

# NITRO-Composer Sound Designer Guide

Version 1.4.0

**The contents in this document are highly  
confidential and should be handled accordingly.**

**Confidential**

These coded instructions, statements, and computer programs contain proprietary information of Nintendo of America Inc. and/or Nintendo Company Ltd. and are protected by Federal copyright law. They may not be disclosed to third parties or copied or duplicated in any form, in whole or in part, without the prior written consent of Nintendo.

# Contents

---

1	Introduction .....	9
2	Overview of Sound Data .....	10
2.1	The Structure of Sound Data .....	10
2.1.1	Waveform Data .....	10
2.1.2	Bank Data .....	10
2.1.3	Sequence Data.....	10
2.1.4	Stream Data .....	10
2.2	Sound Archives .....	10
2.3	Flow of data creation .....	11
2.3.1	Flow of Data Creation.....	11
2.3.2	Procedure for Creating Data .....	11
3	Sound Archive .....	13
3.1	Sound Archive Definition File.....	13
3.2	@PATH.....	15
4	Sequence Data .....	16
4.1	Sequence Data Registration.....	16
4.1.1	Specifying Banks.....	16
4.1.2	Specifying the Player .....	17
4.1.3	Other Parameters.....	17
4.1.4	About Labels .....	17
4.2	Creating Sequence Data .....	18
5	Sequence Archives .....	19
5.1	What is a Sequence Archive?.....	19
5.1.1	Characteristics of Normal Sequence Files .....	19
5.1.2	Characteristics of Sequence Archives.....	20
5.2	Sequence Archive Registration .....	20
5.3	Creating Sequence Archives .....	21
5.3.1	Sequence Tables.....	22
5.3.2	Registering Sequence Data .....	22
5.3.3	Creating Sequence Data .....	23
5.3.3.1	Example of Simple Waveform Playback.....	23
5.3.3.2	Example of Simple Sequence Playback .....	23
5.3.4	Pitch Notation .....	25
6	Bank Data .....	26

6.1	Registering Bank Data.....	26
6.1.1	Association with Waveform Archives.....	26
6.1.2	Example of Registering Bank Data.....	27
6.1.3	Sound Data Interrelationships.....	29
6.1.4	Creating Bank Data.....	30
6.1.5	Original Key.....	30
7	Memory Management.....	31
7.1	Two Types of Heaps.....	31
7.1.1	Sound Heap.....	31
7.1.2	Player Heap.....	31
7.2	Creating Groups.....	32
7.3	Loading Groups.....	33
7.4	Creating Player Heaps.....	33
7.5	Loading to Player Heap.....	34
7.6	Deciding the Player Heap Size.....	34
7.6.1	Note on Size Determination.....	34
8	Stream Playback.....	35
8.1	Registering Stream Data.....	35
8.1.1	Specifying a Player Number.....	35
8.1.2	Other Parameters.....	35
8.2	Stream Player Definitions.....	36
8.3	Creating Stream Data.....	36
9	SoundPlayer.....	37
9.1	How to Use SoundPlayer.....	37
9.1.1	MakeSound.bat.....	38
9.2	Sound Playback.....	38
9.3	Selecting the Sound Type.....	39
9.4	Screen Displays.....	39
9.4.1	Displaying Label Names.....	39
9.4.2	Displaying Parameters.....	39
9.4.3	Sequence Archive Number.....	40
9.5	Channel Meter.....	40
9.6	Real-time MIDI playback.....	41
9.6.1	What is Real-Time MIDI Playback?.....	41
9.6.2	Preparation.....	41
9.6.3	Bank Selection.....	42
9.6.4	MIDI Signal Transmission.....	42
9.7	Manual Load Mode.....	42

9.7.1	What is Manual Load Mode?	42
9.7.2	Operation in Manual Load Mode	42
9.7.3	Mode Switching	43
9.7.4	Loading Data	43
9.7.5	Memory Display	43
9.8	Key Operations	44
9.9	Error Messages	45
10	Error Handling	46
10.1	Error Generation	46
10.2	Determine the Cause of the Error	46
10.3	Error Handling	46
10.4	Confirm that an Error Has Been Fixed	47
11	The Next Step	48

## Codes

Code 3-1	The Sound Archive Definition File	13
Code 3-2	Registering Sequence Data	14
Code 4-1	The Sequence Data Section	16
Code 4-2	Adding Sequences to the Sequence Data Section	16
Code 5-2	The Sequence Archive Section	20
Code 5-3	Text Sequence Archive	21
Code 5-4	Sequence table	22
Code 6-1	The Bank Data Section	26
Code 6-2	The Waveform Archive Section	26
Code 6-3	Example of bank data registration	27
Code 6-4	Example of bank data registration (2)	28
Code 6-5	Bank Definition File	30
Code 7-1	Group Information Section	32
Code 7-2	Player Information Section	33
Code 8-1	Stream Data Section	35
Code 8-2	Stream Player Information Section	36

