

NITRO-Composer サウンドデザイナーガイド

Ver 1.5.0

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	はじめに	7
2	サウンドデータの概要	7
2.1	サウンドデータの構成	7
2.1.1	波形データ	7
2.1.2	バンクデータ	7
2.1.3	シーケンスデータ	7
2.1.4	ストリームデータ	7
2.2	サウンドアーカイブ	7
2.3	データ作成の流れ	8
2.3.1	データ作成フロー	8
2.3.2	データ作成の手順	8
3	サウンドアーカイブ	10
3.1.1	サウンドアーカイブ定義ファイル	10
3.1.2	@PATH	11
4	シーケンスデータ	12
4.1	シーケンスデータの登録	12
4.1.1	バンクの指定	12
4.1.2	プレイヤーの指定	13
4.1.3	その他のパラメータ	13
4.1.4	ラベルについて	13
4.2	シーケンスデータの作成	13
5	シーケンスアーカイブ	14
5.1	シーケンスアーカイブとは?	14
5.1.1	普通のシーケンスファイルの特徴	14
5.1.2	シーケンスアーカイブの特徴	14
5.2	シーケンスアーカイブの登録	15
5.3	シーケンスアーカイブの作成	15
5.3.1	シーケンステーブル	16
5.3.2	シーケンスデータの登録	16
5.3.3	シーケンスデータの作成	17
5.3.3.1	単純な波形再生の例	17
5.3.3.2	単純なシーケンス再生の例	17
5.3.4	音程表記	18
6	バンクデータ	19
6.1	バンクデータの登録	19
6.1.1	波形アーカイブとの関連づけ	19
6.1.2	バンクデータの登録の例	19
6.1.3	各サウンドデータの相互関係	21

6.2	バンクデータの作成	21
6.2.1	オリジナルキー	22
7	メモリ管理	23
7.1	2種類のヒープ	23
7.1.1	サウンドヒープ	23
7.1.2	プレイヤーヒープ	23
7.2	グループの作成	23
7.3	グループのロード	25
7.4	プレイヤーヒープの作成	25
7.5	プレイヤーヒープへのロード	25
7.6	プレイヤーヒープサイズの決定	26
7.6.1	サイズ決定の注意	26
8	ストリーム再生	27
8.1	ストリームデータの登録	27
8.1.1	プレイヤー番号の指定	27
8.1.2	その他のパラメータ	27
8.2	ストリームプレイヤーの定義	28
8.3	ストリームデータの作成	28
9	SoundPlayer	29
9.1	SoundPlayerの使い方	29
9.1.1	MakeSound.bat	29
9.2	サウンドの再生	30
9.2.1	複数サウンド同時再生	30
9.3	サウンドタイプの選択	31
9.4	画面表示	31
9.4.1	ラベル名表示	31
9.4.2	パラメータ表示	31
9.4.3	シーケンスアーカイブ番号	32
9.5	チャンネルメーター	32
9.6	リアルタイムMIDI再生	32
9.6.1	リアルタイムMIDI再生とは	32
9.6.2	準備	33
9.6.3	バンクの選択	33
9.6.4	MIDI信号送信	33
9.7	手動ロードモード	33
9.7.1	手動ロードモードとは	33
9.7.2	手動ロードモードでの動作	34
9.7.3	モードの切り替え	34
9.7.4	データのロード	34
9.7.5	メモリ表示	34
9.8	キー操作	35

9.9	エラーメッセージ.....	35
10	エラーの対処.....	37
10.1	エラーの発生.....	37
10.2	原因の特定.....	37
10.3	エラーの対処.....	37
10.4	対処の確認.....	37
11	次のステップ.....	39

コード

コード 3-1	サウンドアーカイブ定義ファイル.....	10
コード 3-2	シーケンスデータ登録.....	11
コード 4-1	シーケンスデータセクション.....	12
コード 4-2	シーケンスデータセクション（シーケンスの追加）.....	12
コード 5-1	シーケンスアーカイブセクション.....	15
コード 5-2	テキストシーケンスアーカイブ.....	15
コード 5-3	シーケンステーブル.....	16
コード 6-1	バンクデータセクション.....	19
コード 6-2	波形アーカイブセクション.....	19
コード 6-3	バンクデータ登録の例.....	20
コード 6-4	バンクデータ登録の例（その2）.....	20
コード 6-5	バンク定義ファイル.....	21
コード 7-1	グループ情報セクション.....	24
コード 7-2	プレイヤー情報セクション.....	25
コード 8-1	ストリームデータセクション.....	27
コード 8-2	ストリームプレイヤー情報セクション.....	28

表

表 5-1	ナチュラル/シャープ/フラット.....	18
表 9-1	SoundPlayerのキー操作.....	35
表 9-2	SoundPlayerエラーメッセージ.....	35

図

図 2-1	データ作成フロー.....	8
図 5-1	シーケンスアーカイブの概念図.....	14
図 5-2	音程表記.....	18
図 6-1	各サウンドデータの相互関係の例.....	21
図 7-1	グループ設定とサウンドデータの関係.....	24
図 7-2	グループ設定とサウンドデータの関係.....	25
図 9-1	SoundPlayer画面.....	29
図 9-2	SoundPlayer上画面（シーケンスアーカイブ番号）.....	30
図 9-3	SoundPlayer上画面（パラメータ表示）.....	31

図 9-4 SoundPlayerチャンネルメーター	32
図 9-5 SoundPlayerチャンネルメーター(赤表示)	32
図 9-6 SoundPlayer画面(リアルタイムMIDI)	33
図 9-7 SoundPlayer画面(手動ロードモード)	34

改訂履歴

版	改訂日	改 訂 内 容	承認者	担当者
1.5.0	2007-03-14	1. SoundPlayer の複数サウンド同時再生機能に関する説明追加		井田
1.4.0	2005-01-31	1. 音程表記における大文字小文字の区別を廃止したことに伴う説明追加 2. NITRO-Player に関する説明追加 3. SoundPlayer チャンネルメーターに関する説明追加 4. サンプルデータ中の se.mus 変更に伴う修正 5. "NITRO"を"ニンテンドーDS"に変更		井田
1.3.0	2004-10-12	1. プレイヤーの指定にラベルが使えるようになったことに伴う修正 2. SoundPlayer が IS-AGB-MIDI 対応になったことに伴う修正		井田
1.2.1	2004-09-16	1. SoundPlayer エラーメッセージスペルミス修正 2. *.sadl ファイルの呼称を「サウンドラベルファイル」に統一		井田
1.2.0	2004-09-02	1. SoundPlayer リアルタイム MIDI 機能に関する説明追加		井田
1.1.0	2004-08-10	1. ストリームに関する説明追加		井田
1.0.0	2004-07-20	1. 全体的な構成の変更 2. SoundPlayer.bin を SoundPlayer.srl に変更 3. SoundPlayer にシーケンスタイプ選択機能追加 4. SoundPlayer エラーメッセージの追加 5. サンプルデータ中の SE 用波形アーカイブを削除 6. バンクと波形アーカイブの関連づけの自由度が上がったことに伴う修正 7. サウンドマップファイルの説明を「サウンドアーカイブマニュアル」に移動		井田
0.5.0	2004-06-01	1. ファイルシステム対応に伴う修正 2. SoundPlayer 開発環境変更に伴う修正 3. サウンドマップファイルの説明追加 4. シーケンスアーカイブ概念図追加 5. サウンドアーカイブ中の「登録ブロック」の名称を「セクション」に変更 6. 各サウンドデータの相互関係の図を微修正		井田
0.4.0	2004-04-12	1. エラーの対処追加		井田
0.3.0	2004-04-01	1. SoundPlayer の使い方変更に伴う修正 2. サウンドラベルリストに関する説明追加 3. 音程表記でフラットについて加筆 4. シーケンスアーカイブについて説明補足 5. サウンドデータのロード管理について説明補足		井田
0.2.0	2004-03-18	シーケンスフォーマット変更に伴う修正		井田
0.1.0	2004-03-01	初版		井田

1 はじめに

本ドキュメントではサウンドデータ制作者向けに、サウンドデータ作成の始め方について説明しています。まず、サウンドデータの概要について説明し、あとでサンプルデータを見ながら具体的に説明していきます。また最後に、サウンドをニンテンドーDS で再生確認できるツール「SoundPlayer」について、詳しく説明します。

2 サウンドデータの概要

2.1 サウンドデータの構成

NITRO-Composer で扱うサウンドデータは、大きく 4 種類に分けられます。

- 波形データ
- バンクデータ
- シーケンスデータ
- ストリームデータ

以下、それぞれについて、簡単に説明していきます。

2.1.1 波形データ

AIFF または WAV フォーマットのモノラル波形ファイルが扱えます。後述のバンクデータに関連づけるために、複数の波形データを1つにまとめた波形アーカイブという形で使用します。

2.1.2 バンクデータ

音源データに相当します。各プログラムナンバ毎に、波形ファイルを割り当てて音源を構成します。バンクデータは、テキストファイルに記述します。

2.1.3 シーケンスデータ

フォーマット 0 または 1 の SMF(スタンダード MIDI ファイル)が扱えます。また、1 つのテキストファイルに複数のシーケンスを記述する、シーケンスアーカイブという形式のシーケンスデータを扱うこともできます。

2.1.4 ストリームデータ

ストリーム方式で再生するための波形データです。AIFF または WAV フォーマットのモノラルまたはステレオ波形が扱えます。

2.2 サウンドアーカイブ

NITRO-Composer では、上記のサウンドデータを個々に扱わずに、全てのサウンドデータを1つのサウンドアーカイブというファイルにまとめて扱うことができます。サウンドアーカイブを使うことで、データの管理が単純になり、メモリ効率も

