

# NITRO-Composer

## シーケンスデータマニュアル

2008-05-30

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	概要 .....	8
1.1	シーケンスファイル .....	8
1.2	シーケンスアーカイブ .....	8
2	基本事項 .....	9
2.1	タイムベース .....	9
2.2	コメント .....	9
2.3	ラベル .....	9
2.3.1	ラベルに使える文字 .....	9
2.3.2	ローカルラベル .....	9
3	シーケンスファイル .....	10
3.1	概要 .....	10
3.2	スタンダードMIDIファイル .....	10
3.2.1	SMFフォーマット .....	10
3.2.2	トラック .....	10
3.2.3	ループ指定 .....	10
3.2.4	SMF先頭部の空白 .....	10
3.2.5	MIDIイベント .....	11
3.2.6	MusicPlayer2000 との互換性 .....	11
3.3	テキストシーケンス .....	11
3.3.1	概要 .....	11
3.3.2	トラックオープン処理 .....	12
3.3.3	トラック処理 .....	12
3.4	SMFへのテキストコマンド埋め込み .....	13
3.4.1	概要 .....	13
3.4.2	埋め込み方 .....	13
3.4.3	全トラック出力 .....	13
3.4.4	ラベル名展開 .....	13
3.4.5	リアルタイムMIDI再生時の注意 .....	14
4	シーケンスアーカイブ .....	15
4.1	概要 .....	15
4.2	サウンドアーカイブラベルファイルの利用 .....	16
4.3	構成 .....	16
4.3.1	シーケンステーブル .....	16
4.3.2	シーケンスデータ .....	17
4.4	シーケンステーブル .....	17
4.4.1	シーケンスラベル名 .....	18
4.4.2	データラベル名 .....	18
4.4.3	バンク .....	18
4.4.4	ボリューム .....	18

4.4.5	発音プライオリティ .....	19
4.4.6	プレイヤープライオリティ .....	19
4.4.7	プレイヤー .....	19
4.5	数値表記 .....	19
4.5.1	2進数、16進数表記 .....	19
4.5.2	ビット表記 .....	19
4.5.3	数式 .....	20
5	シーケンスコマンド .....	21
5.1	ノートコマンド .....	22
5.2	休符コマンド .....	23
5.3	シーケンスの終了 .....	23
5.4	プログラムチェンジ .....	23
5.5	テンポチェンジ .....	23
5.6	トラックボリューム .....	23
5.7	メインボリューム .....	24
5.8	ピッチベンド .....	24
5.9	トランスポーズ .....	24
5.10	トラックパン .....	24
5.11	発音プライオリティ .....	25
5.12	タイモード .....	25
5.13	ノートウェイトモード .....	25
5.14	ミュート .....	25
5.15	ポルタメント .....	26
5.16	スワイプ .....	26
5.17	トラック確保 .....	27
5.18	トラック起動 .....	27
5.19	シーケンスジャンプ .....	27
5.20	シーケンス呼び出し .....	27
5.21	ループ .....	28
5.22	エンベロープ .....	28
5.23	モジュレーション .....	28
6	拡張シーケンスコマンド .....	30
6.1	概要 .....	30
6.2	ランダムコマンド .....	30
6.2.1	書式 .....	30
6.2.2	ランダムコマンド一覧 .....	30
6.2.3	ランダム音長ノートコマンド .....	31
6.3	変数コマンド .....	31
6.3.1	変数とは? .....	31
6.3.2	変数演算コマンド .....	32

6.3.3	変数コマンド .....	32
6.3.4	書式 .....	32
6.3.5	変数コマンド一覧 .....	32
6.3.6	音長変数ノートコマンド .....	33
6.3.7	変数間演算コマンド .....	33
6.4	条件コマンド .....	33
6.4.1	比較コマンド .....	33
6.4.2	条件コマンド .....	34
6.5	プログラムとの通信 .....	34
6.5.1	MIDIシーケンスとの通信 .....	34
6.5.2	条件コマンドとの組み合わせ .....	34
6.6	変数のデバッグ .....	35
7	付録 .....	36
7.1	全シーケンスコマンド一覧表 .....	36
7.2	MIDIコントロールコード対応表 .....	38
7.2.1	RPN対応表 .....	39
7.3	MusicPlayer2000 用SMFとの互換性 .....	40
7.3.1	トラック番号 .....	40
7.3.2	モジュレーション .....	40
7.3.3	チューニング .....	40
7.3.4	擬似エコー .....	40
7.3.5	プライオリティ .....	40
7.3.6	トラック単位のループ .....	40
7.3.7	音量計算 .....	40
7.3.8	効果音など作成時のテンポ .....	40
7.3.9	4分音符分解能 .....	40

## コード

コード 3-1	テキストシーケンスファイル .....	11
コード 3-2	トラックオープン処理 .....	12
コード 3-3	トラック処理 .....	13
コード 4-1	テキストシーケンスアーカイブ .....	15
コード 4-2	シーケンステーブル .....	16
コード 4-3	シーケンスデータ .....	17
コード 4-4	シーケンステーブル .....	17
コード 5-1	ノートコマンドの例 .....	23
コード 6-1	変数間演算の例 .....	33
コード 6-2	条件コマンドの例 .....	34
コード 6-3	条件コマンドとの組み合わせ例 .....	34

## 表

表 4-1	シーケンステーブル要素 .....	18
-------	-------------------	----

表 4-2 演算子 .....	20
表 5-1 シーケンスコマンド一覧表 .....	21
表 5-2 ミュートコマンドの引数 .....	26
表 5-3 モジュレーション変化の度合い .....	29
表 5-4 モジュレーションタイプ一覧 .....	29
表 6-1 ランダムコマンド一覧表 .....	30
表 6-2 変数演算コマンド一覧 .....	32
表 6-3 変数コマンド一覧表 .....	32
表 6-4 比較コマンド一覧 .....	33
表 6-5 変数デバッグコマンド一覧 .....	35
表 7-1 全シーケンスコマンド一覧表 .....	36
表 7-2 MIDIコントロールコード対応表 .....	38



図 6-1 ローカル変数とグローバル変数 .....	31
----------------------------	----

## 改訂履歴

改訂日	改訂内容
2008-05-30	1. NITRO-System の名称変更による修正 (NITRO-System を TWL-System に変更)。
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
2007-11-26	1. エンベロープコマンドの無効化に関する説明追加 2. MIDI RPN 対応 3. call/ret、loop_start/loop_end コマンドに関する注意書き追加
2006-05-29	1. mute コマンドの説明追加 2. SMF へのテキストコマンド埋め込みにおけるラベル名展開機能の拡張に伴う説明追加 3. 誤記修正
2005-03-28	1. 数値表記に関する説明追加
2005-01-31	1. printvar コマンドの説明追加 2. "NITRO"を"ニンテンドーDS"に変更
2004-11-10	1. volume と volume2 コマンドの違いの説明を詳しくした 2. randvar コマンドの説明を詳しくした 3. 誤字修正
2004-10-12	1. シーケンスアーカイブの@SEQ_TABLE で、プレイヤーをラベルで指定する方法の説明追加、
2004-09-16	1. loop_start コマンドの引数に 0 を入れられるようにし、無限ループとして扱うようにした 2. SMF から loop_start、loop_end コマンドを使えるようにした 3. SMF でのループマーカーに、"loop_start"および"loop_end"を使えるようにした 4. SMF 先頭部の空白に関する説明修正 5. SMF でのトラック単位のループ機能追加に伴い、「MusicPlayer2000 用 SMF との互換性」を修正 6. シーケンスアーカイブの@SEQ_TABLE で、バンクをラベルで指定する方法の説明追加
2004-09-02	1. SMF へのテキストコマンド埋め込みに関する説明追加
2004-08-10	1. smfconv の出力形式が変更になったことに伴う修正
2004-07-20	1. スタイルの調整
2004-06-01	1. 拡張シーケンスコマンド追加 2. 全シーケンスコマンド一覧表追加 3. MIDI コントロールコード対応表追加 4. シーケンスアーカイブのシーケンスラベル名にインデックス番号を指定する方法を記述 5. 1つのプレイヤーで複数のシーケンスを再生できるようになったことに伴い、シーケンスアーカイブのシーケンステーブルでプレイヤー番号に関する記述修正 6. プレイヤープライオリティの説明修正 7. MIDI コントロールチェンジ(29)を、値を 24 倍して、sweep_pitch コマンドへ変換 8. prg、mod_delay コマンドの最大値を 65535 から 32767 に変更 9. volume2 コマンド追加 10. bendrange コマンドの値の範囲を明記
2004-04-12	1. シーケンステーブルの説明を章に分割 2. シーケンスラベルとデータラベルの名称の区別をつける

