

NITRO-Composer サウンドツールマニュアル

2008-04-08

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
1.1	本ドキュメントの構成	7
1.1.1	2章 プリプロセッサ命令	7
1.1.2	3章 サウンドアーカイバ	7
1.1.3	4章 SMFコンバータ smfconv	7
1.1.4	5章 シーケンスコンバータ seqconv	7
1.1.5	6章 バンクコンバータ bankconv	7
1.1.6	7章 波形ファイルコンバータ waveconv	7
1.1.7	8章 波形ファイルアーカイバ wavearc	7
1.1.8	9章 ストリームコンバータ strmconv	7
2	プリプロセッサ命令	8
2.1	概要	8
2.2	プリプロセッサ命令	8
2.3	#define	8
2.3.1	例	9
2.4	#undef	9
2.4.1	例	10
2.5	#include	10
2.5.1	例	11
2.6	#if	11
2.6.1	例	12
2.7	#ifdef / #ifndef	12
2.7.1	例	13
2.8	#else	13
2.8.1	例	13
2.9	#endif	14
2.10	#elif	14
2.10.1	例	14
3	サウンドアーカイバ sndarc	16
3.1	概要	16
3.2	実行ファイルの場所	16
3.3	使用方法	16
3.3.1	書式	16
3.3.2	実行結果	16
3.3.2.1	サウンドラベルファイル	16
3.3.2.2	サウンドアーカイブラベルファイル	17
3.3.2.3	サウンドマップファイル	17
3.3.3	オプション	17
3.3.4	コンバートファイルタイプ	17
3.3.5	サウンドデータのDMA転送	18

3.3.5.1	サウンドアーカイブのオフセット.....	18
3.3.5.2	アライメントされるデータ.....	18
4	SMFコンバータ smfconv.....	19
4.1	概要.....	19
4.2	実行ファイルの場所.....	19
4.3	使用方法.....	19
4.3.1	書式.....	19
4.3.2	実行結果.....	19
4.3.3	オプション.....	19
4.4	詳細.....	20
4.4.1	トラック.....	20
4.4.2	ループ指定.....	20
4.4.3	SMF先頭部の空白.....	20
4.4.4	MIDIイベント.....	20
5	シーケンスコンバータ seqconv.....	21
5.1	概要.....	21
5.2	実行ファイルの場所.....	21
5.3	使用方法.....	21
5.3.1	書式.....	21
5.3.2	実行結果.....	21
5.3.3	オプション.....	21
6	バンクコンバータ bankconv.....	23
6.1	概要.....	23
6.2	実行ファイルの場所.....	23
6.3	使用方法.....	23
6.3.1	書式.....	23
6.3.2	2つのモード.....	23
6.3.3	波形リストファイル出力モード.....	23
6.3.3.1	波形グループ番号.....	23
6.3.3.2	実行結果.....	24
6.3.4	オプション.....	24
6.3.5	バンクコンバートモード.....	24
6.3.5.1	実行結果.....	24
6.3.5.2	オプション.....	24
7	波形ファイルコンバータ waveconv.....	26
7.1	概要.....	26
7.2	実行ファイルの場所.....	26
7.3	使用方法.....	26
7.3.1	書式.....	26
7.3.2	実行結果.....	26
7.3.3	オプション.....	26

7.4	波形フォーマット	27
7.4.1	ファイル形式	27
7.4.2	チャンネル数	27
7.4.3	量子化ビット数	27
7.4.4	サンプリングレート	27
7.4.5	オリジナルキー	27
7.4.6	ループ	27
7.5	詳細	27
7.5.1	ループ補正	27
7.5.2	ループスタート位置の限界	27
8	波形ファイルアーカイバ wavearc	29
8.1	概要	29
8.2	実行ファイルの場所	29
8.3	使用方法	29
8.3.1	書式	29
8.3.2	オプション	29
9	ストリームコンバータ strmconv	30
9.1	概要	30
9.2	実行ファイルの場所	30
9.3	使用方法	30
9.3.1	書式	30
9.3.2	実行結果	30
9.3.3	オプション	30
9.4	波形フォーマット	31
9.4.1	ファイル形式	31
9.4.2	チャンネル数	31
9.4.3	量子化ビット数	31
9.4.4	サンプリングレート	31
9.4.5	ループ	31
9.5	リサンプリング	31

表

表 2-1	プリプロセッサ命令一覧	8
表 3-1	sndarcオプション一覧	17
表 3-2	コンバートファイルタイプ	18
表 4-1	smfconvオプション一覧	19
表 5-1	seqconvオプション一覧	21
表 6-1	bankconv波形リストファイル出力モードオプション一覧	24
表 6-2	bankconvバンクコンバートモードオプション一覧	24
表 7-1	waveconvオプション一覧	26
表 7-2	ループスタート位置の限界	28
表 8-1	wavearcオプション一覧	29

