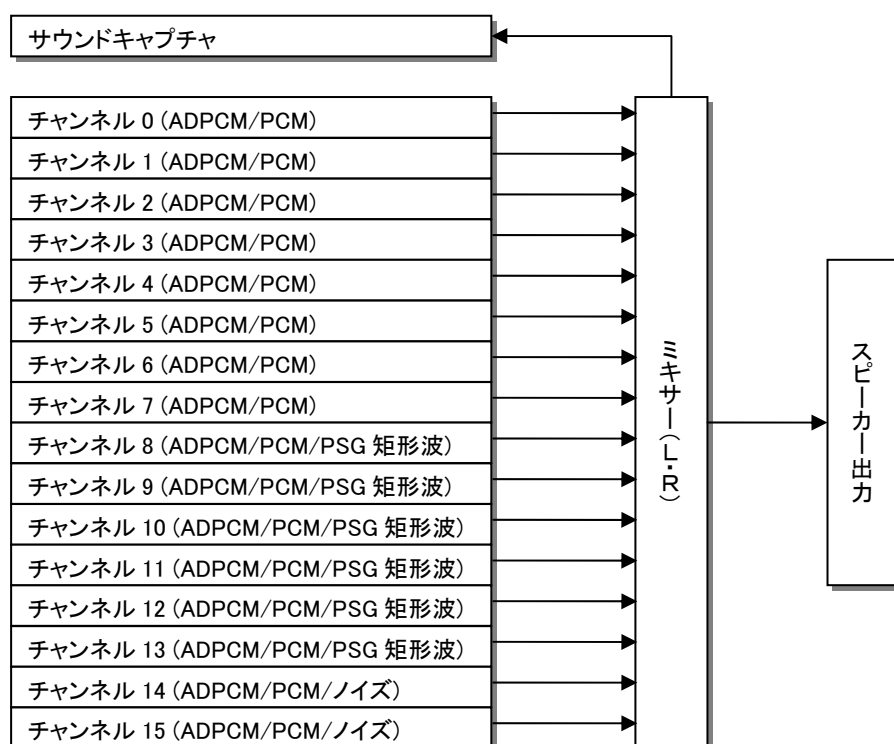
シーケンスやバンクのデータ形式は共通ですので、NITRO-Composer のドキュメントを参照してください。

2 サウンドハードウェア

2.1 サウンド回路概略

ニンテンドーDS のサウンドハードウェアは、それぞれ独立して制御可能な 16 チャンネル分のサウンド回路、それぞれのサウンド回路の音を足し合わせるミキサー、サウンドの出力をメモリに書き出すことが出来るサウンドキャプチャなどから成ります。

図 2-1 サウンド回路概略



2.2 チャンネル

チャンネルは、1度に1つの音を発音できる回路を表します。チャンネルは 16 個あり、最大同時に 16 音の再生が可能です。0 番～15 番のチャンネルには以下のような違いがあります。

表 2-1 チャンネル番号

チャンネル番号	機能
0, 2	ADPCM / PCM を再生できます。このチャンネルの出力をサウンドキャプチャの入力とすることもできます。
1, 3	ADPCM / PCM を再生できます。サウンドキャプチャとタイマーを共用しているため、サウンドキャプチャを使うときは、サウンドキャプチャの出力チャンネルとしてしか使えません。

4 ～ 7	ADPCM / PCM を再生できます。
8 ～ 13	ADPCM / PCM または PSG 矩形波を再生できます。
14, 15	ADPCM / PCM またはホワイトノイズを再生できます。

2.2.1 ADPCM / PCM

ADPCM/PCM の再生が可能なチャンネルでは、16bit の PCM、8bit の PCM 及び IMA-ADPCM が再生できます。

2.2.2 PSG矩形波

PSG 矩形波の再生が可能なチャンネルでは、デューティ比が設定可能な矩形波を再生できます。

2.2.3 ノイズ

ノイズの再生が可能なチャンネルでは、ホワイトノイズを再生できます。ホワイトノイズに設定項目はありません。

2.3 サウンドキャプチャ

ニンテンドーDS には出力波形データをメモリに書き出すサウンドキャプチャが 2 基搭載されています。上記の図ではミキサーの L チャンネル及び R チャンネルの出力をキャプチャしていますが、チャンネル 0 とチャンネル 2 の出力をキャプチャするように変更することも可能です。