	3. シーケンスラベルの2重定義の注意書き追加
2004-04-01	<ol style="list-style-type: none"><li>1. MIDI コントロールチェンジ(84)の porta への割り当て</li><li>2. エンベロープコマンド追加</li><li>3. SMF の MusicPlayer2000 との互換性情報を記載</li><li>4. loop_start/loop_end コマンド追加</li><li>5. タイモードでは単音発音しかできないという記述追加</li><li>6. jump コマンド使用時の無限ループに対する注意事項追加</li><li>7. mod_range が音程変化のみしか効かないと書かれていた誤りを削除</li><li>8. 発音プライオリティ比較に関する誤りを訂正</li></ol>
2004-03-18	全体的な構成の変更 テキストフォーマット仕様変更にとまなう修正 ノートコマンド length 0 の時の特別仕様に関して追記 ポルタメント・スウィープの仕様拡張に関して追記
2004-03-01	初版

# 1 概要

サウンドシーケンスは、スタンダード MIDI ファイルのような楽譜データに相当するデータフォーマットです。時間軸に沿って、発音や音量の変更などの処理を行うようなコマンドが記述されています。シーケンスデータは単なるコマンドの集合であるため、実際には音源データに相当するバンクデータを使って音を鳴らします。

シーケンスフォーマットには、大きく分けて2種類あります。普通のシーケンスファイルと、複数のシーケンスデータを1つのファイルにまとめたシーケンスアーカイブです。まずは、この2種類について簡単に説明します。

## 1.1 シーケンスファイル

---

単にシーケンスファイルと呼ぶときは、この1ファイル1シーケンスの形式のデータを指します。

シーケンスファイルは、スタンダード MIDI ファイルと同じものと見なすことができます。SMF コンバータ `smfconv` を使えば、スタンダード MIDI ファイルをこの形式のシーケンスデータに変換することができます。

## 1.2 シーケンスアーカイブ

---

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をシーケンスアーカイブと呼びます。

このデータへは、`smfconv` を使って変換することができませんので、テキストエディタで作成することになります。シーケンスアーカイブを使うことによって、データサイズを小さく抑えることができます。また多くのシーケンスを1つのファイルで扱えるため、単純なシーケンスデータの場合、短時間で作成することができます。



## 2 基本事項

### 2.1 タイムベース

---

タイムベース(四分音符分解能)は、48です。

テンポ240の時、ちょうど1ティックと1回のサウンド処理の長さが同じになります。なので、テンポを240の約数にしておけば、発音などのばらつきは最小限に抑えられます。なお、テンポのデフォルト値は、120です。

### 2.2 コメント

---

テキスト形式のシーケンスファイル、シーケンスアーカイブともに、コメントは;(セミコロン)で始まり、改行までです。

```
;;; comment

Track_0:  ; comment
```

### 2.3 ラベル

---

文字列の後に:(コロン)を書くと、ラベル定義になります。ラベルを定義することで、その位置を別の場所から指定することができるようになります。

#### 2.3.1 ラベルに使える文字

---

ラベル名は、1文字目がアルファベットで始まり、2文字目以降にはアルファベットに加えて、数字及びアンダースコア(\_)が使えます。

```
LoopStart:
test_seq:
Track_01
```

#### 2.3.2 ローカルラベル

---

ラベル名の1文字目をアンダースコア(\_)にすると、そのラベルはローカルラベルとして扱われます。

ローカルラベルとは、別のローカルラベルでないラベルに挟まれた区間でのみ有効なラベルです。従って、別の区間では、同じローカルラベル名を使うことができます。

```
label_1:
    ;; ここでは、3行目の_localが使える
_local:
    ;; ここでは、3行目の_localが使える

label_2:
    ;; ここでは、7行目の_localが使える
_local:
    ;; ここでは、7行目の_localが使える

label3:
    ;; ここでは、_localが使えない
```

## 3 シーケンスファイル

### 3.1 概要

---

シーケンスファイルは、1つのシーケンスを1つのファイルに格納したものです。通常、スタンダード MIDI ファイルをコンバートして作成します。

スタンダード MIDI ファイルは、1度テキスト形式のシーケンスファイルに変換されます。テキスト形式のシーケンスファイルでは、後で説明するシーケンスアーカイブと同じシーケンスコマンドを使って記述されています。

テキストシーケンスファイルは、最終的にシーケンスコンバータ `seqconv` により、拡張子 `*.sseq` のバイナリデータに変換された後、TWL またはニンテンドーDS で再生されることになります。

### 3.2 スタンダードMIDIファイル

---

スタンダード MIDI ファイル(以下、SMF)は、市販のシーケンサなどで作成します。以下、SMF を作成する上での、注意事項を記します。

#### 3.2.1 SMFフォーマット

---

SMF フォーマットは、フォーマット 0 または、1 のみ使用できます。フォーマット 2 には対応していません。

#### 3.2.2 トラック

---

トラックは最大 16 個使用できます。ただし、NITRO-Composer システム全体として、合計 32 トラックまでしか使用できません。

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。

#### 3.2.3 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。(開き括弧 [ および、閉じ括弧 ] は、それぞれ "loop\_start" および "loop\_end" という文字列で代用することもできます。)

また、トラック単位でループさせたい場合は、ループ始点に値が 0 のコントロールチェンジ 89 を、ループ終点にコントロールチェンジ 90(値は任意)を、入れます。

#### 3.2.4 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。

ただし、最初のノートオンより前の位置に、ループスタートがあった場合などは、ループスタート位置よりまえの部分のみカットされます。

### 3.2.5 MIDIイベント

コンバートされるMIDIイベントについては、「5 シーケンスコマンド」にあるシーケンスコマンド一覧表を参照してください。

### 3.2.6 MusicPlayer2000 との互換性

AGB用サウンドドライバであるMusicPlayer2000 用に作成したSMFは、そのままコンバートすることは可能ですが、いくつかの仕様が変更されていますので、正確に再現するためには修正が必要です。詳しくは、「7.3MusicPlayer2000 用SMFとの互換性」を参照してください。

## 3.3 テキストシーケンス

### 3.3.1 概要

smfconv を使って、SMF をコンバートすると、下記のようなテキストシーケンスファイルが生成されます。

#### コード 3-1 テキストシーケンスファイル

```
;;;;;;;;;;;;;
;
; mid/kart64_title.smft
;   generated by smfconv
;
;;;;;;;;;;;;;

SMF_kart64_title_Begin:
    alloctrack 0x0eff
SMF_kart64_title_Start:
    opentrack 1, SMF_kart64_title_Track_1
    opentrack 2, SMF_kart64_title_Track_2
    opentrack 3, SMF_kart64_title_Track_3
    opentrack 4, SMF_kart64_title_Track_4
    opentrack 5, SMF_kart64_title_Track_5
    opentrack 6, SMF_kart64_title_Track_6
    opentrack 7, SMF_kart64_title_Track_7
    opentrack 9, SMF_kart64_title_Track_9
    opentrack 10, SMF_kart64_title_Track_10
    opentrack 11, SMF_kart64_title_Track_11

;;;;;;;;;;;;;
; Track 0
;;;;;;;;;;;;;

SMF_kart64_title_Track_0:
    notewait_off
; Measure 1 -----
    tempo      140
    prg        0
SMF_kart64_title_Track_0_LoopStart:
    wait       1
    volume     127
    pan        64
    bendrange  2
    pitchbend  0
    mod_speed  16
    mod_depth  0
    wait       767
```

```
; Measure 2 -----
```

このように、テキストシーケンスファイルでは、シーケンスコマンドの羅列で書かれています。シーケンスコマンドは、基本的に上から順番に処理されます。

各シーケンスコマンドの説明は、「5 シーケンスコマンド」を参照してください。

### 3.3.2 トラックオープン処理

一番はじめのコメントの後に、トラックのオープン処理が書かれています。

#### コード 3-2 トラックオープン処理

```
SMF_kart64_title_Begin:
    alloctrack 0x0eff
SMF_kart64_title_Start:
    opentrack 1, SMF_kart64_title_Track_1
    opentrack 2, SMF_kart64_title_Track_2
    opentrack 3, SMF_kart64_title_Track_3
    opentrack 4, SMF_kart64_title_Track_4
    opentrack 5, SMF_kart64_title_Track_5
    opentrack 6, SMF_kart64_title_Track_6
    opentrack 7, SMF_kart64_title_Track_7
    opentrack 9, SMF_kart64_title_Track_9
    opentrack 10, SMF_kart64_title_Track_10
    opentrack 11, SMF_kart64_title_Track_11
```

まず始めに覚えておかなければならないことは、1番始めのシーケンスデータは、トラック0で処理されるということです。トラック0はシーケンスがスタートした時点ですでに確保されていて、そこからシーケンス処理が始まります。複数のトラックを使いたい場合は、トラック0から別のトラックを確保し、起動させる必要があります。