表 9-1 strmconvオプション一覧	30
-----------------------------	----



図 2-1 共通部分抜き出しの例	11
図 2-2 複数人編集の例	11

改訂履歴

改訂日	改訂内容
2008-04-08	1. 改訂履歴の書式を変更 2. ページのヘッダを修正
2007-03-14	1. サウンドデータの DMA 転送に関する説明追加 2. sndarc のアライメント指定オプション追加 3. sndarc のコンバートファイルタイプ指定オプション追加 4. bankconv の引数リストファイル指定オプション追加
2005-09-01	1. sndarc の出力ファイル名指定オプション追加 2. sndarc の前処理ファイル指定オプション追加
2005-01-31	1. #include<ファイル>形式の書式に関する説明追加 2. seqconv 及び bankconv に、-I オプション追加 3. "NITRO"を"ニンテンドーDS"に変更
2004-10-12	1. waveconv のループスタート位置制限に関する説明追加
2004-09-16	1. sndarc が出力する*.sadl ファイル及び*.sbd1 ファイルに関する説明修正 2. *.sadl ファイルの呼称を「サウンドラベルファイル」に統一 3. バンクリストファイル(*.sbd1)の呼称をサウンドアーカイブラベルファイルに変更
2004-09-02	1. sndarc が出力するバンクリストファイルに関する説明追加 2. #include の例を変更
2004-08-10	1. strmconv の追加
2004-07-20	1. sndarc に-b オプション追加 2. bankconv の仕様変更 3. waveconv にループ補正に関する説明追加 4. スタイルの整形
2004-06-01	1. #include の例を*.spd1 ファイルを使ったものに変更 2. 各コンバータの説明に「実行結果」の見出しを追加 3. sndarc、seqconv、bankconv で出力される各リストファイルに関する記述の追加や修正
2004-04-12	初版

1 はじめに

本ドキュメントではサウンドデザイナー向けに、サウンドツールの詳細な使い方を説明しています。すでに、NITRO-Composer を使ってサウンドデータの作成を行える方を対象としていますので、初めての方は、まず「サウンドデザイナーガイド」などから読み進めてください。

1.1 本ドキュメントの構成

本ドキュメントは、各章独立に構成されていますので、必要な章だけ読んで頂いても構いません。以下、各章の概要を紹介します。

1.1.1 2章 プリプロセッサ命令

プリプロセッサ命令について説明しています。プリプロセッサ命令とは、全てのテキスト形式のサウンドデータファイル中で使用できる命令で、テキストの記述をサポートするものです。

1.1.2 3章 サウンドアーカイバ

サウンドアーカイバ `sndarc` をコマンドラインから直接使う方法について説明しています。

1.1.3 4章 SMFコンバータ `smfconv`

SMF コンバータ `smfconv` をコマンドラインから直接使う方法について説明しています。

1.1.4 5章 シーケンスコンバータ `seqconv`

シーケンスコンバータ `seqconv` をコマンドラインから直接使う方法について説明しています。

1.1.5 6章 バンクコンバータ `bankconv`

バンクコンバータ `bankconv` をコマンドラインから直接使う方法について説明しています。

1.1.6 7章 波形ファイルコンバータ `waveconv`

波形ファイルコンバータ `waveconv` をコマンドラインから直接使う方法について説明しています。

1.1.7 8章 波形ファイルアーカイバ `wavearc`

波形ファイルアーカイバ `wavearc` をコマンドラインから直接使う方法について説明しています。

1.1.8 9章 ストリームコンバータ `strmconv`

ストリームコンバータ `strmconv` をコマンドラインから直接使う方法について説明しています。

2 プリプロセッサ命令

2.1 概要

プリプロセッサ命令とは、全てのテキスト形式のサウンドデータファイル中で使用できる命令で、テキストの記述をサポートするものです。

プリプロセッサ命令の1つに`#define`というものがあります。これは、次のように使います。

```
#define LENGTH 24
```

このように書いておくと、それ以降の行では、数字 `24` を書くところの代わりに、`LENGTH` と書くことができるようになります。例えば、そのような箇所が複数あった場合、`#define` の行の `24` と書いてある値を変更するだけで、すべての値を一度に変更できるようになります。これは、値を試行錯誤して決めているときなどに、非常に有効です。

2.2 プリプロセッサ命令

プリプロセッサ命令には、以下のようなものがあります。

表 2-1 プリプロセッサ命令一覧

プリプロセッサ命令	説明	参照
<code>#define</code>	置換マクロを定義します	8
<code>#undef</code>	置換マクロの定義を無効にします	9
<code>#include</code>	別のファイルを組み込みます	10
<code>#if</code>	真偽の条件評価を行います	11
<code>#ifdef</code>	定義済みマクロについて条件評価を行います	12
<code>#ifndef</code>	未定義マクロについて条件評価を行います	12
<code>#else</code>	<code>#if</code> , <code>#ifdef</code> , <code>#ifndef</code> , <code>#elif</code> の偽部の評価区分の始まりを示します	13
<code>#endif</code>	<code>#if</code> , <code>#ifdef</code> , <code>#ifndef</code> , <code>#elif</code> で始まる評価区分の終わりを示します	14
<code>#elif</code>	<code>#if</code> , <code>#ifdef</code> , <code>#ifndef</code> の偽部での条件評価を行います	14

少し、難解な説明ですが、下記の具体例をみると、すぐに理解できると思います。

2.3 `#define`

置換マクロを定義します。書式は次の通りです。

`#define` マクロ名 置換の並び

これ以降の行で、マクロ名と同じ文字列が存在すると、それは置換の並びで指定したものに置換されて処理されます。マクロ名は、単語の一部ではなく全体が一致したときのみ置換が行われます。また、アルファベットの大文字・小文字は

区別されます。

マクロ名は、ラベル名と同じように、アルファベットで始まり、アルファベット・数字・アンダースコア()が続く文字列を指定できます。通常は、小文字のアルファベットは使いません。

置換の並びは、自由な記述が可能です。当然、マクロを使用する箇所の書式にあわせる必要があります。例えば、数値を記入するところで使うマクロを定義するなら、置換の並びは数値を表していなければなりません。また、置換の並びは、省略することもできます。`#ifdef` で使うために定義だけしたい場合などには、省略します。