上がります。

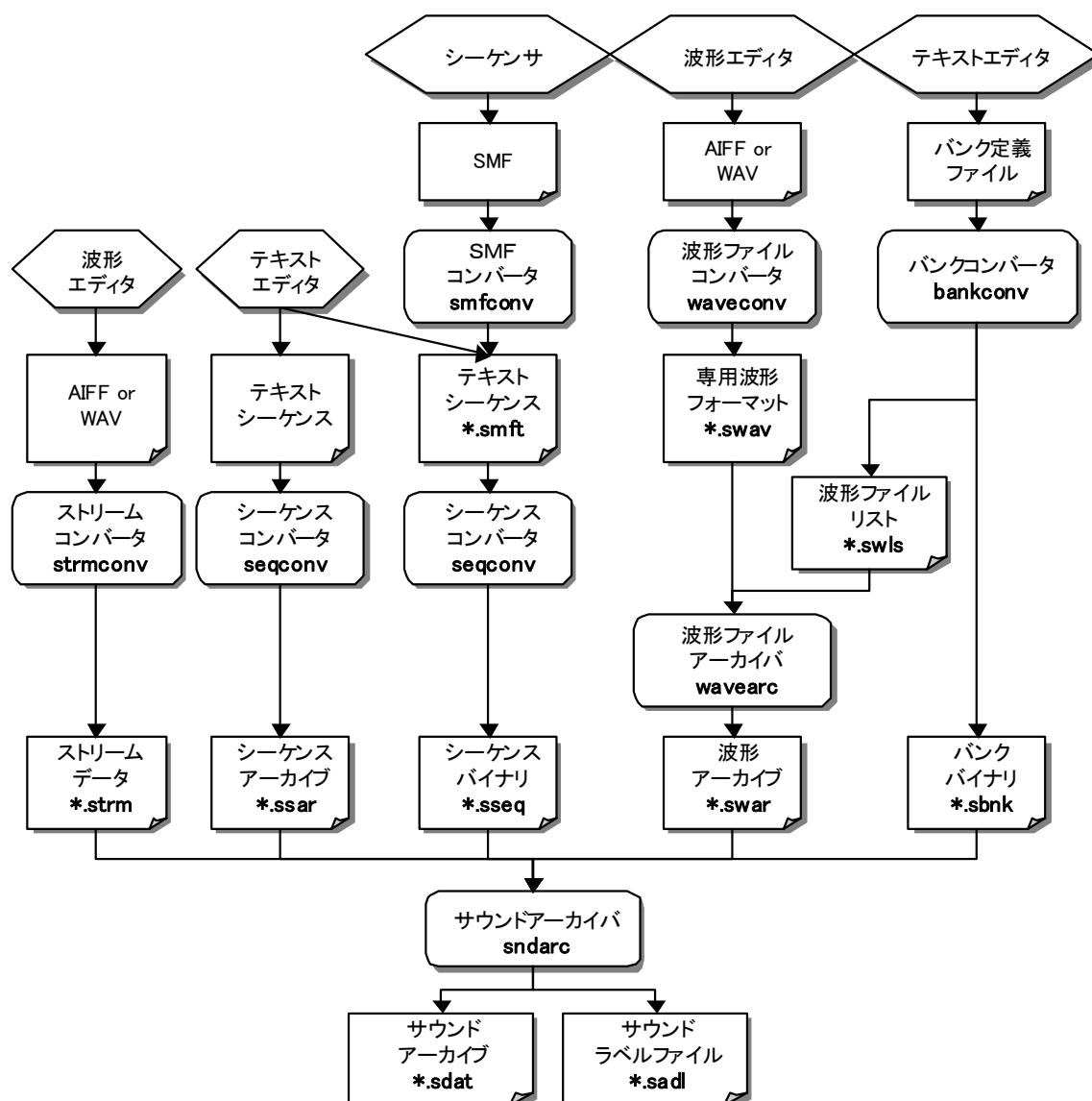
最終的にはこのファイルを作ることがサウンドデザイナーの役割になります。

2.3 データ作成の流れ

2.3.1 データ作成フロー

データ作成の流れは、図で表すと下のようになります。

図 2-1 データ作成フロー



2.3.2 データ作成の手順

上の図のように、サウンドデータの作成には何段階かの手順が必要になります。個々の動作については、あとでサンプル

ルに触りながら説明しますが、実作業において、これらを全て理解する必要はありません。

最終出力であるサウンドアーカイブを生成するツール「サウンドアーカイバ **sndarc**」を使うだけで、その他の変換処理も全て同時に実行されます。また、ファイルの更新日時を比較して、必要最低限のファイルのみを変換しますので、変換時間もあまりかかりません。

サウンドデザイナーは、最終出力「サウンドアーカイブ(*.sdar)」と「サウンドラベルファイル(*.sdl)」をサウンドプログラマーに渡します。「サウンドアーカイブ」は全サウンドデータをまとめたファイルで、これを ROM に格納して音を鳴らします。「サウンドラベルファイル」はシーケンスデータなどの番号がラベルとして定義されたファイルで、プログラムでインクルードして使います。

3 サウンドアーカイブ

それでは、今までの説明の中で出てきたサウンドアーカイブを実際に見てみましょう。

以下、\$NitroSystem/tools/nitro/SoundPlayer/data ディレクトリにあるファイルを元に説明していきます。以下このディレクトリを、\$data と記述します。実作業では、この\$data ディレクトリを別の場所にコピーしてから編集していくと、作業をスムーズに始めることができます。

3.1.1 サウンドアーカイブ定義ファイル

サウンドアーカイブを作成するためには、サウンドアーカイブ定義ファイルを作成する必要があります。サウンドアーカイブ定義ファイルは、テキストファイルなので、テキストエディタで作成します。サンプルでは、\$data/sound_data.sarc がサウンドアーカイブ定義ファイルなので、テキストエディタで開いてみてください。

コード 3-1 サウンドアーカイブ定義ファイル

```
;/=====
;
; NITRO-Composer sample
;
;/=====

;;;;;;;;;;;;;
;; Wave Archive

@WAVEARC
@PATH "swar"
WAVE_SE      : AUTO, "se.swls"
WAVE_BGM     : AUTO, "bgm.swls"

;;;;;;;;;;;;;
;; Bank

@BANK
@PATH "bnk"
BANK_SE      : TEXT, "se.bnk", WAVE_SE
BANK_BGM     : TEXT, "bgm.bnk", WAVE_BGM

;;;;;;;;;;;;;
;; Player

@PLAYER
PLAYER_BGM   : 1, 8000
PLAYER_SE    = 10 : 1
PLAYER_VOICE : 1

;;;;;;;;;;;;;
;; Sequence

@SEQ
@PATH "mid"
SEQ_MARIOKART : SMF, "kart64_title.mid", BANK_BGM, 127, 64, 64, PLAYER_BGM

;;;;;;;;;;;;;
;; Sequence Archive
```

```

@SEQARC
@PATH "mus"
SEQ_SE : TEXT, "se.mus"

;;;;;;;;;;
;; Stream Player

@STRM_PLAYER
PLAYER_STRM : STEREO, 6, 7

;;;;;;;;;;
;; Stream

@STRM
@PATH "strm"
STRM_MARIOKART : PCM8, "kart_title.32.aiff", 127, 64, PLAYER_STRM
STRM_FANFARE : PCM8, "fanfare.32.aiff", 127, 64, PLAYER_STRM

;;;;;;;;;;

@GROUP

GROUP_STATIC :
SEQ_SE
BANK_SE
BANK_BGM

```

セミコロンで始まる行はコメントです。NITRO-Composer で扱うテキストファイルは全て、セミコロンで始まる行をコメントと見なします。

まずは、次の箇所に注目してください。

コード 3-2 シーケンスデータ登録

```

;;;;;;;;;;
;; Sequence

@SEQ
@PATH "mid"
SEQ_MARIOKART : SMF, "kart64_title.mid", BANK_BGM, 127, 64, 64, PLAYER_BGM

```

ここは、シーケンスデータを登録している箇所です。ここでは、"kart64_title.mid"というファイル名のスタンダード MIDI ファイルを登録しています。

次の章ではこの箇所について、もう少し詳しく説明します。

3.1.2 @PATH

ここで、ちょっと注釈です。

サウンドアーカイブ定義ファイル中で、@PATH で始まる記述があります。

```

@PATH "mid"
SEQ_MARIOKART : SMF, "kart64_title.mid", BANK_BGM, 127, 64, 64, 0

```

これは、"kart64_title.mid"というファイルが存在するディレクトリを指定しています。つまり実際には、"mid/kart64_title.mid"というファイルを登録していることとなります。

4 シーケンスデータ

この章では、シーケンスデータの登録と作成について説明します。

4.1 シーケンスデータの登録

まずは、シーケンスデータの登録の仕方について説明します。

サウンドアーカイブ定義ファイル中で、@SEQ ではじまる箇所が、シーケンスデータを登録するシーケンスデータセクションです。

コード 4-1 シーケンスデータセクション

```
;;;;;;;;;;  
;; Sequence  
  
@SEQ  
  @PATH "mid"  
SEQ_MARIOKART : SMF, "kart64_title.mid",      BANK_BGM, 127, 64, 64, 0
```

SEQ_MARIOKART で始まる1行が、1つのシーケンスデータを登録しています。新しくシーケンスファイルを追加したい場合は、1行追加します。

コード 4-2 シーケンスデータセクション（シーケンスの追加）

```
;;;;;;;;;;  
;; Sequence  
  
@SEQ  
  @PATH "mid"  
SEQ_MARIOKART      : SMF, "kart64_title.mid", BANK_BGM, 127, 64, 64, PLAYER_BGM  
SEQ_NEW_SEQUENCE : SMF, "new.mid",          BANK_MK64, 127, 64, 64, PLAYER_BGM
```