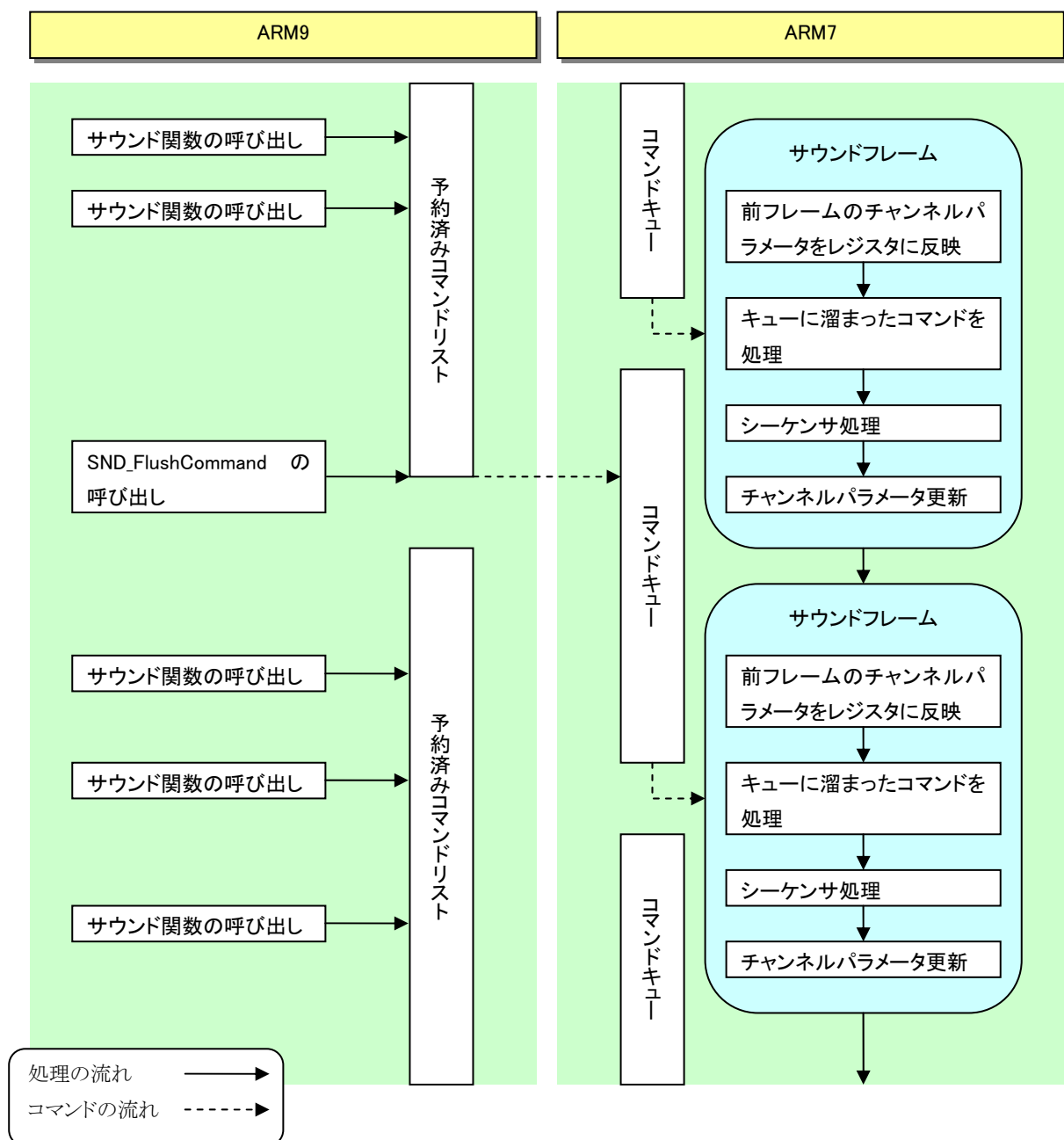
キャプチャ波形の分解能は 8bit もしくは 16bit です。

3 ARM7 コマンド処理

3.1 コマンド処理の流れ

サウンドドライバでは、各サウンド関数が呼び出されてもすぐに処理が開始されるわけではありません。サウンド関数が呼び出されると ARM9 側で予約済みコマンドリストに追加され、SND_FlushCommand を呼び出して初めて ARM7 で処理が開始されるようになっています。