同じマクロ名を2回定義するとエラーになりますが、`#undef` を使って一度無効化すると再定義することができます。

2.3.1 例

```
#define LENGTH 24

cn4 127, LENGTH
dn4 127, LENGTH
en4 127, LENGTH
fin
```

は、以下の記述と同じです。

```
cn4 127, 24
dn4 127, 24
en4 127, 24
fin
```

置換の並びは、自由に記述できますので、次の記述も同じ意味です。

```
#define VEL_LEN 127, 24

cn4 VEL_LEN
dn4 VEL_LEN
en4 VEL_LEN
fin
```

次は、`#ifdef` と組み合わせた例です。

```
#define ENABLE_FLAG

cn4 127, 24
dn4 127, 24
#ifdef ENABLE_FLAG
en4 127, 24
#endif
fin
```

`#ifdef` と `#endif` で囲まれた行は、`ENABLE_FLAG` が定義されているときのみ有効になります。`#define ENABLE_FLAG` の行を削除すると、`en4 127, 24` の行は無効になります。

2.4 #undef

置換マクロの定義を無効にします。書式は次の通りです。

`#undef マクロ名`

`#define` で定義されたマクロを無効化して、未定義の状態にします。定義されていないマクロを指定しても、何もありません。

マクロを再定義したい場合に使ったり、`#ifdef/#ifndef`と組み合わせて使ったりします。

2.4.1 例

```
#define LENGTH 24

    cn4 127, LENGTH
    dn4 127, LENGTH
    en4 127, LENGTH

#undef LENGTH
#define LENGTH 12

    cn4 127, LENGTH
    dn4 127, LENGTH
    en4 127, LENGTH
fin
```

このように、`#undef`を使えば、同じマクロ名でも2回定義できるようになります。

```
#define ENABLE_FLAG
#undef ENABLE_FLAG

    cn4 127, 24
    dn4 127, 24
#ifdef ENABLE_FLAG
    en4 127, 24
#endif
fin
```

`#ifdef`と`#endif`で囲まれた行は、`ENABLE_FLAG`が定義されているときのみ有効になりますが、`#undef`で無効にしているため、`en4 127, 24`の行は無効になります。

2.5 #include

別のファイルを組み込みます。書式は次の通りです。

```
#include"ファイル名"

#include<ファイル名>
```

ファイル名で指定したファイルを、現在の位置に組み込みます。`#include`を使うことで、幾つかのファイルの共通部分を1つのファイルにまとめたり、1つのファイルを複数の人で編集するときに、別々のファイルにわけたりすることができるようになります。

ファイル名には、絶対パス、相対パスのいずれも使用できます。相対パスで指定した場合、起点となるディレクトリが、2種類の書式によって異なります。`#include"ファイル名"`形式の書式では、現在のファイルからの相対パスになります。`#include<ファイル名>`形式の書式では、あらかじめ登録されたインクルードパスからの相対パスとなります。

インクルードパスは、コンバータのコマンドラインオプション-Iを使って指定することができます。複数のインクルードパスを指定すると、始めに登録したところから順に探索されます。サウンドアーカイバ `sndarc` を使用している場合は、デフォルトで、サウンドアーカイブ定義ファイル(*.sarc)が存在するディレクトリが、インクルードパスとして登録されています。従って、`#include<ファイル名>`形式の書式は、サウンドアーカイブ定義ファイル(*.sarc)が存在するディレクトリからの相対パスであると考えることができます。

2.5.1 例

バンク定義ファイルにて、インストゥルメントにラベル名をつけていると、拡張子*.spd1 のプログラムナンバリストファイルに次のような内容のものが出力されます。

```
#define PRG_PIANO 0
#define PRG_ORGAN 1
#define PRG_GUITAR 2
```

このファイルをテキストシーケンスの先頭でインクルードすることにより、プログラムチェンジでラベルを使って記述できるようになります。

```
#include "../bnk/se.spd1"

prg PRG_ORGAN
cn4 127, 24
fin
```

また、共通部分を1つのファイルにまとめたり、1つのファイルを複数人で編集したりするときは、次のようにします。

図 2-1 共通部分抜き出しの例

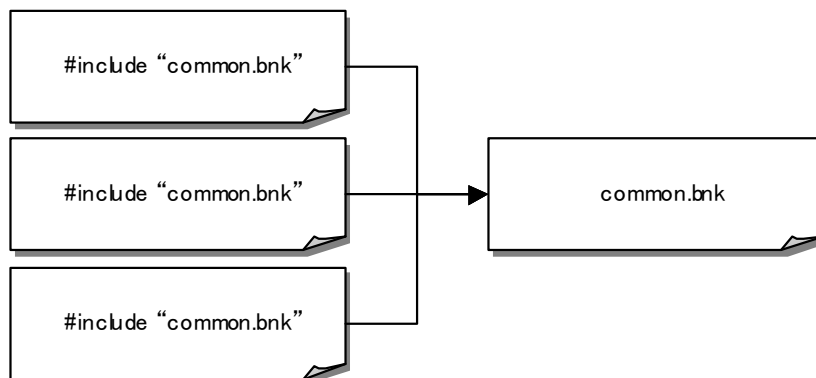
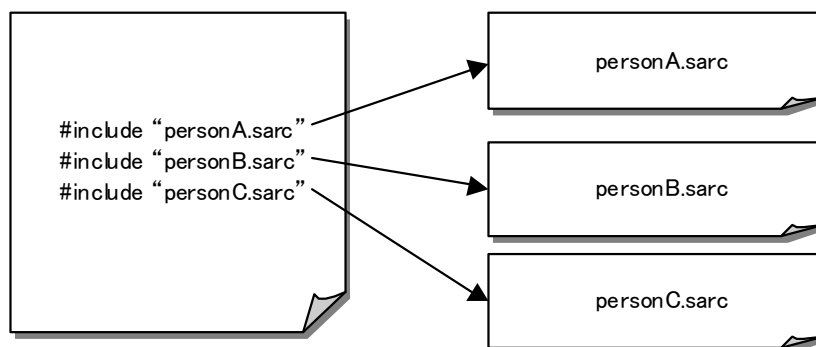