新しく追加されたシーケンス"new.mid"が、2番目のシーケンスデータとして登録されます。

SEQ_NEW_SEQUENCE というラベルは、このシーケンスを指定するために使うラベルです。プログラムからこのシーケンスを再生しようとするときに、このラベルを使って指定することができます。

4.1.1 バンクの指定

シーケンスを再生するためには、バンクデータが必要です。そのため、どのバンクデータを使って再生するのかを指定する必要があります。

上の SEQ_MARIOKART のシーケンスの例では、BANK_BGM というバンクを使うように指定しています。このラベルは、サウンドアーカイブ定義ファイルでバンクデータを登録するときに定義されたラベルで、1つのバンクデータを表しています。バンクデータの登録については後で説明します。

バンクデータが間違っていると、変な音で鳴ったり、音が鳴らなかったりします。シーケンスデータを作るときには、どのバンクデータを使って再生するかを確認し、シーケンスデータセクションにて、正確に関連づける必要があります。

4.1.2 プレイヤーの指定

シーケンスは32個あるプレイヤーの内の1つを使って再生します。そのため、どのプレイヤーで再生するかを、指定する必要があります。

デフォルトでは1つのプレイヤーで同時に再生できるのは、1つのシーケンスだけになっています。そのため、同時に再生したいシーケンスは、別々のプレイヤーで再生する必要があります。プレイヤーに関する詳細は、「サウンドシステムマニュアル」をご覧ください。

上の例では、行の一番最後に書いてあるように `PLAYER_BGM` というプレイヤーを使うように指定しています。このラベルは、サウンドアーカイブ定義ファイル中で定義されたラベルで、1つのプレイヤーを表しています。また、プレイヤーラベルの代わりに、0~31 の数字で指定することもできます。プレイヤーラベルについて詳しくは、「サウンドアーカイブマニュアル」をご覧ください。

4.1.3 その他のパラメータ

その他のパラメータは、ボリュームやプライオリティなどの設定値ですが、ここでの説明は省略します。詳しくは、「サウンドアーカイブマニュアル」をご覧ください。

4.1.4 ラベルについて

上の例でもあったように、サウンドデータのテキストファイルで、ラベルを扱うことがあります。ラベルとは、特定のデータや場所を指し示すシンボルです。特定のデータや場所を指し示すためには、まずそれに対してラベルを定義し、別の場所でそのラベルを使って指定します。

ラベル名のフォーマットは、次のことに従う必要があります。まず1文字目は、アルファベットの太文字か小文字または、アンダースコア(`_`)でなければなりません。2文字目以降は、アルファベット太文字・小文字、アンダースコア(`_`)に加え、数字も使えます。

ただし、1文字目がアンダースコア(`_`)のラベルは、シーケンスデータにおいて特別な意味を持ちますので、とりあえずは使用しない方が良いでしょう。詳しくは、「シーケンスデータマニュアル」を参照してください。

次に示すものは、正しいラベル名の例です。

```
SEQ_TITLE_BGM
loop_start
Track_01
```

当然ですが、同じラベル名を別のものにつけることはできません。

4.2 シーケンスデータの作成

シーケンスデータは、スタンダード MIDI ファイルとして作成します。つまり、大抵の市販シーケンサで作成することができます。

使用可能な MIDI イベントなどの詳細については、「シーケンスデータマニュアル」をご覧ください。

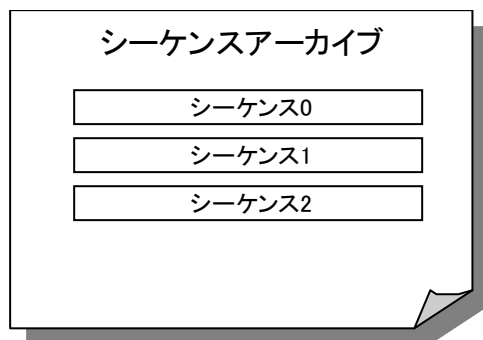
5 シーケンスアーカイブ

シーケンスデータを作成するとき、シーケンスアーカイブとして作成する方法も用意されています。ここでは、シーケンスアーカイブとは何かを説明し、その後でシーケンスアーカイブの登録と作成の方法について説明します。

5.1 シーケンスアーカイブとは？

シーケンスアーカイブとは、複数のシーケンスデータを1つのファイルにまとめたものです。ただし、サウンドアーカイブのように、別々のファイルでシーケンスデータを作ってから、1つにまとめるわけではありません。はじめから、1つのファイルに複数のシーケンスデータを記述して作成します。

図 5-1 シーケンスアーカイブの概念図



単なるシーケンスデータとシーケンスアーカイブとで、シーケンスデータ自体に機能的な差はありません。違うのはシーケンスデータ作成時のスタイル及び、データの管理方法です。

5.1.1 普通のシーケンスファイルの特徴

単体のシーケンスファイルの1番の特徴は、スタンダード MIDI ファイルから作成できることです。シーケンサで作成できることによって、リアルタイムに音を確認しながらシーケンスデータを作成することができます。

このシーケンスデータは通常、再生開始時毎にメインメモリにロードします。そのため、頻繁に再生を繰り返すようなシーケンスの場合、データロードに伴う負荷が大きくなる場合があります。ただし、そのようなシーケンスデータだけ、前もってロードしておくという使い方もできます。

主に BGM の作成に適しています。

5.1.2 シーケンスアーカイブの特徴

シーケンスアーカイブは、テキストエディタで作成します。そのため、一度コンバートしないと、音を確認することはできません。ただし、1つのファイルだけで、何百、何千個ものシーケンスデータを扱えるため、データの管理が楽になります。また、データのサイズも、普通のシーケンスファイルに比べて、小さくすることができます。

シーケンスアーカイブのデータは通常、起動時や場面切り替え時にメインメモリにロードし、ロード中は自由に再生できるような扱いになります。

主に、効果音の作成に適しています。

5.2 シーケンスアーカイブの登録

サウンドアーカイブ定義ファイルに、シーケンスアーカイブを登録する方法について説明します。

サウンドアーカイブ定義ファイル中で、@SEQARC ではじまる箇所が、シーケンスアーカイブを登録するシーケンスアーカイブセクションです。

コード 5-1 シーケンスアーカイブセクション

```
;;;;;;;;;;
;; Sequence Archive

@SEQARC
  @PATH "mus"
  SEQ_SE : TEXT, "se.mus"
```

SEQ_SE で始まる1行が、1つのシーケンスアーカイブを登録しています。つまり、実際にはこれだけで複数のシーケンスデータを登録していることになります。

同じような行を追加することで、シーケンスアーカイブを複数登録することもできます。

5.3 シーケンスアーカイブの作成

シーケンスアーカイブは、シーケンサではなく、テキストエディタで作成します。

\$data/mus/se.mus が、テキスト形式のシーケンスアーカイブです。中身を見てみましょう。

コード 5-2 テキストシーケンスアーカイブ

```
;;;;;;;;;;
;      SeqArc for Sample SE
;;;;;;;;;;

#include <sound_data.sbd1>

@SEQ_TABLE

SE_YOSHI:          yoshi,          BANK_SE, 127, 96, 64, PLAYER_VOICE
SE_WIHAHO:          wihaho,         BANK_SE, 127, 96, 64, PLAYER_VOICE
SE_COIN:            note_only,      BANK_SE, 65, 96, 64, PLAYER_SE
SE_AMBULANCE:       jump_seq,       BANK_SE, 55, 96, 64, PLAYER_SE
SE_REPEAT:          loop_seq,       BANK_SE, 55, 96, 64, PLAYER_SE
SE_PATTERN:         call_seq,       BANK_SE, 55, 96, 64, PLAYER_SE
SE_PORTAMENT:       porta_seq,      BANK_SE, 65, 96, 64, PLAYER_SE
SE_PORTAMENT2:      porta_time_seq, BANK_SE, 65, 96, 64, PLAYER_SE
SE_SWEEP:           sweep_seq,      BANK_SE, 65, 96, 64, PLAYER_SE
SE_VIBRATE:         mod_seq,        BANK_SE, 65, 96, 64, PLAYER_SE
SE_VIBRATE2:        tie_seq,        BANK_SE, 65, 96, 64, PLAYER_SE
SE_SUPER_MARIO:     waitoff_seq,    BANK_SE, 65, 96, 64, PLAYER_SE
SE_SUPER_MARIO2:    opentrack_seq,  BANK_SE, 65, 96, 64, PLAYER_SE

@SEQ_DATA

yoshi:
  prg 0
  cn4 127, 0
  fin
```

```
wihaho:
  prg 1
  cn4 127, 0
  fin
```

まず、`#include <sound_data.sddl>`という記述で始まります。これは、サウンドアーカイブラベルファイル(`sound_data.sddl`)を取り込むことを意味します。サウンドアーカイブラベルファイルは、コンバート時に自動的に生成されるファイルで、このファイルを取り込むことにより、サウンドアーカイブ定義ファイル中で定義したラベルが使用できるようになります。(ラベルを使用せずに番号で指定する場合は、この記述は不要です)