図 3-1 コマンド処理の流れ



3.2 サウンド関数とコマンド

SND ライブラリの関数の多くは ARM7 で処理を行うためのコマンドとなっています。コマンドは ARM9 で蓄積されていき、明示的に SND_FlushCommand を実行してはじめて ARM7 に送られて処理が行われます。

ARM7 での処理を必要としない一部の関数は、呼び出されたときにその場で実行されます。

コマンドとして処理される関数には以下のものがあります。

- シーケンスコマンド

SND_StartSeq
SND_StartPreparedSeq
SND_PauseSeq
SND_SetPlayerVolume
SND_SetPlayerLocalVariable
SND_SetTrackMute
SND_SetTrackPitch
SND_SetTrackModDepth
SND_SetTrackAllocatableChannel

SND_PrepareSeq
SND_StopSeq
SND_SetPlayerTempoRatio
SND_SetPlayerChannelPriority
SND_SetPlayerGlobalVariable
SND_SetTrackVolume
SND_SetTrackPan
SND_SetTrackModSpeed

- チャンネルコマンド

SND_LockChannel
SND_StopUnlockedChannel
SND_SetupChannelPsg
SND_SetChannelVolume
SND_SetChannelPan

SND_UnlockChannel
SND_SetupChannelPcm
SND_SetupChannelNoise
SND_SetChannelTimer

- キャプチャコマンド

SND_SetupCapture

- アラームコマンド

SND_SetupAlarm

- タイマーコマンド

SND_StartTimer

SND_StopTimer

- グローバル設定コマンド

SND_SetMasterVolume
SND_ResetMasterPan

SND_SetMasterPan
SND_SetOutputSelector

- データ無効化コマンド

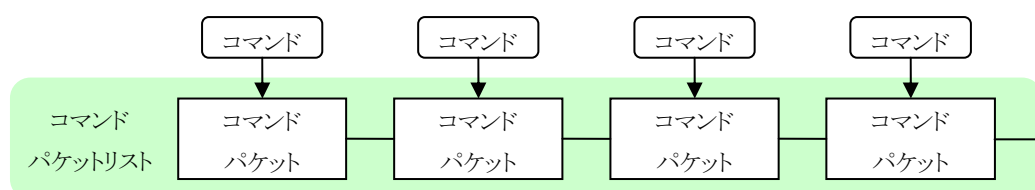
SND_InvalidateSeqData
SND_InvalidateWaveData

SND_InvalidateBankData

3.3 コマンドパケット

コマンドを ARM7 に送るための仕組みとして、コマンドパケットがあります。コマンドパケットは一つのコマンドを格納し、このコマンドパケットをリスト状に連結したコマンドパケットリストをひとかたまりとして、ARM7 へコマンドを送ります。

図 3-2 コマンドパケット



コマンドパケットには3つの状態があります。

- 新たにコマンドを登録できる「空きコマンドパケット」
- 登録されたコマンドがフラッシュされるのを待っている「予約済みコマンドパケット」
- フラッシュされた後、ARM7 で処理が完了するのを待っている「処理完了待ちコマンドパケット」