2行目の `alloctrack` コマンドは、トラックを確保するコマンドです。このコマンドは、シーケンス開始直後しか使えませんので、注意してください。引数はビットマスクで、下位ビットから順にトラック0、トラック1、トラック2、、、を表し、ビットが立っているトラックを確保します。

4行目以降の `opentrack` コマンドで、トラックを起動します。1つ目の引数が、トラック番号で、2つ目の引数がそのトラックのシーケンスの先頭ラベルです。指定したトラックは、ラベルの位置から処理が始まります。

なお、トラック0の処理は、最後の `opentrack` コマンドの後、そのまま下の行へと進みます。

### 3.3.3 トラック処理

各トラック毎の処理は、`kart64_title_Track_0:` のようなラベルに続けて、記述しています。

### コード 3-3 トラック処理

```

////////////////////////////////////
; Track 0
////////////////////////////////////

SMF_kart64_title_Track_0:
    notewait_off
; Measure 1 -----
    tempo      140
    prg         0
SMF_kart64_title_Track_0_LoopStart:
    wait       1
    volume     127
    pan        64
    bendrange  2
    pitchbend  0
    mod_speed  16
    mod_depth  0
    wait       767
; Measure 2 -----

```

各シーケンスコマンドの説明は、「5 シーケンスコマンド」を参照してください。

## 3.4 SMFへのテキストコマンド埋め込み

### 3.4.1 概要

シーケンサでシーケンスデータを作成するのは、音の確認をすぐに行えるため便利です。ただし、SMF には制限が多く、テキストシーケンスで使えるようなコマンドの多くが使用できません。コンバート後の\*.smft ファイルを直接編集すれば可能ですが、テキストエディタで編集しても、再度シーケンサで編集すると上書きで消されてしまいます。

そのため、SMF にテキストコマンドを埋め込んでおき、MIDI イベントと一緒にテキストコマンドを、\*.smft ファイルに出力できる機能があります。これを使えば、どんなテキストコマンドを使うことも出来、シーケンサで編集してもテキストコマンドが消えません。

### 3.4.2 埋め込み方

SMF データに、マーカーまたはテキストイベントとして、次のような文字列を埋め込みます。

```
text_03:    pan_r 30, 90
```

この記述により、トラック(MIDI チャンネル)3 番の指定位置に、" pan\_r 30, 90"というコマンドが出力されます。

### 3.4.3 全トラック出力

次のように書くと、全トラックそれぞれに対して出力されます。

```
text_all:    pan_r 30, 90
```

ただし、使用していないトラックには出力されません。

### 3.4.4 ラベル名展開

例えば、全トラックの同じ位置にラベルを定義したい場合、"text\_all"を使って、次のようにしてもエラーとなります。

```
text_all:BLOCK_A:
```

複数の箇所、BLOCK\_A というラベルが定義されてしまい、ラベルの2重定義エラーとなります。このような場合は、次のようにします。

```
text_all:$BLOCK_A:
```

文字'\$'をつけておくことで、ラベル名が展開され、各トラック毎に次のように出力されます。

```
SMF_filename_Track_0_BLOCK_A:
```

"filename"の箇所は SMF のファイル名が、数字の箇所はトラック番号が入ります。トラック番号がトラック毎に異なるため、ラベルの2重定義は起こりません。

また次のように、'\$'の代わりに '\$\$' とすると、

```
text_all:$$BLOCK_A:
```

SMF のファイル名の部分は展開されなくなります。

```
Track_0_BLOCK_A:
```

### 3.4.5 リアルタイムMIDI再生時の注意

リアルタイム MIDI 再生では、埋め込んだテキストシーケンスの処理が行われません。そのため、コンバートして再生したものと異なる結果になりますので、注意してください。

## 4 シーケンスアーカイブ

### 4.1 概要

シーケンスアーカイブは、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。

下記のサンプルを参考に説明していきます。

#### コード 4-1 テキストシーケンスアーカイブ

```

////////////////////////////////////
;      SeqArc for Sample SE
////////////////////////////////////

#include "../sound_data.sddl"

@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

; ノートコマンドだけ
; コイン音
note_only:
    prg 2
    as5 127, 6
    ds6 127, 48
    fin

; ジャンプによるループ（無限の繰り返し）
; 救急車

```

```

jump_seq:
    prg 3
_loop_start:
    bn4 127, 48
    gn4 127, 48
    jump _loop_start

```

## 4.2 サウンドアーカイブラベルファイルの利用

まず、始めの方に記述されている

```
#include "../sound_data.sbd1"
```

について説明します。

これは、一つ上のディレクトリにある"sound\_data.sbd1"というファイル名のサウンドアーカイブラベルファイルを取り込んでいることを表しています。サウンドアーカイブラベルファイルは、サウンドデータのコンパイル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を、このシーケンスアーカイブで使えるようになります。

すなわち、この1行をファイルの始めに記述することにより、後でバンクを指定するときなどにバンクラベルを使ったりすることができるようになります。バンクをラベルではなく番号で指定する場合には、この1行は不要です。

## 4.3 構成

全体的な構成としては、2つの部分に分かれます。

- シーケンステーブル
- シーケンスデータ

### 4.3.1 シーケンステーブル

下記の部分がシーケンステーブルです。

#### コード 4-2 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

```

最初の行は、@SEQ\_TABLE というコマンドで、シーケンステーブルの始まりを表します。

3行目以降が、実際のシーケンステーブルデータになります。1行1行が1つのシーケンスを表します。詳しくは後で説明します。



### 4.3.2 シーケンスデータ

下記の部分がシーケンスデータです。

#### コード 4-3 シーケンスデータ

```
; ノートコマンドだけ
; コイン音
note_only:
    prg 0
    as5 127, 6
    ds6 127, 48
    fin

; ジャンプによるループ（無限の繰り返し）
; 救急車
loop_seq:
    prg 1
_loop_start:
    bn4 127, 48
    gn4 127, 48
    jump _loop_start
```

各シーケンスの始まりは、ラベル名定義で始まり、そのラベルでシーケンスを特定します。

ラベル定義につづけて、シーケンスコマンドを書き連ねて、1つのシーケンスデータを作成します。各シーケンスコマンドについては、「5 シーケンスコマンド」で詳しく説明します。

## 4.4 シーケンステーブル

シーケンステーブルの中身について説明します。

#### コード 4-4 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
```

各行の書式は次のようになっています。

seqName: dataLabel, bankNo, volume, channelPrio, playerPrio, playerNo
---

各要素の意味は、次の通りです。

表 4-1 シーケンステーブル要素

要素	説明
seqName	シーケンスラベル名
dataLabel	データラベル名
bankNo	バンクラベルまたはバンクナンバー
volume	ボリューム
channelPrio	発音プライオリティ
playerPrio	プレイヤープライオリティ
playerNo	プレイヤーラベルまたはプレイヤー番号

以下、順番に各々の構成要素について説明していきます。

#### 4.4.1 シーケンスラベル名

シーケンスラベル名はそのシーケンスを表す識別子です。大文字のアルファベットで始まり、2文字目以降は、大文字アルファベットに加え、アンダースコア”\_”と数字が使えます。同じラベル名を2回使うことはできません。これは、別のシーケンスアーカイブであっても、同じラベル名が存在してはならないことに注意してください。

ここで指定したラベル名は、シーケンスを再生するときに、プログラムで使ったりします。

また、ラベル名ではなくインデックス番号で指定することもできます。

```
SE_COIN = 2:           note_only,      BANK_SE,  65, 96, 64, PLAYER_SE
SE_AMBULANCE:         loop_seq,       BANK_SE,  55, 96, 64, PLAYER_SE
10:                   call_seq,       BANK_SE,  55, 96, 64, PLAYER_SE
```

1行目は、ラベルとインデックス番号の両方を指定するタイプで、ラベル SE\_COIN で、インデックス番号 2 となります。

2行目の様にラベル名のみ指定すると、1つ前の行のインデックス番号に1を足した値が、インデックス番号として設定されます。この例の場合では、インデックス番号は3になります。

3行目は、インデックス番号のみの指定で、ラベル名は指定しません。従って、このシーケンスを指すときにもラベル名ではなく、インデックス番号で指定する必要があります。

#### 4.4.2 データラベル名

データラベル名は、@SEQ\_DATA 以降に記述するシーケンスデータのラベル名を指定します。

#### 4.4.3 バンク

シーケンスをどのバンクを使って再生するかを、ラベルまたは番号で指定します。ただし、サウンドアーカイブラベルファイル(\*.sbd1)を取り込んでいないと、バンクラベルで指定することはできません。

#### 4.4.4 ボリューム

ボリュームは、シーケンス全体のボリュームを調整するパラメータで、0 ～ 127 の範囲で指定します。

---

#### 4.4.5 発音プライオリティ

---

シーケンス全体に対する発音プライオリティを指定します。

発音プライオリティとは、16 チャンネルという制限の中で、どの音を発音するかを決める優先度です。値の範囲は 0 ～127 で、値が大きいほど優先度は高くなります。