残りの部分は大きく2つに分けられます。

- `@SEQ_TABLE` で始まる、シーケンステーブル
- `@SEQ_DATA` で始まる、シーケンスデータ

以下、それぞれについて説明します。

5.3.1 シーケンステーブル

シーケンステーブルは、シーケンスアーカイブに登録されているシーケンスデータの一覧表です。シーケンステーブルは、`@SEQ_TABLE` で始まります。

コード 5-3 シーケンステーブル

```
@SEQ_TABLE
```

SE_YOSHI:	yoshi,	BANK_SE, 127, 96, 64, PLAYER_VOICE
SE_WIHAGO:	wihago,	BANK_SE, 127, 96, 64, PLAYER_VOICE
SE_COIN:	note_only,	BANK_SE, 65, 96, 64, PLAYER_SE
SE_AMBULANCE:	jump_seq,	BANK_SE, 55, 96, 64, PLAYER_SE
SE_REPEAT:	loop_seq,	BANK_SE, 55, 96, 64, PLAYER_SE
SE_PATTERN:	call_seq,	BANK_SE, 55, 96, 64, PLAYER_SE
SE_PORTAMENT:	porta_seq,	BANK_SE, 65, 96, 64, PLAYER_SE
SE_PORTAMENT2:	porta_time_seq,	BANK_SE, 65, 96, 64, PLAYER_SE
SE_SWEEP:	sweep_seq,	BANK_SE, 65, 96, 64, PLAYER_SE
SE_VIBRATE:	mod_seq,	BANK_SE, 65, 96, 64, PLAYER_SE
SE_VIBRATE2:	tie_seq,	BANK_SE, 65, 96, 64, PLAYER_SE
SE_SUPER_MARIO:	waitoff_seq,	BANK_SE, 65, 96, 64, PLAYER_SE
SE_SUPER_MARIO2:	opentrack_seq,	BANK_SE, 65, 96, 64, PLAYER_SE

シーケンステーブルで記述されている各行1つ1つが、1つのシーケンスデータを表しています。この並び順が、各シーケンスデータの番号になります。

5.3.2 シーケンスデータの登録

シーケンスデータを登録するためには、シーケンステーブルにつきのような1行追加します。

```
SE_NEW:                new_seq,                BANK_SE, 65, 96, 64, PLAYER_SE
```

`SE_NEW` というラベルは、このシーケンスを指定するためのものです。プログラムからこのシーケンスを再生するときには、このラベルを使って、シーケンスを指定することができます。

続いて、`new_seq` とあるラベルは、あとで説明するシーケンスデータに対してつけられたラベルです。つまり、`new_seq` というラベルが付けられたシーケンスデータが別の場所にあって、`SE_NEW` というシーケンスは、そのシーケンスデータを使って再生されるということです。

3番目に `BANK_SE` とあるのは、どのバンクを使って再生するのという指定です。先に述べたように、サウンドアーカイブラベルファイルを取り込んでいるので、サウンドアーカイブ定義ファイル中で定義したバンクラベルを使って、バンクを指定することができます。ラベルの代わりに番号で指定することもできます。

4番目以降の数字は、サウンドアーカイブ定義ファイルのシーケンスデータセクションで、1つのシーケンスデータ毎に設定していた値と同じものになります。つまり、一番最後のラベルは、プレイヤーを表しています。その他の数字については、「シーケンスデータマニュアル」をご覧ください。

5.3.3 シーケンスデータの作成

5.3.3.1 単純な波形再生の例

@SEQ_DATA 以降に、シーケンスデータを記述します。

```
yoshi:
  prg 0
  cn4 127, 0
  fin
```

yoshi: は、シーケンスのラベル定義です。シーケンステーブルでは、このラベルを使ってシーケンスデータを指定します。

それ以降は、シーケンスコマンド列になっています。1つのシーケンスコマンドは、次のような書式になっています。

コマンド 引数, 引数, ...

引数の数や意味は、コマンドの種類によって決まっています。

`prg 0` は、プログラムチェンジコマンドです。シーケンステーブルで指定したバンクの、プログラムナンバ `0` に設定します。

その次の `cn4 127, 0` は、ノートコマンドです。`cn4` すなわち `C4`(ナチュラル)の音程で、ベロシティ `127` の音を鳴らします。最後の `0` は音の長さです。音の長さが `0` とは奇妙ですが、長さが `0` の場合は無限長として解釈され、波形データの最後まで再生することになります。波形データを単純にそのまま再生したければ、この例のように音の長さを `0` とします。

また、ここでは音程を `C4` としましたが、この音程は、後で説明するバンク定義ファイルで設定する、波形データのオリジナルキーに依存します。オリジナルキーと同じ音程で発音すると、波形データそのままの音程で発音します。この例では、バンク定義ファイルでオリジナルキーを `C4` としていますので、波形データそのままの音程で再生されます。

最後の `fin` コマンドは、シーケンスの終了を表します。これを忘れると、そのまま下のシーケンスデータを処理してしまいますので、注意してください。

5.3.3.2 単純なシーケンス再生の例

もう1つの例を見てみましょう。

```
note_only:
  prg 2
  as5 127, 6
  ds6 127, 48
  fin
```

先ほどの例と違って、ノートコマンドが2つあります。また音の長さが `0` ではありません。これはどのように発音されるのでしょうか？

まず1つ目のノートコマンドは、`as5` すなわち `A#5` の音程で鳴ります。音の長さは、4分音符分解能が `48` なので、`6` と

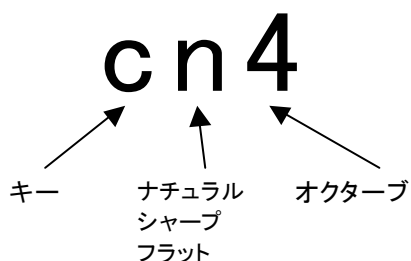
言うことは 32 分音符に相当します。なお音の発音中は、シーケンス処理が止まりますので、次のノートコマンドは A#5 の発音が終わってから処理されます。よって、A#5,D#6 の順番で音が鳴ります。(ここでは説明しませんが、音の発音中もシーケンスを止めずに、同時に3音鳴らすようなこともできます。)

その他のシーケンスコマンドについては、「シーケンスデータマニュアル」をご覧ください。

5.3.4 音程表記

ここで、シーケンスのノートコマンドで出てきた音程表記について説明します。NITRO-Composer では、ある音程を表すために次のような表記を使います。

図 5-2 音程表記



キーは、ド・レ・ミ・ファ・ソ・ラ・シを表します。それぞれ、c・d・e・f・g・a・b と書きます。

2文字目は、ナチュラル/シャープ/フラットを表し、下表の文字から選択します。ただし、この文字は省略して、"c4"のように書くこともできます。この場合、"cn4"と同じ意味になります。

表 5-1 ナチュラル/シャープ/フラット

文字	説明
n	ナチュラル
s	シャープ(半音上がる)
fまたはb	フラット(半音下がる)

オクターブは、数字で表します。オクターブが4の時センターになります。すなわち、an4 が 440Hz になります。マイナスの時は、マイナス記号の代わりに"m"と書きます。例えば、cnm1 のようになります。

音程の範囲は、cnm1 から gn9 までとなります。

なお、音程表記においてアルファベットの太文字小文字の区別はありません。例えば、"Cn4"や"CN4"の様な記述も有効です。

6 バンクデータ

シーケンスを再生するためにはバンクデータが必要です。ここでは、バンクデータの登録と作成の方法について説明します。

6.1 バンクデータの登録

まずは、バンクデータの登録の仕方について説明します。

サウンドアーカイブ定義ファイル中で、@BANK ではじまる箇所が、バンクデータを登録するバンクデータセクションです。

コード 6-1 バンクデータセクション

```
;;;;;;;;;;;;;;  
;; Bank  
  
@BANK  
  @PATH "bnk"  
  BANK_SE= 0      : TEXT, "se.bnk",   WAVE_SE  
  BANK_BGM       : TEXT, "bgm.bnk",  WAVE_BGM
```

BANK_SE などではじまる1行が、1つのバンクデータを登録しています。ここで定義したラベルを使って、バンクデータはシーケンスデータと関連づけられます。

6.1.1 波形アーカイブとの関連づけ

BANK_SE の最後に書いてある WAVE_SE は、このバンクが使用する波形データを指定したものです。WAVE_SE はサウンドアーカイブ定義ファイルの波形アーカイブセクションで定義されるラベルで、1つの波形アーカイブを表します。

波形アーカイブセクションは、次のようになっています。

コード 6-2 波形アーカイブセクション

```
;;;;;;;;;;;;;;  
;; Wave Archive  
  
@WAVEARC  
  @PATH "swar"  
  WAVE_SE      : AUTO, "se.swls"  
  WAVE_BGM     : AUTO, "bgm.swls"
```

波形アーカイブとは、複数の波形データを1つにまとめたものです。波形データはバンクデータに比べて、非常にデータサイズが大きくなるため、バンクデータと波形アーカイブは別々に扱います。別々に扱えるので、メモリをより効率的に使うことができるようになりますが、1つのバンクを登録するためには、バンクの登録と波形アーカイブの登録が必要になります。

6.1.2 バンクデータの登録の例

では、実際に1つのバンクデータを登録する例を見てみましょう。

コード 6-3 バンクデータ登録の例

```
;;;;;;;;;;  
;; Wave Archive  
  
@WAVEARC  
  @PATH "swar"  
WAVE_SE      : AUTO, "se.swls"  
WAVE_BGM     : AUTO, "bgm.swls"  
WAVE_NEW    : AUTO, "new.swls"  
  
;;;;;;;;;;  
;; Bank  
  
@BANK  
  @PATH "bnk"  
BANK_SE= 0    : TEXT, "se.bnk",  WAVE_SE  
BANK_BGM     : TEXT, "bgm.bnk",  WAVE_BGM  
BANK_NEW    : TEXT, "new.bnk", WAVE_NEW
```