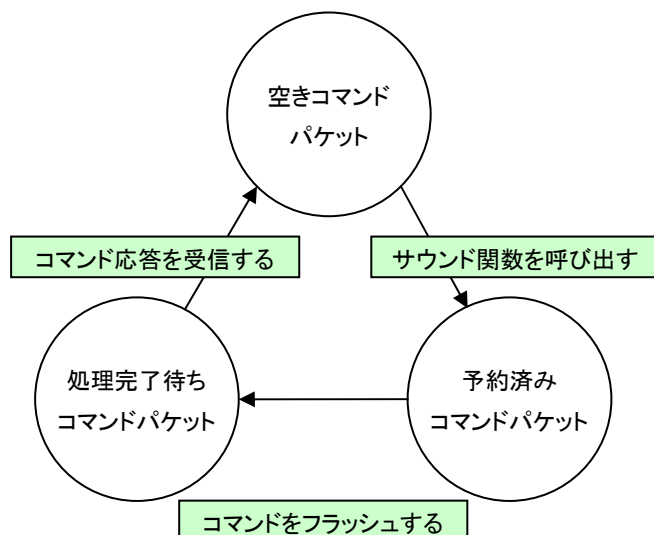
予約済み状態のコマンドを実行するためには、予約済みコマンドパケットリストをフラッシュする必要があります。この操作をコマンドのフラッシュと呼びます。また、処理完了待ちコマンドパケットを再び空きコマンドパケットに戻すためにはコマンド応答を受信して処理の完了を確認する必要があります。

コマンドパケットの数は有限 (256 個) です。定期的にコマンドのフラッシュ及びコマンド応答受信を行い、空きコマンドパケットを確保する必要があります。

それぞれの状態のコマンドパケット数を取得する関数、

SND_CountFreeCommand, SND_CountReservedCommand, SND_CountWaitingCommand が用意されています。

図 3-3 コマンドパケットの状態遷移



3.4 コマンドのフラッシュ

SND_FlushCommand は必要に応じて呼び出すことができ、細かいタイミングの調整が必要な場合は、後の項で説明するコマンドタグと組み合わせることによって ARM7 の処理に同期して動作させることが可能です。しかし、フラッシュしないとコマンドの処理が行われませんし、定期的にフラッシュしないと予約済みコマンドパケットリストが増えて空きコマンドパケットリストが不足していきますので、1フレームに1回程度の間隔で定期的に呼び出すのが良いでしょう。

3.5 コマンド応答の受信

処理完了待ちコマンドパケットを空きコマンドパケットに戻すためには、フラッシュしたコマンドの処理が完了したかどうかの応答を受信する必要があります。SND_RecvCommandReply を呼び出すと、既に処理が完了したコマンドパケットリストの中で最も古いものを空きコマンドパケットリストに戻し、そのリストを取得することができます。

コマンド応答の受信も、コマンドのフラッシュと同様に定期的に行わないと空きコマンドパケットが不足していきます。空きコマンドパケットを安定して確保するために、定期的に SND_RecvCommandReply を呼び出してください。

下記のコードは、毎フレーム呼び出すサウンド関数 `SoundMain` の一例です。毎フレームでコマンドのフラッシュとコマンドの応答受信を行っています。

コード 3-1 コマンドのフラッシュとコマンド応答処理

```
void SoundMain( void )
{
    // ARM7応答を受信する
    while ( SND_RecvCommandReply( SND_COMMAND_NOBLOCK ) != NULL ) {}

    // ARM7へコマンドを発行する
    SND_FlushCommand( SND_COMMAND_NOBLOCK );
}
```

`SND_FlushCommand` と `SND_RecvCommandReply` のパラメータとして、成功するまで関数内でブロックするかどうかを指定することができます。上記の例では `SND_COMMAND_NOBLOCK` を指定し、関数内でのブロックを行っていません。確実にコマンドのフラッシュや応答受信を完了する必要がある場合は `SND_COMMAND_BLOCK` を指定してください。

3.6 コマンドタグ

コマンドタグを使用すると、任意のコマンドの処理が完了したかどうかを判別することができ、ARM7 のコマンド処理と ARM9 のアプリケーションで同期を取ることができます。

`SND_GetCurrentCommandTag` を呼び出すとコマンドタグを取得し、タグを取得した時点より以前のコマンド実行が完了したかどうかを調べることができるようになります。`SND_IsFinishedCommandTag` を使うと、引数に指定したタグ以前のコマンドの実行が完了しているかどうかを調べることができます。`SND_WaitForCommandProc` は、取得したタグ以前のコマンドの実行が完了していない場合コマンドをすぐに実行するように ARM7 へ要求し、実行が完了するまで関数内で待ち続けます。