図 2-2 複数人編集の例



2.6 #if

真偽の条件評価を行います。基本式は次の通りです。

```
#if 評価式  
  
(真の時有効)  
  
#endif
```

評価式を評価して、結果が正しければ、以降の行は有効になりますが、結果が間違っていれば以降の行は無効になります。有効または無効の区間は、**#endif** までになります。

評価式には、数値を指定することもできます。数値を指定した場合、0なら無効、0以外なら有効となります。

2.6.1 例

次のようにして、ある区間を無効にして、コメントの代わりに使えます。

```
#if 0  
    cn4 127, 24  
    dn4 127, 24  
    en4 127, 24  
    fin  
#endif
```

このようにしておけば、あとで有効にするのも簡単です。

```
#if 1  
    cn4 127, 24  
    dn4 127, 24  
    en4 127, 24  
    fin  
#endif
```

このようにしておけば、あとで有効にするのも簡単です。

```
#define VERSION 2  
  
    cn4 127, 24  
    dn4 127, 24  
#if VERSION >= 2  
    en4 127, 24  
#endif  
    fin
```

このように、式を評価することで、有効／無効を切り替えることができます。この例では、結果が正しいので、**en4 127, 24** の行は有効になります。

2.7 #ifdef / #ifndef

マクロの定義または未定義の状態を評価します。書式は次の通りです。

```
#ifdef マクロ名  
  
#ifndef マクロ名
```

評価式の代わりに、マクロ名を指定して定義されているかどうかを評価します。**#ifdef** は、マクロが定義されている時に真となり、**#ifndef** はマクロが未定義の時に真となります。それ以外の動作は、**#if** と同じです。

2.7.1 例

```
#define ENABLE_FLAG

        cn4 127, 24
        dn4 127, 24
#ifdef ENABLE_FLAG
        en4 127, 24
#endif
#ifndef ENABLE_FLAG
        fn4 127, 24
#endif
fin
```

上の例では、ENABLE_FLAG が定義されているので、en4 127, 24 の行が有効になり、fn4 127, 24 の行が無効になります。一方、

```
#define ENABLE_FLAG
#undef ENABLE_FLAG

        cn4 127, 24
        dn4 127, 24
#ifdef ENABLE_FLAG
        en4 127, 24
#endif
#ifndef ENABLE_FLAG
        fn4 127, 24
#endif
fin
```

このように、#undefを行うと、ENABLE_FLAG が未定義状態になりますので、逆に fn4 127, 24 の行が有効になります。

2.8 #else

#if,#ifdef,#ifndef,#elif の偽部の評価区分の始まりを示します。基本式は次の通りです。

```
#if ...
(真の時有効)

#else
(偽の時有効)

#endif
```

#if,#ifdef の評価結果が偽の時、#else～#endif までの区間が有効になります。逆に結果が真の場合は、#else～#endif の区間は無効になります。

2.8.1 例

次のようにすると、2つのバージョンを簡単に切り替えることができます。

```
#if 1
    cn4 127, 24
    dn4 127, 24
    en4 127, 24
    fin
#else
    cn4 127, 24
    en4 127, 24
    gn4 127, 24
    fin
#endif
```

この例では、`#if` ~`#else` までの区間が有効になりますが、`#if 1` を`#if 0`に変えるだけで、`#else`~`#endif` の方を有効にすることができます。

2.9 #endif

`#if`,`#ifdef`,`#ifndef`,`#elif` で始まる評価区分の終わりを示します。基本式は次の通りです。

```
#if 評価式
(真の時有効)
#endif
```

詳しくは、`#if`,`#ifdef`,`#ifndef`,`#elif` の説明をご覧ください。

2.10 #elif

`#if`,`#ifdef`,`#ifndef` の偽部での条件評価を行います。基本式は次の通りです。

```
#if 評価式1
#elif 評価式2
#endif
```

言葉で説明するより、例を見た方がわかりやすいので、下記の例をご覧ください。

2.10.1 例

```
#define VERSION 2

#if VERSION == 1
    cn4 127, 24
#else
    #if VERSION == 2
        dn4 127, 24
    #else
        en4 127, 24
    #endif
#endif
    fin
```

といったものを、`#elif`を使うと、

```
#define VERSION 2

#if VERSION == 1
    cn4 127, 24
#elif VERSION == 2
    dn4 127, 24
#else
    en4 127, 24
#endif
fin
```

のように、簡潔に書くことができます。

3 サウンドアーカイバ sndarc

3.1 概要

サウンドアーカイバ **sndarc** は、複数のサウンドデータファイルを1つのファイルにまとめることができるコマンドラインツールです。各種サウンドデータを一度にコンバートすることもできます。またコンバートは、ファイル更新日時を調べて、必要最低限なものだけ実行されるようになっています。