太字の行が追加した行です。

BANK_NEW というラベルで、"bnk/new.bnk"というバンクファイルを登録しています。使用している波形アーカイブは、WAVE_NEW です。

WAVE_NEW の定義のところで、"swar/new.swls"というファイル名が書いてありますが、このファイルを作成する必要がありません。その前に"AUTO"書いてあるように、このファイルは、"自動"で生成されます。

別の方法として、次のように書くこともできます。

コード 6-4 バンクデータ登録の例(その2)

```
;;;;;;;;;;  
;; Wave Archive  
  
@WAVEARC  
  @PATH "swar"  
WAVE_SE      : AUTO, "se.swls"  
WAVE_BGM     : AUTO, "bgm.swls"  
  
;;;;;;;;;;  
;; Bank  
  
@BANK  
  @PATH "bnk"  
BANK_SE= 0    : TEXT, "se.bnk",  WAVE_SE  
BANK_BGM     : TEXT, "bgm.bnk",  WAVE_BGM  
BANK_NEW    : TEXT, "new.bnk", WAVE_BGM
```

BANK_NEW も、BANK_BGM と同じ、波形アーカイブ WAVE_BGM を使うように指定しています。つまり、WAVE_BGM を、BANK_NEW と BANK_BGM とで共用していることになります。

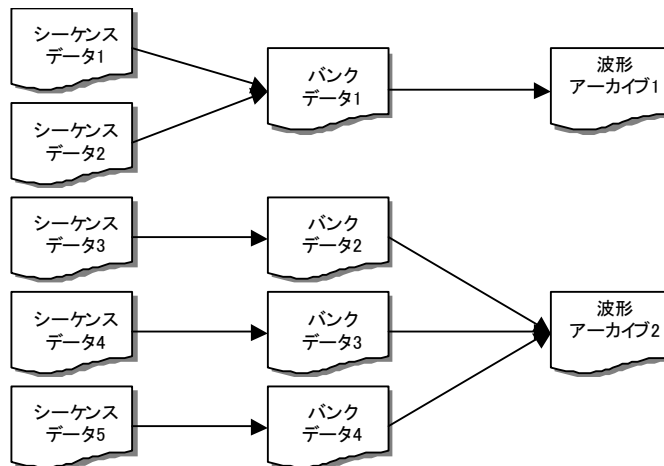
このように指定した場合、WAVE_BGM には、BANK_NEW で使っている波形データと、BANK_BGM で使っている波形データの両方が含まれることになります。たとえば、これら2つのバンクで使っている波形データがほとんど同じだった場合は、共用した方が効率がよくなります。

6.1.3 各サウンドデータの相互関係

ここで、シーケンスデータ、バンクデータ、波形アーカイブの相互関係についておさらいします。

次の図は、サウンドデータの相互関係の例です

図 6-1 各サウンドデータの相互関係の例



1つのシーケンスデータに対し、1つのバンクデータが関連づけられます。ただし、1つのバンクデータを別のシーケンスデータが使うことも可能です。

また、1つのバンクデータに対し、1つの波形アーカイブが関連づけられます。ただし、1つの波形アーカイブを別のバンクデータが使うことも可能です。例えば、別のバンクで同じ波形ファイルを使っている、波形アーカイブを共有化させておけば、メモリを2重に消費することはありません。

実はバンクと波形アーカイブの関連づけ方には、もっと複雑な方法も指定できるようになっています。それを使えば、メモリ効率を上げることができますが、管理が難しくなりますので、ここでの説明は省略します。詳しくは、「バンクデータマニュアル」及び「サウンドアーカイブマニュアル」をご覧ください。

6.2 バンクデータの作成

バンクデータは、テキストエディタでバンク定義ファイルを記述して作成します。

\$data/bnk/se.bnk がバンク定義ファイルです。中身を見てみましょう。

コード 6-5 バンク定義ファイル

```

;*****
;      Bank for Sample SE
;*****

@PATH "../aif"

@INSTLIST
0 : ADPCM, "yoshi.aiff",  cn4, 127, 127, 127, 125
1 : ADPCM, "wihaho.aiff", cn4, 127, 127, 127, 125
2 : PSG, DUTY_4_8, cn4,127,70,70,127
3 : PSG, DUTY_3_8, cn4,127,127,127,127
4 : NOISE, cn4,127,8,8,127

```

0, 1, 2 と並んでいる数字が、プログラムナンバを表します。

例では、プログラムナンバ 0 と 1 に ADPCM の文字と、ファイル名が記述されています。これらは、指定した波形ファイルを ADPCM にコンバートして使うことを意味しています。波形ファイルには、AIFF フォーマットまたは、WAV フォーマットのモノラルファイルを指定できます。

オリジナルキーは、波形データに対して設定せず、バンク定義ファイル中で、ファイル名の後に記述します。

うしろの4つの数字は、エンベロープの ADSR を表しています。

バンク定義ファイルについて詳しくは、「バンクデータマニュアル」をご覧ください。

6.2.1 オリジナルキー

オリジナルキーとは、波形データをそのまま再生したときの音程です。オリジナルキーを正しく設定することにより、シーケンスからのノートオンに対して、正しい音程変化を行うことができますようになります。

音声データのように音程と呼べるようなものが無い場合は、適当なキーを設定しておき、シーケンスから設定したものと同じキーで発音を行えば、波形データそのまの音程で発音されます。例えば、オリジナルキーを"cn4"としておくと、シーケンスデータからオリジナルキーと同じ"cn4"で発音すると、そのままの音程で再生されることになります。

なお、オリジナルキーの設定と、波形データのサンプリングレートとは無関係です。サンプリングレートを変更しても、オリジナルキーの設定を変更する必要はありません。

7 メモリ管理

ニンテンドーDS では、サウンドデータは通常 ROM に格納されます。ただし、サウンドデータを使用するためには、いったんメインメモリ上へロードしなければなりません。ここでは、サウンドデータをロードするために、サウンドデザイナーが設定すべきことについて説明します。

メモリ管理についての詳細は、「サウンドシステムマニュアル」をご覧ください。

7.1 2種類のヒープ

NITRO-Composer では、2種類のヒープが用意されています。ここでヒープとは、サウンドデータをロードするためのメモリ領域のことを指します。

- サウンドヒープ
- プレイヤーヒープ

以下、2種類のヒープそれぞれについて、簡単に説明します。

7.1.1 サウンドヒープ

サウンドヒープは、ゲーム中いつでも必要なデータや、特定の場面で必要なデータなど、必要となる場面が前もってわかっているようなデータをロードするためのヒープです。ゲーム起動時や、場面切り替え時などにロードします。

サウンドヒープは積み上げ方式のヒープで、ロードした順番と逆の順番でしか、データを消すことはできません。例えば、データ1、データ2、データ3の順にロードすると、データ3を消さない限り、あとの2つを消すことはできません。

7.1.2 プレイヤーヒープ

プレイヤーヒープは、実際に必要になった時点で始めてデータをロードするためのヒープです。プレイヤー毎に作成する必要があり、シーケンスの再生が終わると自動的に削除されます。

ゲーム実行中にロードすることになるので、大きなデータや頻繁に必要なデータには向きません。また、シーケンスアーカイブ再生用のデータには使えません。

7.2 グループの作成

サウンドヒープへデータをロードするために、グループを作成しておくとう便利です。グループを作成しておけば、複数のデータのロードが1度に行えます。

グループの作成は、サウンドアーカイブ定義ファイルのグループ情報セクションで行います。

コード 7-1 グループ情報セクション

```

;;;;;;;;;;;;;
;; Group

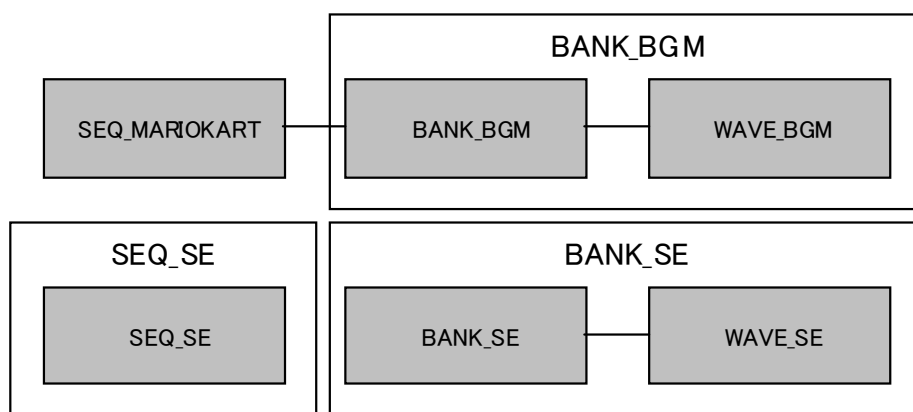
@GROUP

GROUP_STATIC :
    SEQ_SE
    BANK_SE
    BANK_BGM

```

上記の例では、GROUP_STATIC というグループを定義しています。グループ内容は、SEQ_SE と BANK_SE と BANK_BGM になります。具体的には、下図のようなデータがグループに含まれることになります。

図 7-1 グループ設定とサウンドデータの関係



BANK_SE や BANK_BGM の記述では、バンクデータだけではなく、対応する波形アーカイブも同時にロードされます。効果音用のシーケンスは、シーケンスアーカイブのため、バンクと直接関連づけられているわけではありません。そのため、SEQ_SE の指定をしても、効果音用のバンクや波形データは、同時にロードされません。