## Tables

Table 5-1	Natural / Sharp / Flat	25
Table 9-1	Key Operations in SoundPlayer	44
Table 9-2	SoundPlayer Error Messages	45

## Figures

---

Figure 2-1 Flow of data creation.....	11
Figure 5-1 Conceptual Diagram of Sequence Archive.....	19
Figure 5-2 Pitch Notation.....	25
Figure 6-1 Relationships Among Various Types of Sound Data .....	29
Figure 7-1 Relationship between Group Settings and Sound Data .....	32
Figure 7-2 Relationship between Group Settings and Sound Data .....	33
Figure 9-1 SoundPlayer screens.....	37
Figure 9-2 SoundPlayer Top Screen (Sequence Archive number) .....	38
Figure 9-3 SoundPlayer Upper Screen (parameter information) .....	39
Figure 9-4 The SoundPlayer Channel Meter.....	40
Figure 9-5 The SoundPlayer Channel Meter (Red Display) .....	41
Figure 9-6 SoundPlayer display (real-time MIDI).....	42
Figure 9-7 SoundPlayer Screen (Manual Load Mode) .....	43

## Revision History

Version	Revision Date	Description
1.4.0	10/28/2005	<ul style="list-style-type: none"> <li>Figure 5-2 missing. Just reporting it</li> <li>Updated headings and title to improve consistency</li> <li>General editing to help improve clarity</li> <li>Updated tables, figures, and TOC</li> <li>Updated or inserted cross references</li> </ul>
1.4.0	01/31/2005	<ul style="list-style-type: none"> <li>Added a description associated with the elimination of distinction between uppercase letters and lowercase letters in pitch notation</li> <li>Added a description related to NITRO-Player</li> <li>Added a description related to the SoundPlayer channel meter</li> <li>Revised according to changes to <code>se.mus</code> in the sample data</li> <li>Changed "NITRO" to "Nintendo DS"</li> </ul>
1.3.0	10/12/2004	<ul style="list-style-type: none"> <li>Revised to reflect change that allows the use of labels to specify the player.</li> <li>Revised to reflect the addition of <code>IS-AGB-MIDI</code> support for the SoundPlayer.</li> </ul>
1.2.1	09/16/2004	<ul style="list-style-type: none"> <li>Corrected spelling mistakes in SoundPlayer error messages.</li> <li>Unified the <code>*.sadl</code> file names to "sound label files."</li> </ul>
1.2.0	09/02/2004	<ul style="list-style-type: none"> <li>Added the description for the real-time MIDI features of the SoundPlayer.</li> </ul>
1.1.0	08/10/2004	<ul style="list-style-type: none"> <li>Added description of streams.</li> </ul>
1.0.0	07/20/2004	<ul style="list-style-type: none"> <li>Changed the overall organization.</li> <li>Changed <code>SoundPlayer.bin</code> to <code>SoundPlayer.srl</code>.</li> <li>Added sequence type selection feature in SoundPlayer.</li> <li>Added SoundPlayer error messages.</li> <li>Deleted waveform archive for SE in sample data.</li> <li>Revision involving the increased degree of freedom for making association with banks and waveform archives.</li> <li>Moved the explanation of sound map files to "Sound Archive Manual".</li> </ul>
0.5.0	06/01/2004	<ul style="list-style-type: none"> <li>Fix associated with file system support.</li> <li>Fix associated with changes to SoundPlayer development environment.</li> <li>Added description of sound map file.</li> <li>Added conceptual diagram of sequence archive.</li> <li>Changed name of "registration block" in sound archive to "section."</li> <li>Made minor changes to diagram showing relationships of the various sound data.</li> </ul>
0.4.0	04/12/2004	<ul style="list-style-type: none"> <li>Added error handling.</li> </ul>
0.3.0	04/01/2004	<ul style="list-style-type: none"> <li>Made corrections in line with changes to use of SoundPlayer.</li> <li>Added explanation regarding sound label lists.</li> <li>Revised part about flats in pitch notation.</li> <li>Supplemented explanation about Sequence Archive.</li> </ul>

		<ul style="list-style-type: none"><li>• Supplemented explanation about managing the loading of sequence data.</li></ul>
0.2.0	03/18/2004	Made corrections in line with changes to sequence format.
0.1.0	03/01/2004	First version.