通常、SoundPlayer 開発環境で **MakeSound.bat** を実行すると、自動的にサウンドアーカイバが起動されるので、意識して使うことはありません。ただし、下記で説明しているオプションなどの細かい指定をしたい場合は、直接コマンドラインで、サウンドアーカイバを起動する必要があります。

3.2 実行ファイルの場所

sndarc.exe は、`$NitroSystem/tools/win/bin` に入っています。また、各種サウンドデータのコンバートには、同じディレクトリの **seqconv.exe** などを使用します。

3.3 使用方法

3.3.1 書式

コマンドラインの文法と引数は、次の通りです。

```
sndarc [options] <inputfile>
```

<inputfile>には、サウンドアーカイブ定義ファイルを指定します。

3.3.2 実行結果

コンバートを実行すると、<inputfile>の拡張子を、***.sdat** に変えたサウンドアーカイブファイルが出力されます。また、以下の3つのファイルも同時に出力されます。(これらのファイル名は、オプションで変更することができます。)

- サウンドラベルファイル(***.sادل**)
- サウンドアーカイブラベルファイル(***.sادل**)
- サウンドマップファイル(***.smap**)

3.3.2.1 サウンドラベルファイル

拡張子***.sادل** のサウンドラベルファイルは、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を、**#define** を使ってインデックス番号として定義したファイルです。さらにこのファイルは、シーケンスアーカイブのコンバートで出力される拡張子***.ssادل** のシーケンスアーカイブラベルファイルの内容も含んでいます。

このファイルを、プログラムからインクルードすることで、サウンドデータで定義されたラベルをプログラムでも使えるようになります。

このファイルは、各種データのコンバート後に生成されますので、テキストのサウンドデータから**#include** を使って、取り込むことはできません。

3.3.2.2 サウンドアーカイブラベルファイル

拡張子*.sddl のサウンドアーカイブラベルファイルは、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を、`#define` を使ってインデックス番号として定義したファイルです。

このファイルは、各種データのコンバート前に生成されますので、シーケンスアーカイブのテキストファイルなどから、`#include` を使って、取り込むことができます。

3.3.2.3 サウンドマップファイル

拡張子*.smap のサウンドマップファイルは、サウンドアーカイブ中にどのようなデータが格納されているかといった情報が記述されたテキストファイルです。

3.3.3 オプション

オプションには、次のものがあります。

表 3-1 sndarc オプション一覧

オプション	説明
<code>-c</code>	ファイル更新日時を無視して、全てのファイルをコンバートします。
<code>-b</code>	サウンドアーカイブにシンボルデータを出力しません。
<code>-s, --silent</code>	内部で実行中のコマンドを表示しません。
<code>-v, --verbose</code>	細かい内部動作を表示します。
<code>-h, --help</code>	ヘルプ表示を行います。
<code>-o, --sdat filename</code>	サウンドアーカイブファイル(*.sdat)のファイル名を指定します。サウンドマップファイル(*.smap)のファイル名は、ここで指定したファイル名の拡張子を変えたものになります。
<code>--sddl filename</code>	サウンドラベルファイル(*.sddl)のファイル名を指定します。
<code>--sddl filename</code>	サウンドアーカイブラベルファイル(*.sddl)のファイル名を指定します。
<code>-f filename</code>	前処理ファイルを指定します。前処理ファイルは、引数で指定したサウンド定義ファイルの前に処理されます。複数指定した場合、指定した順に処理されます。
<code>--align byte</code>	サウンドデータ中のサウンドデータが、指定したバイト数で、アライメントされます。
<code>--convert flags</code>	コンバートを実行するファイルタイプを指定します。(詳細は後述)

3.3.4 コンバートファイルタイプ

`--convert` オプションを指定すると、コンバートを実行するファイルタイプを指定することができます。例えば、`--convert s` とすると、シーケンスアーカイブファイルのみコンバートされ、その他のバンクやストリームファイルなどはコンバートされません。そのため、シーケンスアーカイブファイルのみを変更した場合に、このような指定を行うと、コンバート時間を短縮させることができます。

指定できるファイルタイプには、以下のものがあります。

表 3-2 コンバートファイルタイプ

ファイルタイプ	説明
a	全てのファイルタイプ (デフォルト)
q	シーケンスファイル
s	シーケンスアーカイブファイル
b	バンクファイル
w	波形アーカイブファイル
m	ストリームファイル
n	どのファイルタイプもコンバートしない

これらは、`--convert sqm` のように複数指定することもできます。

なお、変更しているファイルをコンバートしないように指定すると、不正なデータが出力される可能性があることに注意してください。

3.3.5 サウンドデータのDMA転送

サウンドアーカイブ中のサウンドデータをロードするとき、DMA 転送を使いたい場合は、ロードするデータのオフセット及びサイズが、512 バイトアライメントされている必要があります。通常、オフセットは 32 バイトアライメントで、サイズは実データのファイルサイズになりますが、`--align` オプションを指定することで、オフセット及びサイズを 512 バイトアライメントすることができます。

ただし、以下の点に注意してください。

3.3.5.1 サウンドアーカイブのオフセット

`--align` オプションとは別に、サウンドアーカイブ自体のオフセットを 512 バイトアライメントしておく必要があります。

3.3.5.2 アライメントされるデータ

`--align` オプションが有効なのは、以下のサウンドデータの配置に対してです。

- ストリームデータ(*.strm)
- シーケンスデータ(*.sseq)
- シーケンスアーカイブデータ(*.ssar)
- バンクデータ(*.sbnk)
- 波形アーカイブデータ(*.swar)