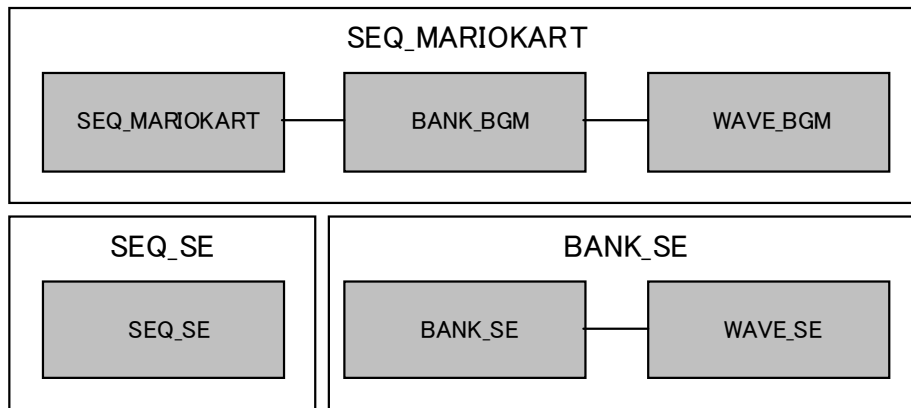
一方、シーケンスデータは、バンクと関連づけられているため、SEQ_MARIOKART を指定すると、BANK_BGM 及び WAVE_BGM も同時にロードされます。

```

GROUP_ALL :
    SEQ_SE
    BANK_SE
    SEQ_MARIOKART

```


図 7-2 グループ設定とサウンドデータの関係



7.3 グループのロード

グループを作成したら、ゲーム中の任意のタイミングでそのグループをロードできます。グループのロードを行うのは、サウンドプログラマーの仕事です。サウンドデザイナーは、どのタイミングでどのグループをロードする必要があるのかを、サウンドプログラマーに伝える必要があります。

7.4 プレイヤーヒープの作成

プレイヤーヒープへロードするためには、プレイヤーヒープを作るところから始めます。

プレイヤーヒープの作成は、サウンドアーカイブ定義ファイルのプレイヤー情報セクションで行います。

コード 7-2 プレイヤー情報セクション

```

;;;;;;;;;;;;;
;; Player

@PLAYER
0 : 1, 8000

```

上の例では、プレイヤー0 に対し、8000 バイトのプレイヤーヒープを作成するように指定しています。8000 の前に 1 と書いてあるのは、シーケンス最大同時再生数の設定ですが、ここでの説明は省略します。詳しくは、「サウンドアーカイブマニュアル」を参照してください。

プレイヤーヒープのサイズを決めるためには、まずどのサウンドデータがここへロードされるのかを把握する必要があります。サイズの決定については、後で説明します。

7.5 プレイヤーヒープへのロード

プレイヤーヒープへのデータのロードは、自動的に行われます。通常、シーケンスを再生しようとしたときにロードされます。

シーケンスの再生には、シーケンスデータ、バンクデータ、波形アーカイブなどが必要ですが、これらの内、サウンドヒープにロードされていないデータが、プレイヤーヒープにロードされます。例えば、バンクデータと波形アーカイブが、すでにサウンドヒープにロードされていると、プレイヤーヒープにはシーケンスデータがロードされます。

7.6 プレイヤーヒープサイズの決定

前にも述べたように、プレイヤーヒープのサイズを決めるためには、どのサウンドデータがここへロードされるのかを把握する必要があります。ここでは、シーケンスデータのみプレイヤーヒープへロードする場合を考えます。

まず、プレイヤーヒープはプレイヤー毎に作る必要があることを思い出してください。つまり、シーケンスデータの中で、このプレイヤーで再生するものだけが、このプレイヤーヒープにロードされることになります。なので、このプレイヤーで再生されるシーケンスデータの中で、もっともサイズの大きいものに合わせておけばよいということになります。

シーケンスデータのサイズを調べる最も良い方法は、NITRO-Player を使うことです。NITRO-Player では、シーケンスデータのサイズがリスト表示されますので、簡単にシーケンスデータのサイズを知ることができます。NITRO-Player については、`$NitroSystem/docs/NitroPlayer` ディレクトリのドキュメントをご覧ください。

その他の方法として、サウンドマップファイルを見るという方法もありますが、ここでの説明は省略します。詳しくは、「サウンドアーカイブマニュアル」をご覧ください。

7.6.1 サイズ決定の注意

プレイヤーヒープと全く同じサイズのサウンドデータをロードしようとしても、メモリ不足で失敗する可能性があります。100バイト程度余裕をもってサイズを指定しておく安全です。

8 ストリーム再生

NITRO-Composer ではシーケンス再生の他に、ストリーム再生を行うこともできます。ストリーム再生とは、ROM から波形データをロードしながら再生する方式です。シーケンス再生とストリーム再生の違いについては、「クイックスタートガイド」で説明しています。

この章では、ストリームデータの登録と作成方法について説明します。

8.1 ストリームデータの登録

まずは、ストリームデータの登録の仕方について説明します。

サウンドアーカイブ定義ファイル中で、@STRM ではじまる箇所が、ストリームデータを登録するストリームデータセクションです。

コード 8-1 ストリームデータセクション

```
;;;;;;;;;;;;;;  
;; Stream  
  
@STRM  
  @PATH "strm"  
  STRM_MARIOKART : PCM8, "kart_title.32.aiff", 127, 64, 0  
  STRM_FANFARE   : PCM8, "fanfare.32.aiff",   127, 64, 0
```

STRM_MARIOKART などではじまる1行が、1つのシーケンスデータを登録しています。

STRM_MARIOKART というラベルは、このストリームデータを指定するために使うラベルです。プログラムからこのストリームを再生しようとするときに、このラベルを使って指定することができます。

8.1.1 プレイヤー番号の指定

ストリームデータはストリームプレイヤーを使って再生します。どのストリームプレイヤーで再生するのかをプレイヤー番号で指定します。上の例では、行の一番最後に書いてある"0"がプレイヤー番号の指定で、ストリームプレイヤー0 を使って再生するように指定しています。

使用するストリームプレイヤーは、後で説明するストリームプレイヤー情報セクションで定義しておく必要があります。ここには、定義したストリームプレイヤーのプレイヤー番号を指定します。

1つのストリームプレイヤーで同時に再生できるのは、1つのストリームだけです。そのため、同時に再生したいストリームは、別々のストリームプレイヤーで再生する必要があります。ただし、複数のストリームデータを同時に再生しようとすると、処理時間が大きく増えることになるので注意してください。

8.1.2 その他のパラメータ

その他のパラメータは、ボリュームやプライオリティなどの設定値ですが、ここでの説明は省略します。詳しくは、「サウンドアーカイブマニュアル」をご覧ください。

8.2 ストリームプレイヤーの定義

ストリームを再生するにはストリームプレイヤーが必要です。サウンドアーカイブ定義ファイル中の、@STRM_PLAYER ではじまるストリームプレイヤー情報セクションで、使用するストリームプレイヤーを定義します。

コード 8-2 ストリームプレイヤー情報セクション

```
;;;;;;;;;;;;;;  
;; Stream Player  
  
@STRM_PLAYER  
0 : STEREO, 6, 7
```

この例では、プレイヤー番号 0 のステレオストリームプレイヤーを定義しています。モノラルプレイヤーを定義する場合は、次のようにします。

```
;;;;;;;;;;;;;;  
;; Stream Player  
  
@STRM_PLAYER  
0 : MONO, 6
```

ステレオプレイヤーでは、ステレオデータもモノラルデータも再生することができますが、モノラルプレイヤーではモノラルデータしか再生できません。その代わりに、ストリームプレイヤーが使用するメモリが、ステレオプレイヤーの半分になります。

"STEREO"や"MONO"のうしろに続く数字は、ストリーム再生に使用するチャンネルのチャンネル番号です。STEREO では、2チャンネル使用するので、2つのチャンネル番号を書きます。

チャンネル番号には、0～15 の値を指定できますが、番号によって幾つかの制限があります。詳しくは、「サウンドシステムマニュアル」に記載してありますが、大抵の状況では、6番、7番あたりを使っておけば良いでしょう。

8.3 ストリームデータの作成

ストリームデータは、AIFF フォーマットまたは WAV フォーマットで作成します。バンクに登録する波形データと基本的には同じです。ただし、ストリーム再生では、モノラルの他にステレオデータも再生することができます。

9 SoundPlayer

SoundPlayer は、サウンドアーカイバで生成したサウンドアーカイブを使って、ニンテンドーDS 上で音の確認ができるツールです。ここでは、SoundPlayer の使い方について説明します。

なお、NITRO-Player を使う場合でも、DS 本体側の表示は SoundPlayer とほぼ同一ですので、SoundPlayer の使い方について、一通り目を通しておくことをお勧めします。

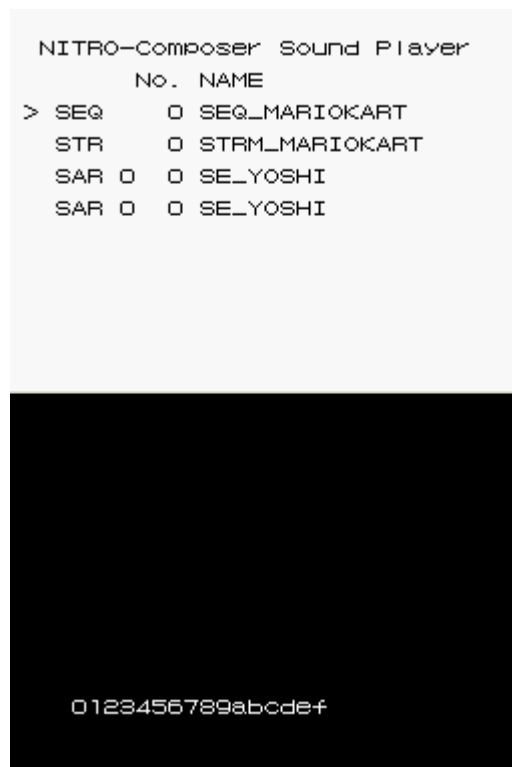
9.1 SoundPlayer の使い方

SoundPlayer を使うためには、次のような手順で SoundPlayer 実行ファイルを作成して、それを実行します。

- \$data ディレクトリ内にサウンドデータを作成します。
- \$data/MakeSound.bat を実行します。
- \$data/SoundPlayer.srl が生成されます。
- IS-NITRO-DEBUGGER に SoundPlayer.srl を読み込ませて実行します。