発音プライオリティについての詳細は、「サウンドシステムマニュアル」をご覧ください。

---

#### 4.4.6 プレイヤープライオリティ

---

プレイヤープライオリティを指定します。

プレイヤープライオリティとは、シーケンス最大同時再生数の制限の中で、どのシーケンスを再生するかを決める優先度です。値の範囲は 0 ～127 で、値が大きいほど優先度は高くなります。

プレイヤープライオリティについての詳細は、「サウンドシステムマニュアル」をご覧ください。

---

#### 4.4.7 プレイヤー

---

プレイヤー番号は、32 個あるプレイヤーの内、どのプレイヤーを使って再生するかをプレイヤーラベルまたはプレイヤー番号で指定します。ただし、サウンドアーカイブラベルファイル(\*.sbd1)を取り込んでいないと、プレイヤーラベルで指定することはできません。

1つのプレイヤーで再生できるシーケンスの数は限定されていますので、これを考慮して、プレイヤー番号を指定する必要があります。

---

### 4.5 数値表記

---

テキストシーケンスファイル中で、各種数値パラメータを指定する箇所について、直接数値を打ち込む方法の他に、以下のような表記も使えます。

---

#### 4.5.1 2進数、16進数表記

---

通常10進数で記述しますが、2進数または16進数で記述することもできます。

2進数または16進数で記述するときは、それぞれ先頭に、0b または 0x を付けます。たとえば、10進数の12は、次のように記述することができます。

```
0b1100  
0xc
```

---

#### 4.5.2 ビット表記

---

ビットフラグのような、あるビットが1か0かが意味を持つ数値を記述する場合、ビット表記が有効です。

ビット表記では、下位何ビットを1にするかを記述します。例えば、下位1, 3ビットと6ビットから8ビットを1とした値を記述する場合、次のようにします。

```
{ 1, 3, 6-8 }
```

これは、0b111001010 に相当します。最下位ビットは0ビットとなることに注意してください。

### 4.5.3 数式

数値は、数式で記述することもできます。数式の各項には、2進数表記、16進数表記、ビット表記を使うこともできます。

例えば、次のような表記が可能です。

```
2 * 4 + 0x10
( 1 << 4 ) + 3
{ 0, 2 } | { 4-6 }
```

数式で扱える演算子と、演算子の優先順位は、以下の通りです。

**表 4-2 演算子**

優先度	演算子	意味
1	*	乗算
	/	除算
2	+	加算
	-	減算
3	>>	右シフト
	<<	左シフト
4	<	左辺が右辺未満
	<=	左辺が右辺以下
	>	左辺が右辺より大きい
	>=	左辺が右辺以上
5	==	左辺と右辺が等しい
6	&	ビット毎の AND
7		ビット毎の OR

## 5 シーケンスコマンド

シーケンスコマンドは、シーケンスデータ中で使うコマンド群です。

表 5-1 シーケンスコマンド一覧表

コマンド名	説明	規定値	MIDI	参照
cn4 <i>velocity, length</i>	ノートコマンド		\$9x, \$8x	22
wait <i>length</i>	休符コマンド			23
fin	シーケンスの終了			23
prg <i>x</i>	プログラムチェンジ	0	\$Cx	23
tempo <i>x</i>	テンポチェンジ	120		23
volume <i>x</i>	トラックボリュームを変更	127	\$Bx, 7	23
volume2 <i>x</i>	トラックボリュームを変更	127	\$Bx, 11	23
main_volume <i>x</i>	プレイヤーのボリュームを変更	127	\$Bx, 12	24
pitchbend <i>x</i>	ピッチベンドを変更	0	\$Ex	24
bendrange <i>x</i>	ピッチベンドレンジを変更	2	\$Bx, 20	24
transpose <i>x</i>	トランスポーズ	0	\$Bx, 13 <sup>i</sup>	24
pan <i>x</i>	トラックパンを変更	64	\$Bx, 10	24
prio <i>x</i>	トラックの発音プライオリティを変更	64	\$Bx, 14	25
tieon / tieoff	タイモード開始 / 終了	off		25
notewait_on notewait_off	ノートウェイトオン / オフ	on		25
mute	ミュートフラグを変更	0		25
porta <i>x</i>	ポルタメントの開始キー設定及び開始	cn4(60)	\$Bx, 84 <sup>ii</sup>	26
porta_time <i>x</i>	ポルタメントタイムの設定	0	\$Bx, 5	26
porta_on / porta_off	ポルタメントの開始 / 終了	off	\$Bx, 65 <sup>iii</sup>	26
sweep_pitch <i>x</i>	スweepピッチ変化量の設定	0	\$Bx, 28 <sup>iv</sup> \$Bx, 29 <sup>v</sup>	26

<sup>i</sup> トランスポーズ(#13)の 0～127 は、-64～+63 になります。

<sup>ii</sup> ポルタメントコントロール(#84)は、60 が cn4(中央のド)になるノートナンバーで指定します。

<sup>iii</sup> ポルタメント(#65)の 0～63 で porta\_off、64～127 で porta\_on になります。

<sup>iv</sup> スweepピッチ(#28)の 0～127 は、-64～+63 になります。

<sup>v</sup> スweepピッチ(#29)の 0～127 は、値が 24 倍され、-1536～+1512 になります。

<code>alloctrack mask</code>	トラック確保			27
<code>opentrack no, label</code>	トラック起動			27
<code>jump label</code>	シーケンスジャンプ			27
<code>call label</code>	シーケンス呼び出し			27
<code>ret</code>	シーケンス復帰			27
<code>loop_start count</code>	ループ始点		\$Bx, 89	28
<code>loop_end</code>	ループ終点		\$Bx, 90	28
<code>attack x</code>	アタック値の設定		\$Bx, 85	28
<code>decay x</code>	ディケイ値の設定		\$Bx, 86	28
<code>sustain x</code>	サステイン値の設定		\$Bx, 87	28
<code>release x</code>	リリース値の設定		\$Bx, 88	28
<code>envelope a,d,s,r</code>	エンベロープ値の設定			28
<code>mod_depth x</code>	モジュレーションデプスの設定	0	\$Bx, 1	28
<code>mod_range x</code>	モジュレーションレンジの設定	1	\$Bx, 23	28
<code>mod_speed x</code>	モジュレーションスピードの設定	16	\$Bx, 21	28
<code>mod_delay x</code>	モジュレーションディレイの設定	0	\$Bx, 26 \$Bx, 27 <sup>vi</sup>	28
<code>mod_type x</code>	モジュレーションタイプの設定	0	\$Bx, 22	28

## 5.1 ノートコマンド

現在のプログラムナンバを使って、発音を行います。

`cn4 velocity, length`

キーは、`cn4`, `cs4`, `dn4`, `ds4`, `en4`, `fn4`, `fs4`, `gn4`, `gs4`, `as4`, `as4`, `bn4`, `cn5` のような書式で書きます。中央のドは `cn4` になります。範囲は、`cnm1` から `gn9` まで指定できます。(m1 はマイナス1を表します)

`velocity` はベロシティです。0 ~ 127 の範囲で、2乗スケール値として解釈されます。すなわち、127 を 100%とすると、

$$( ( velocity / 127 ) ** 2 ) * 100 [\%]$$

になります。

`length` は音の長さで、48 が4分音符の長さに相当します。実際の長さはテンポに依存します。値の範囲は 0 ~ 268435455 です。ただし、0 の時は特別に無限大を意味します。すなわち、波形データの終わりまで再生することになります。波形データがループしている場合は、永遠に鳴り続けます。(プログラムで停止することはできません。)

ノートウェイトオンの状態(デフォルト)では、音の長さと同じ時間シーケンスが停止します。`length` が 0 の場合は、チャンネルの再生が終わるまで待ちます。つまり、波形データの終わりまで再生した後、次のシーケンス処理に入ります。波

<sup>vi</sup> モジュレーションディレイ(27)は、値を 10 倍にして出力します。

形データがループしている場合は、永遠にシーケンスが停止します。(プログラムで停止することはできます。)ただし、発音プライオリティが低くて、再生が途中で強制的に停止させられた場合でも、次のシーケンス処理に移行しますので、注意してください。