波形データ(*.swav)には効果がありませんので、波形データの個別ロード機能使用時は、DMA 転送できませんので、注意してください。

4 SMFコンバータ smfconv

4.1 概要

SMF コンバータ `smfconv` は、フォーマット 0 または 1 のスタンダード MIDI ファイル(以下、SMF)を、テキスト形式のシーケンスファイルにコンバートします。出力したテキスト形式のシーケンスファイルは、シーケンスコンバータ `seqconv` でバイナリファイルにコンバートすることができます。

通常は、サウンドアーカイバ `sndarc` から自動的に呼びだされるので、これを直接使う必要はありません。

4.2 実行ファイルの場所

`smfconv.exe` は、`$NitroSystem/tools/win/bin` に入っています。

4.3 使用方法

4.3.1 書式

コマンドラインの文法と引数は、次の通りです。

```
smfconv [options] <inputfile>
```

<inputfile>には、SMF を指定します。

4.3.2 実行結果

コンバートを実行すると、<inputfile>の拡張子を、*.smft に変えたファイルが出力されます。

4.3.3 オプション

オプションには、次のものがあります。

表 4-1 smfconv オプション一覧

オプション	説明
-o <filename>	出力ファイル名を指定します。
-v, --verbose	細かい内部動作を表示します。
-u, --update	入力ファイルが更新されたときのみコンバートします。
-h, --help	ヘルプ表示を行います。

4.4 詳細

4.4.1 トラック

トラックは最大 16 個使用できます。チャンネル番号 1～16 がそれぞれ、トラック番号 0～15 に対応します。またトラック 0 には、テンポチェンジなど、シーケンス全体に効果のある MIDI イベントも混ぜて出力されます。

4.4.2 ループ指定

SMF で全トラックを同じタイミングでループさせたい場合は、MIDI シーケンサ上でマーカースとして、半角の角括弧（ [,] ）を書き込んでください。開き括弧 [の位置を始点、閉じ括弧] の位置を終点としてループするようにコンバートされます。

4.4.3 SMF先頭部の空白

SMF をコンバートしたとき、最初のノートオンまでの空白部分は自動的にカットされます。カットされないようにするためには、シーケンスの先頭に、音量の小さいダミーのノートを加えてください。

4.4.4 MIDIイベント

コンバートされる MIDI イベントについては、「シーケンスデータマニュアル」にあるシーケンスコマンド一覧表をご覧ください。

5 シーケンスコンバータ seqconv

5.1 概要

シーケンスコンバータ `seqconv` は、テキスト形式のシーケンスファイルをバイナリに変換するコマンドラインツールです。SMF コンバータ `smfconv` が出力する SMFT 形式のテキストファイルや、シーケンスアーカイブ形式のシーケンスファイルをコンバートすることができます。

通常は、サウンドアーカイバ `sndarc` から自動的に呼びだされるので、これを直接使う必要はありません。

5.2 実行ファイルの場所

`seqconv.exe` は、`$NitroSystem/tools/win/bin` に入っています。

5.3 使用方法

5.3.1 書式

コマンドラインの文法と引数は、次の通りです。

```
seqconv [options] <inputfile>
```

<inputfile>には、テキストフォーマットで記述したファイルを指定します。シーケンスアーカイブ形式のファイルをコンバートするときは、`--archive` オプションを指定しなければなりません。

5.3.2 実行結果

コンバートを実行すると、<inputfile>の拡張子を、`*.sseq` に変えたファイルが出力されます。シーケンスアーカイブをコンバートすると、拡張子を`*.ssar` に変えたファイルが出力されます。

シーケンスアーカイブをコンバートしたときには、さらに拡張子`*.ssdl` のシーケンスアーカイブラベルファイルも出力されます。これは、`#define` を使って、シーケンステーブルで設定したシーケンスラベルをインデックス番号として定義したファイルです。

5.3.3 オプション

オプションには、次のものがあります。

表 5-1 `seqconv` オプション一覧

オプション	説明
<code>-a, --archive</code>	シーケンスアーカイブをコンバートします。
<code>-o <filename></code>	出力ファイル名を指定します。
<code>-I <dir></code>	インクルードパスを指定します。

-v, --verbose	細かい内部動作を表示します。
-u, --update	入力ファイルが更新されたときのみコンバートします。
-h, --help	ヘルプ表示を行います。

6 バンクコンバータ bankconv

6.1 概要

バンクコンバータ **bankconv** は、テキストのバンク定義ファイルをバイナリに変換するコマンドラインツールです。バンクコンバートを実行すると、バンクファイルに登録されている **AIFF** や **WAV** などの波形ファイルもコンバートされます。またコンバートは、ファイル更新日時を調べて、必要最低限なぶんだけ実行されるようになっています。

通常は、サウンドアーカイバ **sndarc** から自動的に呼びだされるので、これを直接使う必要はありません。

6.2 実行ファイルの場所

bankconv.exe は、`$NitroSystem/tools/win/bin` に入っています。また、波形ファイルのコンバートには、同じディレクトリの **waveconv.exe** を使用します。

6.3 使用方法

6.3.1 書式

コマンドラインの文法と引数は、次の通りです。

```
bankconv [options] <inputfile>
```

<inputfile>には、バンク定義ファイルを指定します。

6.3.2 2つのモード

bankconv は2つのモードで起動します。1つは複数のバンク定義ファイルから、波形ファイル情報を抽出して1つの波形リストファイルを出力するモード「波形リストファイル出力モード」です。もう1つは、バンク定義ファイルをバンクバイナリファイルを出力する「バンクコンバートモード」です。