実行が正常に行われると、ニンテンドーDS の画面に、次のようなものが表示されます。

図 9-1 SoundPlayer 画面



9.1.1 MakeSound.bat

MakeSound.bat を実行すると、次の2つのことが行われます。

1. サウンドアーカイバ sndarc.exe を起動し、*.sdat ファイルを生成

2. MakeSoundPlayer.bat を起動し、SoundPlayer.srl を生成

以上の動作が正常に行われると、ウィンドウがしばらく表示されて、そのあと自動的に消えます。エラーが発生したときには、ウィンドウが残ったままになり、そこにエラーメッセージが表示されます。エラーの対処は、「10エラーの対処」を参照してください。

\$data/MakeSound.bat は、上記2つの処理を起動するだけの簡単なバッチファイルです。必要に応じて、修正して使っても構いません。

MakeSound.bat を実行しても、データファイルが更新されていなければ、何も行われません。全て再コンバートしたい場合は、MakeSound.bat の代わりに ReMakeSound.bat を実行してください。

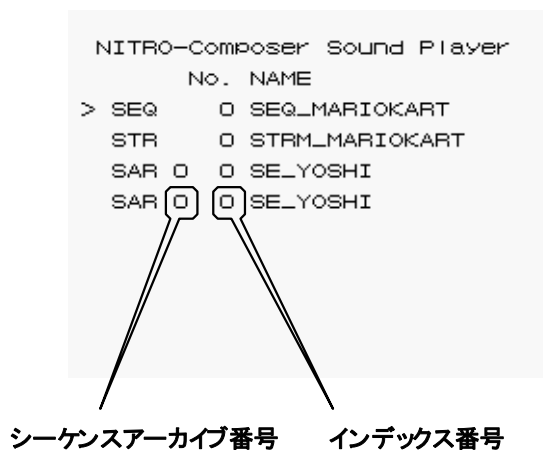
9.2 サウンドの再生

SEQ、STR、SAR と書かれている行は、上下キーを使って、カーソルで選択できます。SEQ、STR、SAR はそれぞれ、シーケンス、ストリーム、シーケンスアーカイブを表し、該当する行にカーソルを合わせると、Aボタンで再生できます。

全部で4つの行で再生することができますが、1つの行では1つのサウンドしか再生できません。なので、2つのサウンドを同時に再生したい場合は、別の行で再生する必要があります。ただし、別の行で再生しても、再生するプレイヤーで同時に再生できるシーケンス数の制限などによって、同時に再生できない場合がありますので、注意してください。

サウンド番号の選択は、十字キー左右で行います。ただし、シーケンスアーカイブの場合、十字キーの左右は、1つのシーケンスアーカイブ内でのインデックス番号の選択になります。シーケンスアーカイブ自体を選択したい場合は、Lボタンを押しながら、十字キーの上下で行います。シーケンスアーカイブ番号は、"SAR"のすぐ右に表示されています。

図 9-2 SoundPlayer 上画面(シーケンスアーカイブ番号)



サウンドの停止は、カーソルを合わせてBボタンです。該当行で再生中のサウンドを止めることができます。

上の例では4つの行で SEQ、STR、SAR、SAR の順に並んでいますが、ストリームデータが無い場合、STR の行が SEQ になったりします。

9.2.1 複数サウンド同時再生

複数の行で、複数のサウンドを同時に再生することができますが、Y ボタンを使うことで、再生開始のタイミングを揃えることができます。

まず、再生するサウンドを選択し、Y ボタンを押します。すると、アスタリスクマーク(*)がつきます。これを、それぞれの行で、同時に再生したいサウンドについて行います。その後、A ボタンで再生すると、アスタリスクマークがついたサウンドが同時に再生されます。

また、通常ストリームを再生するときには、データロードのための再生遅延が発生しますが、Y ボタンでマークをつけたタイミングで、再生準備を行いますので、A ボタンを押したタイミングから遅延無く再生が開始します。

9.3 サウンドタイプの選択

L ボタンを押しながら十字キーの上下を押すと、サウンドタイプまたは、シーケンスアーカイブ番号を選択できます。L ボタンを押しながら上ボタンを押すたびに、次のように変わります。

```
SEQ → SAR 0 → SAR 1 → SAR 2 → ... → STR → SEQ
```

9.4 画面表示

9.4.1 ラベル名表示

NAME の所には、選択中のサウンドのラベル名が表示されます。ラベルが指定されていない場合は、「no name」と表示されます。

9.4.2 パラメータ表示

X ボタンを押すと、ラベル名の代わりにパラメータが表示されます。再度Xボタンを押すと、ラベル名表示に戻ります。

図 9-3 SoundPlayer 上画面(パラメータ表示)

NITRO-Composer Sound Player							
	No.	BNK	VOL	CPr	PPr	PLY	
> SEQ	0	1	127	64	64	0	
STR	0		127		64	0	
SAR 0	0	0	127	96	64	11	
SAR 0	0	0	127	96	64	11	

各行毎に、数字が並んでいます。

No. BNK VOL CPr PPr PLY

1つめの No.は、サウンド番号です。十字キーの左右で値を変更することができます。

2つ目以降の値は、サウンドアーカイブなどで各サウンド毎に設定したパラメータが表示されています。これらの値は、参照するだけで変更することはできません。左から順に、バンク番号、マスターボリューム、チャンネルプライオリティ、プレイヤープライオリティ、プレイヤー番号です。これらの値についての詳細は、「サウンドアーカイブマニュアル」または、「シーケンスデータマニュアル」をご覧ください。

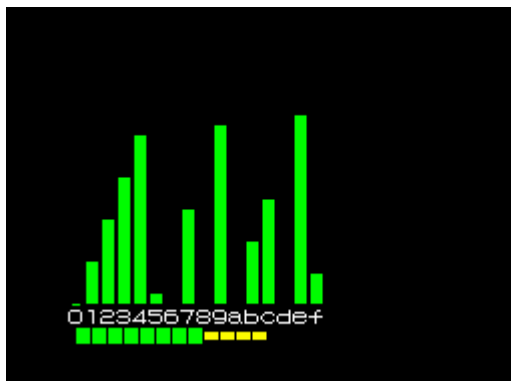
9.4.3 シーケンスアーカイブ番号

SAR のすぐ右に表示されている数字が、シーケンスアーカイブ番号です。Lボタンを押しながら、十字キーの上下で変更することができます。

9.5 チャンネルメーター

下画面に表示されているのは、チャンネルメーターです。チャンネルの発音状況を表示しています。

図 9-4 SoundPlayer チャンネルメーター

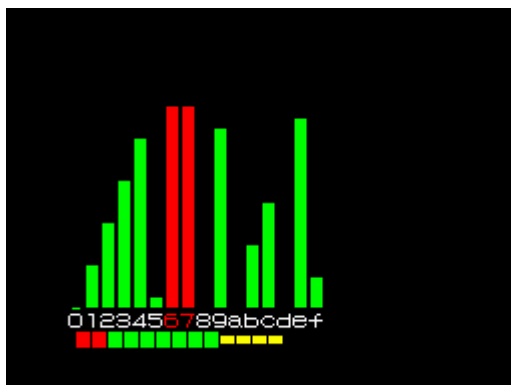


上方の縦長のバーは、各チャンネル毎の出力ボリューム値を表しています。その下の数値(16進数)は、ニンテンドーDS サウンドハードウェアにおけるチャンネル番号を表しています。

一番下の段は、発音チャンネル数を表していて、そのうち黄色の部分はリリース中の発音数を表しています。

また、ストリーム再生など、シーケンス再生以外によってチャンネルが使用されている場合は、次のように赤色で表示されます。

図 9-5 SoundPlayer チャンネルメーター(赤表示)



9.6 リアルタイム MIDI 再生

9.6.1 リアルタイム MIDI 再生とは

MIDI 端子付き IS-NITRO-UIC または IS-AGB-MIDI があれば、MIDI 信号を使って、SoundPlayer から音を鳴らすことができますようになります。この機能を使えば、いちいちシーケンスデータをコンバートしなくても、MIDI シーケン

スデータを再生できるようになります。

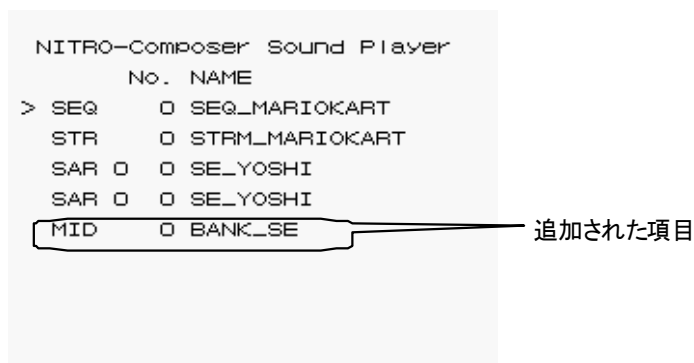
9.6.2 準備

リアルタイム MIDI 再生をするためには、MIDI 端子付き IS-NITRO-UIC または IS-AGB-MDI が必要です。接続方法などについて詳しくは、IS-NITRO-UIC または IS-AGB-MIDI のマニュアルをご覧ください。ここでは、接続手順の一例を紹介します。