コマンドのフラッシュを行わないとコマンド処理が完了しませんので、コマンドタグを使って処理の完了を調べるときは注意が必要です。

3.7 空きコマンドパケット不足時の動作

空きコマンドパケットが不足していると、そのままでは新たにコマンドを追加することができません。空きコマンドパケットが不足している状態で新たにコマンド処理を伴うサウンド関数を呼び出した場合、以下の手順で空きコマンドパケットを確保しようとします。

- 処理完了待ちコマンドがあれば、コマンド応答を受信する
- それでも不足していれば、コマンドをフラッシュし ARM7 に即時実行を要求し、コマンド応答があるまで待つ

コマンドのフラッシュやコマンド応答待ちの動作が含まれるため、サウンド関数の呼び出しに時間がかかることもあります。また、同時に実行すべき処理が、途中のフラッシュによって分断される可能性もあります。処理の分断を回避するために、指定個数の空きコマンドを確保するまで待つ関数 `SND_WaitForFreeCommand` が用意されています。

空きコマンドパケット不足の問題を回避するためには、コマンドのフラッシュとコマンド応答の受信を定期的に行うようにしてください。

3.8 サウンドフレーム

ARM7 のサウンドフレームの間隔は約 **5.2ms** です。コマンドをフラッシュしてから実際に処理が実行されるまでにサウンドフレーム間隔分の遅延が発生する可能性があります。

例外として、コマンドをフラッシュする際に、引数として **SND_COMMAND_IMMEDIATE** を指定した場合、次のサウンドフレームを待たずに、即座にARM7で処理を開始することができます。

4 サウンド再生

4.1 シーケンス再生とチャンネル制御

サウンドドライバを用いた発音方法には、専用フォーマットのシーケンスデータを演奏させるシーケンス再生と、各チャンネルを直接制御して発音させる方法の2種類があります。

シーケンスデータの演奏は、NITRO-Composer を用いたほうがより高度な処理が可能ですので、ここではチャンネルを直接制御する発音方法について解説します。

4.2 チャンネル制御

チャンネルを直接制御して発音させるためには、以下の手順を踏みます。

- チャンネルをロックする
- 目的の再生方法に応じてチャンネルをセットアップする
- タイマーをスタートする

4.2.1 チャンネルのロック

各チャンネルは ARM7 に実装されているシーケンサによって自動的に発音のために確保され、発音が終わると開放されます。そのため、プログラマがチャンネルを直接操作する際にはシーケンサとの衝突を防ぐために `SND_LockChannel` を呼び出してチャンネルのロックを行う必要があります。チャンネルに対する操作は、チャンネルがロックされていることが前提となります。

ロックされたチャンネルはシーケンサからは使用できなくなりますので、ロックする必要がなくなったチャンネルは、`SND_UnlockChannel` を呼び出してロックを解除し、再びシーケンサから使用できるようにしてください。

4.2.2 チャンネルのセットアップ

ロックしたチャンネルに対して、PCM 再生 (`SND_SetupChannelPcm`)、PSG 矩形波再生 (`SND_SetupChannelPsg`)、ホワイトノイズ再生 (`SND_SetupChannelNoise`) の各セットアップ関数を目的に応じて呼び出します。PSG 矩形波とホワイトノイズは、それぞれ再生可能なチャンネル番号に対してのみセットアップ可能です。

4.2.3 タイマーの開始と停止

各チャンネルは、`SND_StartTimer` を呼びだしてタイマーをスタートさせることにより発音が始まります。タイマーのスタートは複数のチャンネルを1回の関数呼び出しで同時に行えますので、各チャンネルの発音タイミングをそろえることができます。また、後述のサウンドキャプチャやサウンドアラームも、同じタイマースタートの呼び出しでタイミングをそろえることができます。