#### コード 5-1 ノートコマンドの例

```
cn4 110, 48
dn4 110, 48
en4 127, 96
```

## 5.2 休符コマンド

---

指定時間シーケンスを停止します。

```
wait length
```

length は音の長さで、48 が4分音符の長さに相当します。実際の長さはテンポに依存します。値の範囲は 0～268435455 です。

## 5.3 シーケンスの終了

---

トラックのシーケンス処理を終了します。

```
fin
```

全トラックのシーケンス処理が終了した時点で、プレイヤーの処理も停止します。

fin コマンドを書き忘れると、下方のシーケンスデータが続けて実行されてしまいます。予期せぬ音が鳴ったり、運が悪いと暴走しますので、十分注意してください。

## 5.4 プログラムチェンジ

---

プログラムチェンジを実行します。

```
prg x
```

プログラムナンバーは、0 ～ 32767 の範囲です。ただし、MIDI からは 127 までしか指定できません。デフォルト値は0です。

## 5.5 テンポチェンジ

---

テンポを変更します。

```
tempo x
```

テンポは、1 ～ 1023 の範囲です。プログラムから設定できるテンポ倍率の値を加味して、最終的なテンポ値になります。デフォルト値は、120です。

## 5.6 トラックボリューム

---

トラックのボリュームを変更します。

```
volume x  
volume2 x
```

ボリューム値は、0 ～ 127 の範囲です。デフォルト値は 127 です。

ベロシティと同様に、2乗スケール値として解釈されます。すなわち、127 を 100%とすると、

$$( ( volume / 127 ) ** 2 ) * 100 [\%]$$

になります。

volume コマンドと volume2 コマンドをどちらから一方だけ使う場合に、機能的な違いはありません。両方同時に指定した場合は、両者の効果は同時に反映されます。この時例えば、volume に 80%、volume2 に 50%を設定していると、 $0.8 \times 0.5 = 0.4$  で、40%のボリュームとなります。

## 5.7 メインボリューム

---

プレイヤーのボリュームを変更します。

```
main_volume x
```

ボリューム値は、0 ～ 127 の範囲です。デフォルト値は 127 です。

ベロシティと同様に、2乗スケール値として解釈されます。すなわち、127 を 100%とすると、

$$( ( volume / 127 ) ** 2 ) * 100 [\%]$$

になります。

## 5.8 ピッチベンド

---

ピッチベンドまたは、ピッチベンドレンジを変更します。

```
pitchbend x  
bendrange x
```

ピッチベンドの値の範囲は、-128 ～ 127 です。デフォルト値は 0 です。

ベンドレンジの単位は半音で、デフォルト値は 2、範囲は 0～127 です。

ピッチベンドの値が MAX の時、ちょうどベンドレンジで指定した高さ分変化します。例えば、ベンドレンジが 2 の場合、ピッチベンドの値が 127 の時+2半音、64 の時+1半音、-64 の時-1 半音変化します。

## 5.9 トランスポーズ

---

トランスポーズを行います。

```
transpose x
```

トランスポーズの値の単位は半音で、-64 ～ +63 の範囲です。デフォルト値は 0 です。

## 5.10 トラックパン

---

トラックのパンを変更します。

```
pan x
```



パン値は、0 ～ 127 の範囲で、0 の時左、127 の時右、64 の時中央になります。デフォルト値は 64 です。

## 5.11 発音プライオリティ

---

トラックの発音プライオリティを設定します。

```
prio x
```

プライオリティ値の範囲は 0 ～ 127 で、デフォルト値は 64 です。

実際には、プレイヤーのプライオリティ値と、このプライオリティとを足し合わせて、最終的な発音プライオリティ値となります。値が大きいほど優先順位が高くなり、値が同じ場合は後から発音する方が優先されます。

## 5.12 タイモード

---

タイモードを変更します。

```
tieon  
tieoff
```

デフォルトはタイモードオフです。

タイモードオン状態でノートオンを行うと、ノートコマンドの音長指定にかかわらず、音が鳴り続けます。続けてノートオンを行うと、発音を続けたまま音程とベロシティだけ変化します。タイモードをオフにするか、シーケンスを停止させると、発音は終了します。

タイモードオンにすると、発音中の音は強制リリースさせられます。また、ノートウェイトモードをオフにしていたとしても、単音でしか鳴らせませんので、注意してください。

## 5.13 ノートウェイトモード

---

ノートウェイトモードを変更します。

```
notewait_on  
notewait_off
```

デフォルトはノートウェイトオンの状態です。

ノートウェイトオンの状態でノートオンを行うと、ノートコマンドの音長と同じ時間だけシーケンスが停止します。オフの状態では、停止せずにそのまま次のコマンドが処理されます。

オンの状態では、音が鳴っている間シーケンスが止まるので、順番に音を鳴らすような時、いちいち休符コマンドを挟む必要が無くて便利ですが、発音中は何も処理が行えません。

一方、オフの状態では、ノートコマンドの後に待ちが入りませんので、同時に2音以上の音を鳴らしたり、発音中にパンを移動させたりすることができます。

## 5.14 ミュート

---

ミュートフラグを変更します。

```
mute x
```

引数に設定できる値の範囲は、0～3 で、デフォルト値は 0 です。数値の意味は、下記の通りです。

表 5-2 ミュートコマンドの引数

引数	説明
0	ミュート解除
1	発音中の音を止めずにミュート
2	発音中の音をリリースしてミュート
3	発音中の音を即座に止めてミュート

mute コマンドは、SDK バージョン 3.1 以降でのみ使用できます。

## 5.15 ポルタメント

ポルタメントの設定を行います。

```
porta key
porta_time time
porta_on
porta_off
```

**porta** コマンドで、ポルタメントモードに入ります。ポルタメントモードでノートコマンドを実行すると、**key** で指定した音程から、ノートコマンドで指定した音程へと、変化しながら再生されます。さらに続けてノートコマンドを実行すると、前のノートコマンドの音程から、今回のノートコマンドの音程へと変化します。

**porta\_time** コマンドで、音程変化の速さを指定します。(MIDI の慣習にあわせて、**porta\_speed** ではなく、**porta\_time** になっています)。値の範囲は **0~255** で、デフォルトは **0** です。値が大きいほど変化が遅く、時間が長くなります。**0** の時は、特別にノート長と同じ長さで変化することを意味します。すなわち、ノートコマンドによる発音の時間と同じ時間をかけて、音程が変化します。

**porta\_on** コマンドも、**porta** コマンド同様、ポルタメントモードに入りますが、開始キーを指定しません。次のノートコマンドは、**porta\_on** コマンドより前に実行されたノートコマンドの音程から変化します。さらに続けてノートコマンドを実行すると、前のノートコマンドの音程から、今回のノートコマンドの音程へと変化します。

**porta\_off** コマンドで、ポルタメントモードが解除されます。

タイモードオンの状態でポルタメントを使うと、ピッチベンドで音程を連続変化させるような表現が可能です。

## 5.16 スイープ

スイープの設定を行います。

```
sweep_pitch pitch
```

**sweep\_pitch** コマンドを使うと、各発音がスイープして鳴るようになります。スイープ設定をした状態で、ノートコマンドを実行すると、ノートコマンドで指定した音程から **pitch** の分ずれた音程で鳴り始め、徐々に正しい音程に変化します。**pitch** の値の範囲は、**-32768~32767** で、デフォルトは **0** です。**64** の時ちょうど1半音ずれます。

正しい音程に戻る速さは、**porta\_time** コマンドで設定した速さと同じになります。

## 5.17 トラック確保

トラックを確保します。

```
alloctrack mask
```

このコマンドは、必ずシーケンスの一番はじめて使う必要があります。

`mask` は、確保するトラックのビットマスクで、下位ビットから順にトラック0、トラック1、トラック2、、、を表し、ビットが立っているトラックを確保します。(トラック0はすでに確保されているので、再下位ビットは無視されます)。また、マクロを使って、次のように記述することもできます。

```
alloctrack TRACK_1 | TRACK_2 | TRACK_3
```

上記の記述により、トラック1、トラック2、トラック3が確保されます。

## 5.18 トラック起動

別のトラックを起動します。

```
opentrack no, label
```

このコマンドは、前もって `alloctrack` で確保したトラックに対してのみ実行できます。

`no` はトラック番号で、0 ~ 15 を指定できますが、現在のトラックを指定することはできません。また、すでに実行中のトラックは一度全ての音をリリースした後、実行されます。

`label` は、シーケンスの先頭を表すラベルです。

なお、現在のトラックより大きい番号のトラックを開いた場合、同じフレームではじめの処理が実行されますが、小さい番号のトラックを開いた場合は、1フレーム遅れて実行されることに注意してください。

## 5.19 シーケンスジャンプ

シーケンス位置を別の場所に飛ばします。

```
jump label
```

`label` は飛び先を表すラベルです。

ジャンプを使うことで、ループするシーケンスを作成することができます。ただし、ループ内に `wait` コマンドや、ノートウエイトオンでのノートコマンドなどの、シーケンスを停止させるコマンドが含まれていないと、永遠にループし続けて暴走してしまいますので、注意してください。