- IS-NITRO-UIC(IS-AGB-MIDI)カートリッジを IS-NITRO-EMULATOR のカートリッジスロットに挿す
- IS-NITRO-DEBUGGER の「プロジェクトの設定」でカートリッジの電源を ON にする
- SoundPlayer を起動する

接続が正しいと、下図のように SoundPlayer 画面に、「MID」の項目が追加されます。

図 9-6 SoundPlayer 画面(リアルタイム MIDI)



9.6.3 バンクの選択

リアルタイム MIDI 再生で使用するバンクを選択します。

カーソルを、「MID」のところにあわせて、左右でバンクを選択できます。選択してAボタンを押すと、そのバンクがロードされ、以下 MIDI 再生用のバンクとして使用できます。

9.6.4 MIDI 信号送信

IS-NITRO-UIC(IS-AGB-MIDI)の MIDI 端子に MIDI ケーブルを繋ぎ MIDI 信号を送信すると、選択したバンクを使って音を鳴らしたりすることができます。デフォルトでは、プログラムナンバー0になっているので、必要に応じてプログラムチェンジメッセージを送信してください。

9.7 手動ロードモード

9.7.1 手動ロードモードとは

SoundPlayer でシーケンスを再生しようとする時、必要なデータを自動的に読み込んで再生します。これは、音の確認のためには非常に便利です。ただし実際のゲーム中では、シーケンスの再生時に自動的にロードすることは稀で、通常は場面切り替え時に、グループ単位でロードしたりします。そのため、グループの設定などが正しく行われていないと、シーケンスを再生することはできません。

手動ロードモードでは、自動的にデータをロードせずに、手作業でデータのロードを行います。そのため、実際のゲーム中でも正しくシーケンスを再生できるかどうかを確認することができます。また、グループをロードすると、どのくらいメモリを消費するのかということも、確認できます。

なお、音の確認用にデータが自動的にロードされるモードのことを、手動ロードモード区別して、自動ロードモードと呼びます。

9.7.2 手動ロードモードでの動作

手動ロードモードでは、データのロードは自動的にには行われません。必要なデータがロードされていない場合は、シーケンスの再生に失敗します。ただし、プレイヤーヒープの設定が行われていて、サイズが十分にあれば、そこにロードして再生します。

9.7.3 モードの切り替え

自動ロードモードと手動ロードモードの切り替えは、SELECT ボタンで行います。SELECT ボタンを押すたびに、2つモードが切り替わります。

図 9-7 SoundPlayer 画面(手動ロードモード)

```
NITRO-Composer Sound Player
      No. NAME
> SEQ   0 SEQ_MARIOKART
  STR   0 STRM_MARIOKART
  SAR 0   ---
  SAR 0   ---
  GRP   0 GROUP_STATIC
Memory 0 %
13392 / 3786927
```

モードを切り替えると、ロードしたデータは全てクリアされます。

9.7.4 データのロード

手動ロードモードでは、チャンネルメーターの代わりに、ロード実行項目と、メモリ使用状況が表示されます。

ロードは、グループ単位、シーケンス単位、シーケンスアーカイブ単位、バンク単位、波形アーカイブ単位のどれでも行うことができます。ただし、シーケンス単位とバンク単位の場合、使用するバンクや波形アーカイブも同時にロードされます。

デフォルトでは、"GRP"と表示され、グループ単位のロードになります。ロード単位の変更は、Lボタンを押しながら、十字キーの上下で行います。

十字キー左右で番号を選択し、Aボタンを押すとデータがロードされます。

L ボタンと B ボタンを押しながら、A ボタンを押すと、ロードしたデータを全てクリアすることができます。

9.7.5 メモリ表示

一番下に、現在使用中のメモリを表示しています。データをロードする毎にメモリが消費されます。

ただし、初期状態でも、サウンドアーカイブのヘッダ情報および、プレイヤーヒープ作成のために、ある程度のメモリが消費されています。

9.8 キー操作

下の表は、キー操作をまとめたものです。

表 9-1 SoundPlayer のキー操作

キー	説明
十字キー上下	カーソル移動
十字キー左右	値の変更
Aボタン	再生開始または、データロード
Bボタン	再生停止
X ボタン	ラベル名／パラメータ表示切り替え
Y ボタン	同時再生マーク付加
Lボタン+十字左右	押している間、値の変更を10ずつ行う
L ボタン+十字上下	シーケンスアーカイブ番号の選択または、 サウンドタイプの選択または、 ロード単位の選択
SELECT	モード切替
L ボタン+B ボタン+A ボタン	メモリクリア

9.9 エラーメッセージ

画面上の一番下の段に、エラーメッセージが表示されることがあります。シーケンスを再生しようとしても、再生できなかった場合は、ここに表示されるエラーメッセージで原因を知ることができます。

エラーメッセージには以下のようなものがあります。

表 9-2 SoundPlayer エラーメッセージ

エラーメッセージ	説明
Low Priority	より優先度の高いシーケンスが再生中のため、再生できません。
Invalid Seq No	未定義のシーケンス番号です。
Invalid SeqArc No	未定義のシーケンスアーカイブ番号です。
Invalid Bank No	未定義のバンク番号です。
Invalid WaveArc No	未定義の波形アーカイブ番号です。
Invalid Group No	未定義のグループ番号です。

Invalid SeqArc Index	未定義のシーケンスアーカイブインデックスです。
Invalid Stream No	未定義のストリーム番号です。
Invalid Stream Player No	未定義のストリームプレイヤー番号です。
Memory Over	メモリが足りないため、これ以上ロードできません。
Too Large Data	必要なデータのサイズが大きすぎるため、再生できません。
Not Found Wave Data	必要な波形データが見つかりません。(手動ロードモード)
Not Found Bank Data	必要なバンクデータが見つかりません。(手動ロードモード)
Not Found Seq Data	必要なシーケンスデータが見つかりません。(手動ロードモード)
Not Found SeqArc Data	必要なシーケンスアーカイブデータが見つかりません。(手動ロードモード)
Not Enough Player Heap for Wave	波形データをロードするには、プレイヤーヒープが小さすぎます。
Not Enough Player Heap for Bank	バンクデータをロードするには、プレイヤーヒープが小さすぎます。
Not Enough Player Heap for Seq	シーケンスデータをロードするには、プレイヤーヒープが小さすぎます。

10 エラーの対処

サウンドデータをコンバートしたときに発生したエラーへの一般的な対処方法を説明します。

なお、MakeSound.bat をダブルクリックして実行したときの状況で説明していますが、別の方法で MakeSound.bat を実行した場合でも参考になります。

10.1 エラーの発生

MakeSound.bat をダブルクリックして、ウィンドウがすぐ消えずに残った場合には、エラーが発生しています。ウィンドウの最後の方には、下記のようなエラーメッセージが表示されます。

```
smfconv: BgmSeq/kart64_title.mid: Cannot open file
sndarc: smfconv: Failed
続行するには何かキーを押してください . . .
```

10.2 原因の特定

幾つかのメッセージが表示されるため、何が原因かを特定するのが難しいですが、基本的には、一番下の行から順に辿っていけば、原因を特定できます。

一番下の行は、単にウィンドウを閉じるためのメッセージを表示しているだけです。次に下から 2 行目を見てみます。

```
sndarc: smfconv: Failed
```

始めが"sndarc:"で始まっていますが、これは以下の文字列が sndarc というツールが出力したメッセージであることを意味します。つまり、"smfconv: Failed"というメッセージを、sndarc というツールが出力したということです。なにやら、「smfconv が失敗した」というような意味のようですが、はっきりした原因はまだわかりません。

次にもう一つ上の行を見てみます。

```
smfconv: BgmSeq/kart64_title.mid: Cannot open file
```

これは、smfconv というツールが、"BgmSeq/kart64_title.mid: Cannot open file"というメッセージを出力したという意味です。「BgmSeq/kart64_title.mid というファイルを開けない」というエラーメッセージのようです。

これでエラーの原因が特定できました。BgmSeq/kart64_title.mid というファイルが開けなかったために、smfconv が失敗したということのようです。

10.3 エラーの対処

「BgmSeq/kart64_title.mid というファイルが開けなかった」ということは、おそらくファイルが存在しなかったためと思われますので、実際にファイルがあるかどうかを確認する必要があります。

または、ファイル名の指定を間違ったことが原因かも知れません。その場合は、サウンドアーカイブ定義ファイルの記述を修正する必要があります。

10.4 対処の確認

エラーの対処を行ったら、もう一度 MakeSound.bat をダブルクリックして、コンバートしてみます。

ウィンドウがすぐに閉じれば、エラーは解決です。再び、ウィンドウが残ってしまったら、もう一度はじめてやり直してください。

11 次のステップ

ここまでで、サンプルデータを見ながら、サウンドデータの構成や、おおまかな仕様について学びました。より詳細な仕様について知りたいときは、下記のマニュアルを参照してください。

- サウンドアーカイバについて詳しく知りたい時は、「サウンドアーカイブマニュアル (NITRO_Composer_SoundArchiveManual.pdf)」をご覧ください。
- シーケンスデータについて詳しく知りたいときは、「シーケンスデータマニュアル (NITRO_Composer_SequenceDataManual.pdf)」をご覧ください。
- バンクデータについて詳しく知りたいときは、「バンクデータマニュアル (NITRO_Composer_BankDataManual.pdf)」をご覧ください。

また、NITRO-Player を使ってみたいという方は、「NITRO-Player ユーザーマニュアル」が\$NitroSystem/docs/NitroPlayer/NITRO_Player_UserManual.pdf にありますので、ご覧下さい。

© 2004-2007 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。