# 1 Introduction

This document is for sound data designers and explains how to create sound data. This document provides an overview of sound data and specific explanations using example data. How to use the SoundPlayer tool to check sounds on the Nintendo DS is also addressed.

## 2 Overview of Sound Data

### 2.1 The Structure of Sound Data

---

NITRO-Composer accepts four types of sound data:

- Waveform data
- Bank data
- Sequence data
- Stream data

The following is a simple explanation of these sound data.

#### 2.1.1 Waveform Data

---

Mono waveform files in AIFF and WAV format are accepted. Use multiple sets of waveform data in a waveform archive with bank data will be described in 6.1.1 - Association with Waveform Archives.

#### 2.1.2 Bank Data

---

Bank data corresponds to sound source data. Waveform files are assigned to each program number to set up the sound source. Bank data are text files.

#### 2.1.3 Sequence Data

---

NITRO-Composer accepts Standard MIDI File (SMF) format 0 or format 1. NITRO-Composer also accepts sequence archives, a text file that declares multiple sequences.

#### 2.1.4 Stream Data

---

Stream data is waveform data used to stream playback. AIFF or mono or stereo WAV format is accepted.

### 2.2 Sound Archives

---

NITRO-Composer handles all sound data sets in one sound archive, rather than handling each sound data set separately. Using sound archives simplifies data management and uses memory more efficiently.

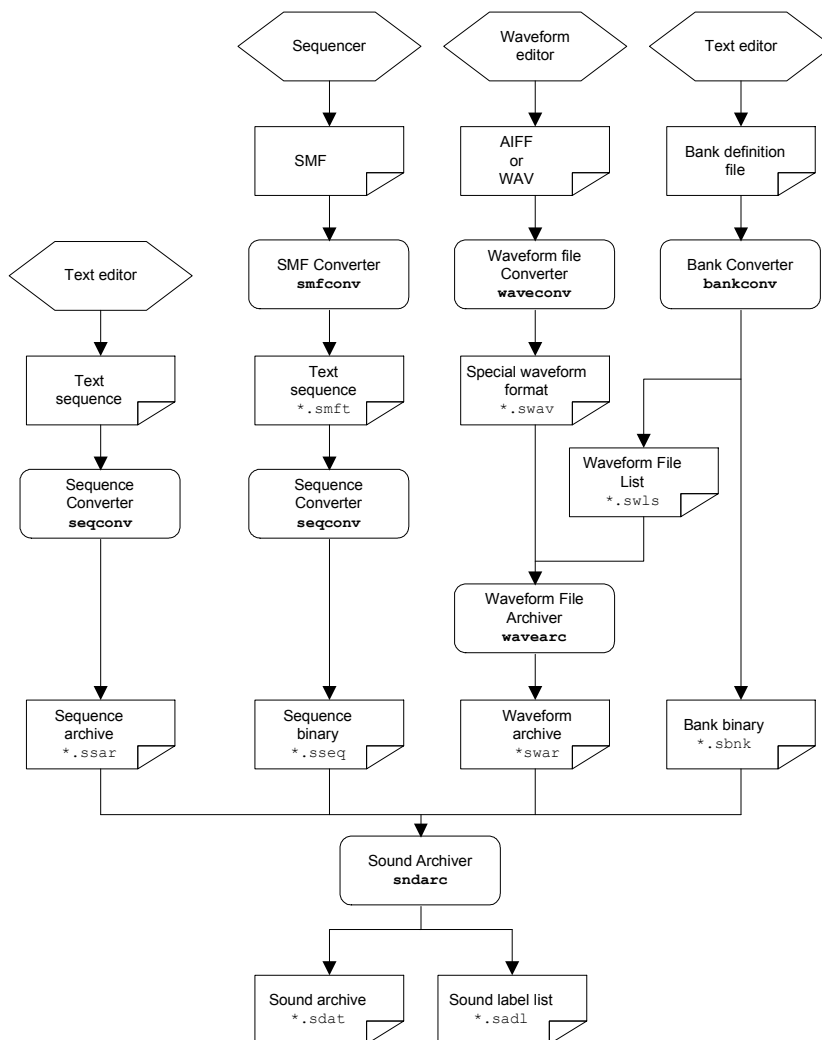
Ultimately, the sound designer is responsible for creating the sound archive file.

## 2.3 Flow of data creation

### 2.3.1 Flow of Data Creation

Figure 2-1 shows the flow of data creation.

Figure 2-1 Flow of data creation



### 2.3.2 Procedure for Creating Data

As shown in Figure 2-1, sound data creation takes many steps in the process. Each operation is explained in later chapters using samples, but understanding each operation to go through the process is unnecessary.