## 5.20 シーケンス呼び出し

シーケンス位置を別の場所に飛ばしますが、あとで元の位置に戻ってくるようにします。

```
call label  
ret
```

`label` は飛び先を表すラベルです。

飛んだ先で、`ret` コマンドを使うことで、元の呼び出し位置に戻ります。飛び先で再び `call` コマンドを実行することもできますが、3階層までしか呼び出せません。また、「5.21ループ」の入れ子階層とあわせて、3階層までとなることに注意

してください。

call/ret コマンドと loop\_start/loop\_end コマンドは、それぞれ入れ子で呼び出す必要があります。たとえば、call コマンドで飛んだ先で、loop\_start コマンドを使った場合、loop\_end コマンドの後でしか ret コマンドを使えません。loop\_start/loop\_end の間では、ret コマンドを使えないことに注意してください。

## 5.21 ループ

回数を指定して、ある区間の処理を繰り返し実行します。

```
loop_start count
loop_end
```

loop\_start でループ開始点を設定します。count はループ回数で、0～255 の範囲で指定できます。0 の時は無限ループになります。

loop\_end でループ終点を設定します。ループ回数が指定した回数に満たない場合は、loop\_start で設定したループ開始点へ戻ります。

ループ区間の中で、さらにループさせることもできますが、3階層までしかできません。また、「5.20シーケンス呼び出し」の入れ子階層とあわせて、3階層までとなることに注意してください。

call/ret コマンドと loop\_start/loop\_end コマンドは、それぞれ入れ子で呼び出す必要があります。たとえば、call コマンドで飛んだ先で、loop\_start コマンドを使った場合、loop\_end コマンドの後でしか ret コマンドを使えません。loop\_start/loop\_end の間では、ret コマンドを使えないことに注意してください。

MIDI からこのコマンドを使用した場合、リアルタイム MIDI 再生時には無視されることに注意してください。

## 5.22 エンベロープ

エンベロープの設定を行います。

```
attack x
decay x
sustain x
release x
envelope a, d, s, r
```

デフォルトでは、バンクのエンベロープ値が使われますが、これらのコマンドを使うことで、エンベロープの値を上書きすることができます。release コマンドなどを単独で用いた場合、他のアタックなどの値は、バンクで設定した値が使われます。envelope コマンドを使うことで、一度に全ての値を設定することができます。

なお、値に-1を設定すると、バンクのエンベロープ値を使用する状態に戻すことができます。

## 5.23 モジュレーション

モジュレーションの設定を行います。

```
mod_depth depth
mod_range range
mod_speed speed
mod_delay delay
mod_type type
```

mod\_depth でモジュレーションの深さを設定します。値の範囲は 0 ～ 127 で、デフォルト値は 0 です。最大値での

変化の度合いは、下の通りです。(ただし、後述のmod\_rangeが1の場合)

**表 5-3 モジュレーション変化の度合い**

タイプ	変化の度合い
音程変化	±1 半音の間で変化します。
音量変化	±6.0dB の間で変化します。
定位変化	左 MAX ～ 右 MAX の間で変化します。

mod\_range でモジュレーションの最大変化幅を設定します。値の範囲は、0～127 で、デフォルト値は 1 です。例えば音程変化の場合、デフォルトでは mod\_depth が最大の時、±1 半音の間で変化しますが、mod\_range を 12 にすると、±12 半音すなわち1オクターブの幅で、モジュレーションがかかります。

mod\_speed でモジュレーションの速さを設定します。値の範囲は 0 ～ 127 で、デフォルト値は 16 です。1 の時約 0.4Hz で線形的に変化しますので、0.0Hz ～ 約 50Hz の範囲で設定できます。

mod\_delay でモジュレーションの遅延時間を設定します。値の範囲は、0 ～ 32767 で、デフォルト値は、0 です。単位は1サウンドフレーム(約 5ms)で、テンポには依存しません。MIDI からは、コントロールチェンジ 26 または、27 を使って設定します。コントロールチェンジ 26 の方は、値をそのまま出力しますが、27 の方は値を 10 倍して出力します。結果、0 ～ 1270 の範囲で設定できます。

mod\_type はモジュレーションのタイプを設定します。設定できる値は、下のとおりです。デフォルトは 0=ビブラート(音程変化)です。

**表 5-4 モジュレーションタイプ一覧**

数値	ラベル	説明
0	MOD_TYPE_PITCH	ビブラート(音程変化)
1	MOD_TYPE_VOLUME	トレモロ(音量変化)
2	MOD_TYPE_PAN	パン(定位変化)

限界以上の振幅は頭打ちになって変化しません。たとえば、音量変化させようとしても、元の音量が最大だった場合、音が大きくなる方の変化はせず、小さくなる方だけ変化することに注意してください。

## 6 拡張シーケンスコマンド

### 6.1 概要

拡張シーケンスコマンドとは、上で説明したシーケンスコマンドの機能に対し、より自由度の高い設定を行えるようにしたコマンド群です。

通常のシーケンス再生には、普通のシーケンスコマンドを使うだけで十分ですが、拡張シーケンスコマンドを使うことによって、より凝った演出や、インタラクティブなシーケンス再生などが行えるようになります。

### 6.2 ランダムコマンド

幾つかのシーケンスコマンドでは、値に乱数を設定できます。

```
pitchbend_r -12 , 12
```

上記の例では、-12～+12 の間の範囲の乱数値で、pitchbend コマンドを実行します。これにより、鳴らす毎に微妙に音程を変えて再生することができます。

#### 6.2.1 書式

ランダムコマンドの基本的な書式は次の通りです。

```
(command)_r min, max
```

最小値 min の値から、最大値 max の値までの範囲で乱数が生成され、その値を command の引数として実行します。

#### 6.2.2 ランダムコマンド一覧

表 6-1 ランダムコマンド一覧表

wait_r	prg_r	volume_r	volume2_r
main_volume_r	pitchbend_r	pan_r	transpose_r
mute_r	porta_time_r	sweep_pitch_r	mod_depth_r
mod_speed_r	attack_r	decay_r	sustain_r
release_r	mod_delay_r	loop_start_r	setvar_r
addvar_r	subvar_r	mulvar_r	divvar_r
shiftvar_r	randvar_r	cmp_eq_r	cmp_ge_r
cmp_gt_r	cmp_le_r	cmp_lt_r	cmp_ne_r

表中 setvar\_r 以降のコマンドについては、後で説明します。

普通のシーケンスコマンドとの対応表は、「7.1全シーケンスコマンド一覧表」をご覧ください。

### 6.2.3 ランダム音長ノートコマンド

ノートコマンドに対し、"\_r"をつけてランダムコマンドとすることもできます。この時、音長がランダムに設定されることになります。

```
cn4_r velocity, min, max
```

音程をランダムに変化させたい場合は、`transpose_r` で代用してください。また、ベロシティをランダムに変化させたい場合は、`volume_r` や `volume2_r` など代用してください。

## 6.3 変数コマンド

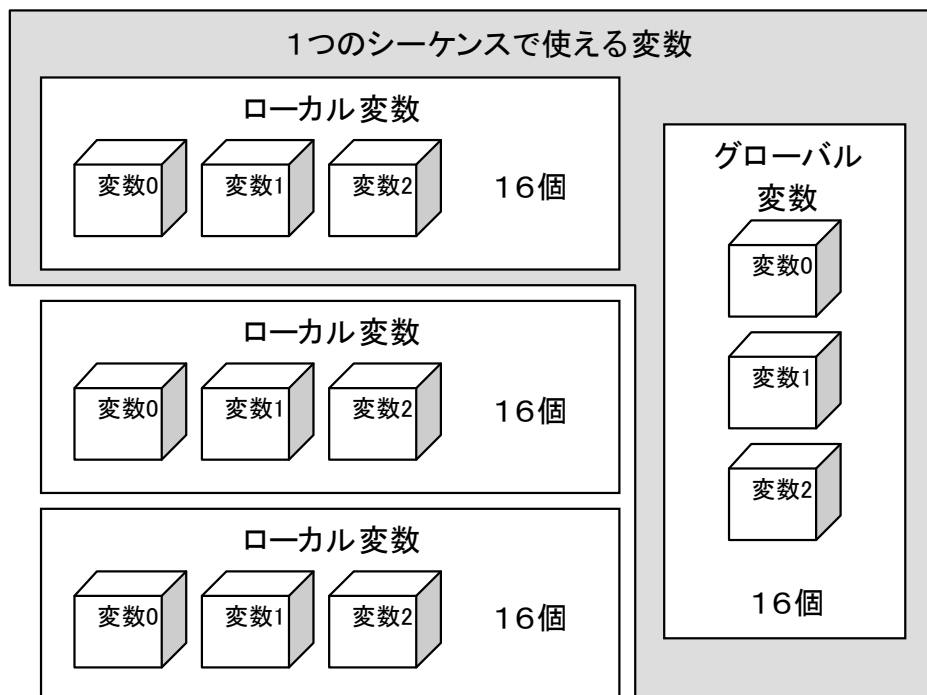
シーケンス中で変数を扱うことができます。

### 6.3.1 変数とは？

変数とは、数値を保存しておけるメモリのことです。1つの変数には、-32768～32767 の範囲の値を格納できます。

1つのシーケンス内で自由に使える変数が16個ずつあり、システム全体で共有して使用する変数が16個あります。シーケンス内のみで扱う変数のことを、「ローカル変数」。システム全体で共有する変数を「グローバル変数」と呼びます。