-l オプションを指定すると、「波形リストファイル出力モード」になり、指定しないと、「バンクコンバートモード」になります。

6.3.3 波形リストファイル出力モード

6.3.3.1 波形グループ番号

入力ファイル名に続けて、波形グループ番号を指定します。

```
bankconv -l bank1.bnk:2 bank2.bnk:1
```

ファイル名だけ指定すると、波形グループ番号 **0** を指定したことになります。波形グループ番号は、バンク定義ファイル中で **@WGROUP** により指定した値です。指定した波形グループに登録された波形ファイルのみを抽出します。

@WGROUP を使っていない場合は、全ての波形ファイルは波形グループ **0** に登録されています。

6.3.3.2 実行結果

複数のバンク定義ファイルから、波形ファイル情報を抽出して、波形リストファイルを出力します。波形リストファイル名は、`-o` オプションで指定する必要があります。生成された波形リストファイルは、「バンクコンバートモード」や、波形アーカイバ `wavearc` で使用することができます。

複数のバンク定義ファイル中で同じ波形ファイルを使用している場合は、1つの波形ファイルとして登録されます。

抽出された波形ファイルに対しては、波形コンバータ `waveconv` でコンバートがかけられます。

6.3.4 オプション

オプションには、次のものがあります。

表 6-1 `bankconv` 波形リストファイル出力モードオプション一覧

オプション	説明
<code>-l</code>	波形リストファイル出力モードにします。(必須)
<code>-c</code>	ファイル更新日時を無視して、全ての波形ファイルをコンバートします。
<code>-o <filename></code>	出力ファイル名を指定します。(必須)
<code>-I <dir></code>	インクルードパスを指定します。
<code>-s, --silent</code>	内部で実行中のコマンドを表示しません。
<code>-v, --verbose</code>	細かい内部動作を表示します。
<code>-u, --update</code>	入力ファイルが更新されたときのみコンバートします。
<code>-h, --help</code>	ヘルプ表示を行います。

6.3.5 バンクコンバートモード

6.3.5.1 実行結果

コンバートを実行すると、`<inputfile>`の拡張子を、`*.sbnk` に変えたファイルが出力されます。波形ファイルを使用する場合は、各波形セットに対応した波形リストファイルを、オプション`--wave0` から`--wave3` で指定する必要があります。

拡張子`*.spdl` のプログラムナンバリストファイルも出力されます。これは、`#define` を使って、各インストゥルメントに対し設定したラベルをプログラムナンバとして定義したファイルです。

6.3.5.2 オプション

オプションには、次のものがあります。

表 6-2 `bankconv` バンクコンバートモードオプション一覧

オプション	説明
<code>--wave0 <filename></code>	波形セット 0 番の波形リストファイルを指定します。
<code>--wave1 <filename></code>	波形セット 1 番の波形リストファイルを指定します。
<code>--wave2 <filename></code>	波形セット 2 番の波形リストファイルを指定します。

--wave3 <filename>	波形セット 3 番の波形リストファイルを指定します。
-o <filename>	出力ファイル名を指定します。
--arglist <filename>	入力ファイル名を、指定したファイルに記述して渡します。
-I <dir>	インクルードパスを指定します。
-s, --silent	内部で実行中のコマンドを表示しません。
-v, --verbose	細かい内部動作を表示します。
-u, --update	入力ファイルが更新されたときのみコンバートします。
-h, --help	ヘルプ表示を行います。

7 波形ファイルコンバータ waveconv

7.1 概要

波形ファイルコンバータ `waveconv` は、AIFF または WAV 形式の波形ファイルを、NITRO-Composer 専用波形ファイルフォーマット、SWAV 形式のファイルに変換するコマンドラインツールです。

通常は、サウンドアーカイバ `bankconv` から自動的に呼びだされるので、これを直接使う必要はありません。

7.2 実行ファイルの場所

`waveconv.exe` は、`$NitroSystem/tools/win/bin` に入っています。

7.3 使用方法

7.3.1 書式

コマンドラインの文法と引数は、次の通りです。

```
waveconv [options] <inputfile>
```

<inputfile>には、波形ファイルを指定します。

7.3.2 実行結果

コンバートを実行すると、<inputfile>の拡張子を、*.swav に変えたファイルが出力されます。

7.3.3 オプション

オプションには、次のものがあります。

表 7-1 waveconv オプション一覧

オプション	説明
-s, --pcm16	16bit PCM 形式にコンバートします。
-c, --pcm8	8bit PCM 形式にコンバートします。
-a, --adpcm	ADPCM 形式にコンバートします。
-o <filename>	出力ファイル名を指定します。
-v, --verbose	細かい内部動作を表示します。
-u, --update	入力ファイルが更新されたときのみコンバートします。
-h, --help	ヘルプ表示を行います。

7.4 波形フォーマット

7.4.1 ファイル形式

扱える波形ファイルのフォーマットは、「Audio Interchange File Format(AIFF)」「Wave(Microsoft)」の2種類です。

7.4.2 チャンネル数

チャンネル数は1、すなわちモノラルでなければなりません。

7.4.3 量子化ビット数

量子化ビット数は、コンバータで自動的に出力形式に変換されますので制限はありませんが、できるだけ出力形式に揃えておくことが望ましいです。例えば、8bitPCM で出力する場合は8ビットに、ADPCM で出力する場合には16ビットにします。

7.4.4 サンプリングレート

サンプリングレートはほぼ無制限で、現実的な範囲である4kHz～44.1kHz程度は使用可能です。サンプリングレートが高いほど音質は良くなりますが、データ転送の頻度が多くなり処理負荷増大の原因になりますので、注意してください。