発音を停止するには、`SND_StopTimer` を呼び出してタイマーをストップさせます。タイマーストップの呼び出しもタイマースタートと同じように複数のチャンネルをまとめて制御できます。

4.2.4 チャンネルパラメータ

各チャンネルにボリューム、タイマー、パンを設定することができます。これらは各セットアップ関数を呼び出すときに設

定するほか、それぞれ専用の関数 `SND_SetChannelVolume`、`SND_SetChannelTimer`、`SND_SetChannelPan` を呼び出して、タイマーがスタートした後も値を変更することができます。

5 サウンドキャプチャ

5.1 サウンドキャプチャの概要

ニンテンドーDS にはサウンドキャプチャ機能が2基備わっています。

番号	キャプチャ対象	タイマー
キャプチャ 0	ミキサーの左チャンネルまたはチャンネル 0 の出力をキャプチャします。	チャンネル 1 のタイマーを共用します。
キャプチャ 1	ミキサーの右チャンネルまたはチャンネル 2 の出力をキャプチャします。	チャンネル 3 のタイマーを共用します。

キャプチャを使用するときは、チャンネル 1 及び 3 のタイマーをキャプチャに共用しますので、チャンネル 1 及び 3 は、キャプチャしたデータを再び出力する用途に使えますが、任意のタイマー値を設定して発音することができなくなります。

5.2 サウンドキャプチャの使い方

キャプチャを使用するための手順はチャンネルを使用する時の手順に似ています。

- キャプチャで共用するタイマーを使用しているチャンネルをロックする
- `SND_SetupCapture` でキャプチャのパラメータをセットアップする
- `SND_SetChannelTimer` で、共用しているタイマーの周波数を設定する
もしくは、キャプチャしたデータを再生する場合は `SND_SetupChannelPcm` で設定する
- タイマーを開始する

キャプチャしたデータを演算処理して再び出力することによって、エフェクト効果を作り出すことができます。NITRO-Composer には、リバーブや出力エフェクトなどがサウンドキャプチャを使用して実装されています。

5.3 サウンドキャプチャの不具合について

NITRO 互換モードでは、サウンドキャプチャのハードウェアに不具合があり、チャンネル 0 または 2 の出力をキャプチャする時に、ただしくキャプチャ出来ないことがあります。

不具合の詳細は「TWL プログラミングマニュアル」を参照してください。

6 サウンドアラーム

6.1 サウンドアラームの概要

サウンドアラームは ARM7 のタイマーを使用したアラームシステムです。サウンドアラームを使用すると、チャンネルの発音やキャプチャなどと同期した処理を行うことができます。サウンドアラームは 0 番から 7 番まで、同時に 8 個使用することができます。

6.2 サウンドアラームの使い方

サウンドアラームを使用する手順は以下のようになります。

- SND_SetupAlarm を呼びだして使用するサウンドアラームのセットアップを行う
- SND_StartTimer を呼びだしてサウンドアラームを開始する

SND_StartTimer は、チャンネル、サウンドキャプチャ、サウンドアラームを同時に開始できます。同期させたいチャンネルやサウンドキャプチャと一緒にサウンドアラームを開始してください。

7 ドライバ情報取得

7.1 ドライバ情報取得の概要

ドライバ情報取得のための関数群を利用して、現在のドライバの状態を知ることができます。ただし、ARM7 で処理されている情報を取得するため、ARM9 との同期について注意する必要があります。

7.2 情報構造体の取得

ARM7 で処理されているチャンネル、プレイヤー、トラックの現在の状態を取得することができます。同期を取るために以下の手順で情報を取得します。

- SND_ReadDriverInfo を呼び出してドライバ情報を取得する
この関数はコマンド予約関数ですので、取得した情報にアクセスするためにはコマンドをフラッシュし、コマンドの実行完了を待つようにしてください。
- チャンネル、プレイヤー、トラックの情報を取得する関数(SND_ReadChannelInfo、SND_ReadPlayerInfo、SND_ReadTrackInfo)を呼び出す