図 6-1 ローカル変数とグローバル変数



シーケンス中では、各変数を番号で指定します。0から15の値を指定すると、ローカル変数16個の内の1つを表します。16から31の値を指定すると、グローバル変数16個の内の1つを表します。ただし、変数番号としては16を引いた値になります。すなわち、16から31で、グローバル変数の0から15を表します。

変数の初期値は、-1 です。グローバル変数はシステム起動時に、ローカル変数はシーケンス開始時に初期化されます。

### 6.3.2 変数演算コマンド

それぞれの変数に対して、値のセット、四則演算などを行います。

表 6-2 変数演算コマンド一覧

コマンド名	説明
setvar <i>varNo</i> , <i>x</i>	変数に <i>x</i> をセット
addvar <i>varNo</i> , <i>x</i>	変数の値に、 <i>x</i> を加算
subvar <i>varNo</i> , <i>x</i>	変数の値から、 <i>x</i> を引く
mulvar <i>varNo</i> , <i>x</i>	変数の値に、 <i>x</i> をかける
divvar <i>varNo</i> , <i>x</i>	変数の値を、 <i>x</i> で割る
shiftvar <i>varNo</i> , <i>x</i>	変数の値を、 <i>x</i> ビット左ヘシフト(負値なら右ヘシフト)
randvar <i>varNo</i> , <i>x</i>	変数に、0～ <i>x</i> の範囲の乱数値をセット(負値なら、 <i>x</i> ～0の範囲)

### 6.3.3 変数コマンド

上記の変数演算コマンドでセットした変数の値を使って、シーケンスコマンドを実行することができます。

```
pitchbend_v 0
```

上記の例では、ローカル変数 0 の値を使って、pitchbend コマンドを実行します。

### 6.3.4 書式

変数コマンドの基本的な書式は、次の通りです。

```
(command)_v varNo
```

varNo で指定した変数の値を使って、command を実行します。

なお、各シーケンスコマンドの引数には値の範囲が設定されていますが、その範囲を超えた値を変数に設定していた場合の動作は保証できませんので、ご注意ください。

### 6.3.5 変数コマンド一覧

ランダムコマンドとして使えるシーケンスコマンドは、全て変数コマンドとしても使えます。

表 6-3 変数コマンド一覧表

wait_v	prg_v	volume_v	volume2_v
main_volume_v	pitchbend_v	pan_v	transpose_v
mute_v	porta_time_v	sweep_pitch_v	mod_depth_v
mod_speed_v	attack_v	decay_v	sustain_v
release_v	mod_delay_v	loop_start_v	setvar_v



addvar_v	subvar_v	mulvar_v	divvar_v
shiftvar_v	randvar_v	cmp_eq_v	cmp_ge_v
cmp_gt_v	cmp_le_v	cmp_lt_v	cmp_ne_v

表中の cmp\_ で始まるコマンドは、比較コマンドと言い、後で説明します。

普通のシーケンスコマンドとの対応表は、「7.1全シーケンスコマンド一覧表」をご覧ください。

### 6.3.6 音長変数ノートコマンド

ノートコマンドに対し、"\_v"をつけて変数コマンドとすることもできます。この時、音長の指定を変数で行うことになります。

```
cn4_v velocity varNo
```

音程を変数指定したい場合は、transpose\_v で代用してください。また、ベロシティを変数指定したい場合は、volume\_v や volume2\_v など代用してください。

### 6.3.7 変数間演算コマンド

上記の変数演算コマンドは、ある変数に対し、ある数値を足したり引いたりするコマンドですが、変数コマンドの setvar\_v などを使用することにより、変数間の代入や、ある変数に別の変数の値を足したりすることができます。

#### コード 6-1 変数間演算の例

```
setvar_v 0, 1 ; 変数0に変数1の値を代入
addvar_v 2, 3 ; 変数2に変数3の値を足す
```

## 6.4 条件コマンド

変数を使って、あるシーケンスコマンドを実行する／実行しないといったことを制御することができます。

### 6.4.1 比較コマンド

条件コマンドを使うためには、その前に比較コマンドを実行します。比較コマンドで2つの値を比較した結果に基づいて、条件コマンドを実行するかしないかを決めることになります。

比較コマンドには以下のものがあります。

表 6-4 比較コマンド一覧

コマンド名	数式	説明
cmp_eq varNo, x	(変数) == x	変数の値と x が等しければ真(equal の略)
cmp_ge varNo, x	(変数) >= x	変数の値が x より大きいか、等しければ真(greater equal の略)
cmp_gt varNo, x	(変数) > x	変数の値が x より大きければ真(greater than の略)
cmp_le varNo, x	(変数) <= x	変数の値が x より小さいか、等しければ真(less equal の略)
cmp_lt varNo, x	(変数) < x	変数の値が x より小さければ真(less than の略)

<code>cmp_ne varNo, x</code>	(変数) <code>!= x</code>	変数の値と <code>x</code> が等しくなければ真 (not equal の略)
------------------------------	------------------------	---

比較の結果、真となれば、以降の条件コマンドは実行され、偽となれば、以降の条件コマンドは実行されません。

### 6.4.2 条件コマンド

全てのシーケンスコマンドに対し、条件コマンドが存在します。条件コマンドは、元のコマンド名の最後に"\_if"をつけます。

```
pitchbend_if +48
```

次の例では、乱数に基づいてある音を鳴らしたり、鳴らさなかったりします。

#### コード 6-2 条件コマンドの例

```
randvar 0, 100 ; 0~100の乱数値セット
cmp_le 0, 80 ; 乱数値が80以下なら真
cn4 96, 12
dn4_if 96, 12 ; 真の時のみ鳴る
en4 96, 12
fin
```

## 6.5 プログラムとの通信

変数を使って、プログラムと通信を行うことができます。変数は、プログラムから読み取り及び書き込みができますので、これを使って情報のやりとりができます。

### 6.5.1 MIDIシーケンスとの通信

変数を使って、MIDI シーケンスからの情報をプログラムに通知することができます。

コントロールチェンジ 16～19 の4つは、それぞれ `setvar` コマンドに変換されます。コントロールチェンジ 16 はローカル変数0番にセットします。以下、17、18、19 はそれぞれローカル変数1、2、3 番にセットします。

例えば、曲の構成ブロックの切れ目ごとに変数をセットすると、プログラム側で、現在どこの構成ブロックを再生中なのかを知ることができるようになります。

### 6.5.2 条件コマンドとの組み合わせ

条件コマンドとプログラムからの変数書き込みを組み合わせると、プログラムからのタイミングで、シーケンスを変化させることができます。

#### コード 6-3 条件コマンドとの組み合わせ例

```
_loop:
  cmp_eq 0, 1
  jump_if _loop2 ; 変数0が1の時のみジャンプ
  cn4 96, 12
  dn4 96, 12
  jump _loop
_loop2:
  cn4 96, 12
  en4 96, 12
  jump _loop2
```

上記の例では、プログラム側で何もしなければ、ド・レ・ド・レと鳴り続けますが、プログラムでローカル変数0番に1を書き

込むことによって、ド・ミ・ド・ミと鳴るように変化します。

## 6.6 変数のデバッグ

複雑な変数処理を扱うと、処理のミスにより、思った通りの動作をしない場合があります。そのようなときに、想定通りの値が変数に入っているかどうかを確認するためのシーケンスコマンドが用意されています。

表 6-5 変数デバッグコマンド一覧

コマンド名	説明
printvar <i>varNo</i>	変数の値をデバッグ出力

このコマンドが処理されると、次のような1行がデバッグ出力されます。

```
#0[1]: printvar No.0 = 1
```

printvar の後の意味は、変数番号 0 の値が 1 であることを表しています。

"#0[1]"の部分は、プレイヤー識別番号 0、トラック番号 1 であることを表しています。プレイヤー識別番号は、数値自身には重要な意味はありません。ただ、プレイヤー識別番号が異なれば、それは違うシーケンスからの出力であることがわかるだけです。トラック番号は、あるシーケンス中のどのトラックから出力されたかを表しています。

このシーケンスコマンドは、NITRO-Player または SoundPlayer 上でシーケンスを再生する場合にのみ有効です。それ以外の方法で再生した場合には、このシーケンスコマンドは無視されます。