7.4.5 オリジナルキー

オリジナルキーは、波形ファイルで設定せずに、バンクファイルに記述します。波形ファイルに設定しても無視されます。

7.4.6 ループ

1種類のループに対応しています。

7.5 詳細

7.5.1 ループ補正

波形コンバータ waveconv では、コンバート時にループ補正を行っています。

ニンテンドーDS ではハードの仕様上、ループの長さがある値の倍数(ADPCM の場合は8の倍数)で指定しなければなりません。これは、波形データ作成の作業上、非常に面倒ですので、コンバート時にループの長さがその値の倍数になるように、サンプリングレートを変換しています。

例えば、ループの長さが500だった場合、8の倍数の504に補正するため、サンプリングレートを $504 \div 500$ 、つまり1.008倍しています。その分、波形データサイズが増大することになります。

7.5.2 ループスタート位置の限界

ニンテンドーDS ではハードの仕様上、ループの開始位置を大きく後ろへ設定することはできません。

ループ開始位置の限界は、次のようにフォーマット毎に異なります。またこの限界値は、上記ループ補正を行った後の値であることに注意してください。

表 7-2 ループスタート位置の限界

フォーマット	最大値[sample]
ADPCM	524272
16bit PCM	131070
8bit PCM	262140

ループスタート位置を限界値より後ろへ設定したものをコンバートすると、エラーとなります。

8 波形ファイルアーカイバ wavearc

8.1 概要

波形ファイルアーカイバ **wavearc** は、波形ファイルのリストを元に、複数の波形ファイルを1つのファイルにまとめるコマンドラインツールです。波形ファイルは、波形ファイルコンバータ **waveconv** が出力した **SWAV** 形式の波形ファイルです。

通常は、サウンドアーカイバ **sndarc** から自動的に呼びだされるので、これを直接使う必要はありません。

8.2 実行ファイルの場所

wavearc.exe は、`$NitroSystem/tools/win/bin` に入っています。

8.3 使用方法

8.3.1 書式

コマンドラインの文法と引数は、次の通りです。

```
wavearc [options] <inputfile>
```

<inputfile>には、波形ファイルのリストを指定します。テキスト形式で、1行毎に1つの波形ファイルのファイル名を書き連ねたものです。

コンバートを実行すると、<inputfile>の拡張子を、*.swar に変えたファイルが出力されます。

8.3.2 オプション

オプションには、次のものがあります。

表 8-1 wavearc オプション一覧

オプション	説明
-o <filename>	出力ファイル名を指定します。
-v, --verbose	細かい内部動作を表示します。
-u, --update	入力ファイルが更新されたときのみコンバートします。
-h, --help	ヘルプ表示を行います。

9 ストリームコンバータ strmconv

9.1 概要

ストリームコンバータ `strmconv` は、AIFF または WAV 形式の波形ファイルを、NITRO-Composer 専用ストリームファイルフォーマット、STRM 形式のファイルに変換するコマンドラインツールです。

通常は、サウンドアーカイバ `sndarc` から自動的に呼びだされるので、これを直接使う必要はありません。

9.2 実行ファイルの場所

`strmconv.exe` は、`$NitroSystem/tools/win/bin` に入っています。

9.3 使用方法

9.3.1 書式

コマンドラインの文法と引数は、次の通りです。

```
strmconv [options] <inputfile>
```

<inputfile>には、波形ファイルを指定します。

9.3.2 実行結果

コンバートを実行すると、<inputfile>の拡張子を、*.strm に変えたファイルが出力されます。

9.3.3 オプション

オプションには、次のものがあります。

表 9-1 strmconv オプション一覧

オプション	説明
<code>-s, --pcm16</code>	16bit PCM 形式にコンバートします。
<code>-c, --pcm8</code>	8bit PCM 形式にコンバートします。
<code>-a, --adpcm</code>	ADPCM 形式にコンバートします。
<code>-o <filename></code>	出力ファイル名を指定します。
<code>-v, --verbose</code>	細かい内部動作を表示します。
<code>-u, --update</code>	入力ファイルが更新されたときのみコンバートします。
<code>-h, --help</code>	ヘルプ表示を行います。

9.4 波形フォーマット

9.4.1 ファイル形式

扱える波形ファイルのフォーマットは、「Audio Interchange File Format(AIFF)」「Wave(Microsoft)」の2種類です。

9.4.2 チャンネル数

チャンネル数は任意です。

9.4.3 量子化ビット数

量子化ビット数は、コンバータで自動的に出力形式に変換されますので制限はありませんが、できるだけ出力形式に揃えておくことが望ましいです。例えば、8bitPCM で出力する場合は 8 ビットに、ADPCM で出力する場合には 16 ビットにします。

9.4.4 サンプリングレート

サンプリングレートについてのコンバータ上の制限はありません。サンプリングレートが高いほど音質は良くなりますが、データ転送の頻度が多くなり処理負荷増大の原因になりますので、注意してください。

9.4.5 ループ

1種類のループに対応しています。

9.5 リサンプリング

ストリームコンバータ `strmconv` では、コンバート時にリサンプリングを行うことがあります。

NITRO-Composer では任意のサンプリングレートでストリーム再生を行うことができません。これは、波形データ作成の作業上、面倒ですので、コンバート時に再生可能なサンプリングレートになるように、サンプリングレートを変換しています。そのため、変換後のデータサイズは若干大きくなります。

© 2004-2008 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。