以下のコードは、ドライバ情報の構造体を取得し、その場でコマンド完了を待ってから各種情報を取得する例です。

コード 7-1 ドライバ情報構造体の取得

```
u32 tag;
SNDDriverInfo driverInfo;
SNDChannelInfo channelInfo;
SNDPlayerInfo playerInfo;
SNDTrackInfo trackInfo;

/* ドライバ情報の取得完了まで待つ */
SND_ReadDriverInfo( &driverInfo );
tag = SND_GetCurrentCommandTag( );
SND_FlushCommand( SND_COMMAND_BLOCK );
SND_WaitForCommandProc( tag );

/* チャンネル0の情報を取得 */
SND_ReadChannelInfo( &driverInfo, 0, &channelInfo );

/* プレイヤー1の情報を取得 */
SND_ReadPlayerInfo( &driverInfo, 1, &playerInfo );

/* プレイヤー0のトラック3の情報を取得 */
SND_ReadTrackInfo( &driverInfo, 0, 3, &trackInfo );
```

7.3 その他の情報取得

構造体取得関数以外にも、SND_ReadDriverInfo を呼び出さずに使用できる情報取得関数があります。

これらの関数は ARM7 コマンド関数とは非同期に実行されます。したがって、例えば SND_StartTimer を実行してから SND_GetPlayerStatus を呼び出したときに 0 が帰ってきた場合、その原因が、コマンドがまだ完了していないためにチャンネルがアクティブになっていないためなのか、再生が終了してしまってアクティブになっていないためなのか

判断がつきません。

同期をとって情報を取得するためには、コマンドタグを使用して `SND_WaitForCommandProc` を呼び出すなど、`ARM7` コマンドの処理が完了したことを確認するようにしてください。

<code>SND_GetPlayerStatus</code>	プレイヤーがアクティブかどうかを取得します
<code>SND_GetChannelStatus</code>	チャンネルがアクティブかどうかを取得します
<code>SND_GetCaptureStatus</code>	サウンドキャプチャがアクティブかどうかを取得します
<code>SND_GetPlayerLocalVariable</code>	シーケンスローカル変数を取得します
<code>SND_GetPlayerGlobalVariable</code>	シーケンスグローバル変数を取得します。
<code>SND_GetPlayerTickCounter</code>	シーケンスティックカウンタを取得します。

8 NITRO-Composerと併用する際の注意点

8.1 プレイヤーの使用

NITRO-Composer を使用してシーケンスデータを再生する場合、サウンドドライバからプレイヤーを操作することができません。つまり、NITRO-Composer のシーケンス再生とサウンドドライバのシーケンス再生を同時に使用することはできません。

8.2 チャンネルの使用

サウンドドライバのチャンネルをロックする関数 `SND_LockChannel` 及びロックを解除する関数 `SND_UnlockChannel` は、サウンドドライバのシーケンス再生に対してのみロックをかけることができます。

NITRO-Composer を使用している場合にチャンネルのロック、アンロックを行いたい場合は、NITRO-Composer の関数 `NNS_SndLockChannel` 及び `NNS_SndUnlockChannel` を使用してください。

8.3 サウンドキャプチャの使用

NITRO-Composer を使用している場合にサウンドドライバのサウンドキャプチャを使用するには、NITRO-Composer の関数 `NNS_SndLockCapture` を呼び出して、NITRO-Composer がキャプチャを使用しないようにしてください。

それと同時に、`NNS_SndLockChannel` を呼び出して、チャンネル 1 とチャンネル 3 をロックする必要があります。

8.4 サウンドアラームの使用

NITRO-Composer は内部でサウンドアラームを使用しますので、NITRO-Composer を使用している場合にサウンドアラームを使うには、NITRO-Composer の関数 `NNS_SndAllocAlarm` を呼び出して、使用可能なアラーム番号を取得してください。取得した番号のアラームは、NITRO-Composer 内部では使用しません。

使用が終わったアラームは `NNS_SndFreeAlarm` で解放してください。

© 2005–2008 Nintendo

任天堂株式会社の許諾を得ることなく、本書に記載されている内容の一部あるいは全部を無断で複製・複写・転写・頒布・貸与することを禁じます。