プリント文は、MCS サーバもしくは、IS-NITRO-DEBUGGER の出力ウィンドウへ出力されます。

## 7 付録

### 7.1 全シーケンスコマンド一覧表

表 7-1 全シーケンスコマンド一覧表

コマンド	説明	規定値	範囲	変数 vii	参照
cn4 <i>velocity, length</i>	ノートコマンド			○	22
wait <i>x</i>	ウェイトコマンド			○	23
fin	シーケンスの終了				23
prg <i>x</i>	プログラムチェンジ	0	0～32767	○	23
tempo <i>x</i>	テンポチェンジ	120	1～1023		23
volume <i>x</i>	トラックボリューム	127	0～127	○	23
volume2 <i>x</i>	トラックボリューム(エクスプレッション)	127	0～127	○	23
main_volume <i>x</i>	メインボリューム	127	0～127	○	24
pitchbend <i>x</i>	ピッチベンド	0	-128～127	○	24
bendrange <i>x</i>	ピッチベンドレンジ	2	0～127		24
transpose <i>x</i>	トランスポーズ	0	-64～63	○	24
pan <i>x</i>	トラックパン	64	0～127	○	24
prio <i>x</i>	トラック発音プライオリティ	64	0～127		25
tieon / tieoff	タイモードオン／オフ	off			25
notewait_on notewait_off	ノートウェイトオン／オフ	on			25
mute <i>x</i>	ミュート	0	0～3	○	25
porta <i>x</i>	ポルタメント開始キー設定及びポルタメントオン	cn4(60)	0～127		26
porta_time <i>x</i>	ポルタメントタイム	0	0～255	○	26
porta_on / porta_off	ポルタメントオン／オフ	off			26
sweep_pitch <i>x</i>	スイープピッチ	0	-32768 ～ 32767	○	26
alloctrack <i>mask</i>	トラック確保				27
opentrack <i>no, label</i>	トラック起動				27
jump <i>label</i>	シーケンスジャンプ				27
call <i>label</i>	シーケンス呼び出し				27
ret	シーケンス復帰				27
loop_start <i>x</i>	ループ開始		0～255	○	28
loop_end	ループ終点				28
attack <i>x</i>	エンベロープのアタック		-1～127	○	28
decay <i>x</i>	エンベロープのディケイ		-1～127	○	28

vii 変数コマンド及びランダムコマンドとしても使えるかどうかを示します。

sustain <i>x</i>	エンベロープのサステイン		-1～127	○	28
release <i>x</i>	エンベロープのリリース		-1～127	○	28
envelope <i>a,d,s,r</i>	エンベロープのADSR		-1～127		28
mod_depth <i>x</i>	モジュレーションデプス	0	0～127	○	28
mod_range <i>x</i>	モジュレーションレンジ	1	0～127		28
mod_speed <i>x</i>	モジュレーションスピード	16	0～127	○	28
mod_delay <i>x</i>	モジュレーションディレイ	0	0～32767	○	28
mod_type <i>x</i>	モジュレーションタイプ	0	0～2		28
setvar <i>no, x</i>	変数代入			○	32
addvar <i>no, x</i>	変数加算			○	32
subvar <i>no, x</i>	変数減算			○	32
mulvar <i>no, x</i>	変数乗算			○	32
divvar <i>no, x</i>	変数除算			○	32
shiftvar <i>no, x</i>	変数シフト			○	32
randvar <i>no, x</i>	乱数値代入			○	32
printvar <i>no</i>	変数デバッグ出力				35
cmp_eq <i>no, x</i>	比較(==)			○	33
cmp_ge <i>no, x</i>	比較(>=)			○	33
cmp_gt <i>no, x</i>	比較(>)			○	33
cmp_le <i>no, x</i>	比較(<=)			○	33
cmp_lt <i>no, x</i>	比較(<)			○	33
cmp_ne <i>no, x</i>	比較(!=)			○	33

## 7.2 MIDIコントロールコード対応表

表 7-2 MIDIコントロールコード対応表

10 進数	16 進数	コマンド	参照	10 進数	16 進数	コマンド	参照
0	00h			64	40h		
1	01h	mod_depth	28	65	41h	porta_on / porta_off	25
2	02h			66	42h		
3	03h			67	43h		
4	04h			68	44h		
5	05h	porta_time	25	69	45h		
6	06h	(Data Entry)	39	70	46h		
7	07h	volume	23	71	47h		
8	08h			72	48h		
9	09h			73	49h		
10	0Ah	pan	24	74	4Ah		
11	0Bh	volume2	23	75	4Bh		
12	0Ch	main_volume	24	76	4Ch		
13	0Dh	transpose	24	77	4Dh		
14	0Eh	prio	25	78	4Eh		
15	0Fh			79	4Fh		
16	10h	setvar 0	32	80	50h		
17	11h	setvar 1	32	81	51h		
18	12h	setvar 2	32	82	52h		
19	13h	setvar 3	32	83	53h		
20	14h	bendrange	24	84	54h	porta	25
21	15h	mod_speed	28	85	55h	attack	28
22	16h	mod_type	28	86	56h	decay	28
23	17h	mod_range	28	87	57h	sustain	28
24	18h			88	58h	release	28
25	19h			89	59h	loop_start	28
26	1Ah	mod_delay	28	90	5Ah	loop_end	28
27	1Bh	mod_delay ( x 10 )	28	91	5Bh		
28	1Ch	sweep_pitch	26	92	5Ch		
29	1Dh	sweep_pitch ( x 24 )	26	93	5Dh		
30	1Eh			94	5Eh		
31	1Fh			95	5Fh		
32	20h			96	60h		
33	21h			97	61h		
34	22h			98	62h		
35	23h			99	63h		
36	24h			100	64h	( RPN LSB )	39
37	25h			101	65h	( RPN MSB )	39
38	26h			102	66h		
39	27h			103	67h		
40	28h			104	68h		
41	29h			105	69h		
42	2Ah			106	6Ah		
43	2Bh			107	6Bh		
44	2Ch			108	6Ch		
45	2Dh			109	6Dh		
46	2Eh			110	6Eh		
47	2Fh			111	6Fh		
48	30h			112	70h		
49	31h			113	71h		
50	32h			114	72h		
51	33h			115	73h		
52	34h			116	74h		
53	35h			117	75h		
54	36h			118	76h		
55	37h			119	77h		
56	38h			120	78h		
57	39h			121	79h		
58	3Ah			122	7Ah		
59	3Bh			123	7Bh		
60	3Ch			124	7Ch		
61	3Dh			125	7Dh		
62	3Eh			126	7Eh		
63	3Fh			127	7Fh		

- #13 tranpose は、値から 64 を引きます。
- #16～#19 はそれぞれ、ローカル変数 0～3 に値をセットします。
- #28、#29 sweep\_pitch は、値から 64 を引きます。(＃29 sweep\_pitch はその後24倍します)
- #65 porta\_on / porta\_off は、値が 64 以上の時 porta\_on、64 未満の時 porta\_off になります。
- #120～#127 はチャンネルモードメッセージで、コントロールチェンジではありません。

### 7.2.1 RPN対応表

MSB	LSB	コマンド	参照
00h	00h	bendrange	24
7Fh	7Fh		

## 7.3 MusicPlayer2000 用SMFとの互換性

---

AGB サウンド開発ツール MusicPlayer2000(以下 MP2000)用に作成した SMF との相違点を挙げます。ただし、これで全てではないことにご注意ください。

### 7.3.1 トラック番号

---

MP2000 では、若い方から順にトラック番号が割り付けられていましたが、NITRO-Composer では、シーケンスデータの番号がそのままつけられ、欠番が自動的に詰められなくなりました。

### 7.3.2 モジュレーション

---

モジュレーションの値の解釈が MP2000 と異なりますので、値の調整が必要になります。

### 7.3.3 チューニング

---

MIDI コントロールチェンジ 24 のチューニング機能は無くなりました。

### 7.3.4 擬似エコー

---

MIDI コントロールチェンジ 30,31 の擬似エコー機能は無くなりました。

### 7.3.5 プライオリティ

---

プライオリティの MIDI コントロールコードが、33 から 14 に変更になりました。

### 7.3.6 トラック単位のループ

---

MP2000 では、ループ始点が MIDI コントロールコード 30 で値が 100、ループ終点が MIDI コントロールコード 30 で値が 101 でした。

NITRO-Composer では、ループ始点が MIDI コントロールコード 89 で値が 0、ループ終点が MIDI コントロールコード 90(値は任意)です。

### 7.3.7 音量計算

---

音量計算の方法が、MP2000 と若干異なりますので、音量の再調整が必要かも知れません。

### 7.3.8 効果音など作成時のテンポ

---

MP2000 では、テンポを 150 にすると、処理基準時間にぴったり同期しましたが、NITRO-Composer では、240 または 120 の時に、ぴったり同期します。

### 7.3.9 4分音符分解能

---

MP2000 では、タイムベース 24 でしたが、48 になりました。



© 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。