Using the tool `sound archiver sndarc`, generate the final output sound archive and executes all other conversion processes at the same time. This tool compares timestamps and converts only necessary files. Therefore, the conversion process does not take much time.

The sound designer gives the final output Sound Archive (\*.sdat) and Sound Label File (\*.sagl) to the sound programmer. The sound archive contains all of the sound data, and is stored in ROM to play sounds. The sound label file includes the sequence data numbers defined as labels and can be used by including the file in an include statement in the program.

## 3 Sound Archive

Now, let's take a close look at the sound archive mentioned in the introduction.

The files in `$NitroSystem/tools/nitro/SoundPlayer/data` are used to explain the sound archive. From this point forward this directory will be called `$data`. Copy `$data` to another location before you start editing the files for your own work.

### 3.1 Sound Archive Definition File

A Sound Archive definition file must be created to create a Sound Archive. Because sound archive definition files are text files, use a text editor to create the file. The sound archive definition file for the sample shown in Code 3-1 is `$data/sound_data.sarc`. Open this file using a text editor.

#### Code 3-1 The Sound Archive Definition File

```

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;
;;  NITRO-Composer Sample
;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

;;;;;;;;;;;;;;;;
;; 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

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

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

;;;;;;;;;;;;;;;;

;; Sequence

```

```

@SEQ
  @PATH "mid"
  SEQ_MARIOKART_TITLE : 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

GROUP_STATIC = 0:
  SEQ_SE
  BANK_SE
  BANK_BGM

```

Lines that start with semicolons are comments. In the text files handled by NITRO-Composer, any line that starts with a semicolon is treated as a comment. First, note the following section of code:

### Code 3-2 Registering Sequence Data

```

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

@SEQ
  @PATH "mid"
  SEQ_MARIOKART_TITLE : SMF, "kart64_title.mid",BANK_BGM, 127, 64, 64, PLAYER BGM

```

This last line in the code is where the sequence data is registered. In code3-2, a standard MIDI file with the filename of `kart64_title.mid` is registered.

In Chapter 4 - Sequence Data, sequence data registration is explained in detail.

## 3.2 @PATH

---

Here is commentary.

There is a statement that starts with @PATH in the sound archive definition file.

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

This statement specifies the directory that has a file called `kart64_title.mid`. In other words, this statement actually registers a file called `mid/kart64_title.mid`.

## 4 Sequence Data

This chapter explains how to create and register sequence data.

### 4.1 Sequence Data Registration

---

First, how to register sequence data is explained.

The section that starts with @SEQ in the Sound Archive Definitions file is the Sequence Data Section that registers sequence data.

#### Code 4-1 The Sequence Data Section

```

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

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

```

In Code 4-1 the statement that starts with SEQ\_MARIOKART64\_TITLE registers one set of sequence data. To add another sequence file, add another statement.

#### Code 4-2 Adding Sequences to the Sequence Data Section

```

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

@SEQ
@PATH "mid"
SEQ_MARIOKART_TITLE : SMF, "kart64_title.mid", BANK_BGM, 127, 64, 64, PLAYER BGM
SEQ_NEW_SEQUENCE    : SMF, "new.mid",          BANK_MK64, 127, 64, 64, PLAYER BGM

```

In Code 4-2, the newly added sequence new.mid is registered the second set of sequence data.

The label SEQ\_NEW\_SEQUENCE specifies this sequence. When this sequence plays back in the program, this label can be used to refer to the sequence.

#### 4.1.1 Specifying Banks

---

To play back sequences, you need bank data. Therefore, specify the bank data to use to play the sequences.

In the SEQ\_MARIOKART sequence example in Code 4-1, the bank BANK\_BGM is specified. This label was defined when the bank data was registered in the sound archive definition file, and this label



represents one set of bank data. Bank data registration is explained in subsequent sections.

If there are mistakes in the bank data, distorted sounds may play or no sounds may play at all. When creating sequence data, you need to check which bank data will be used for playback and correctly associate the sequence data and the bank data in the Sequence Data Section.

### 4.1.2 Specifying the Player

---

Sequences are played using one of 32 players. Therefore, it is necessary to specify which player to use for playback.

By default, only one sequence can be played at a time on one player. Therefore, sequences that play simultaneously need to be played on different players. For more information about players, see “Sound System Manual.”

The player `PLAYER_BGM` is specified in the last line of Code 4-2. This label is defined in the sound archive definition file and represents a player.

Additionally, instead of using a player label, a number from 0 to 31 can be used to specify a player.

For details about player labels, see the Sound Archive Manual.

### 4.1.3 Other Parameters

---

The other parameters are settings for volume, priority, and other settings, but those settings are not explained in this manual. See the Sound Archive Manual for details.

### 4.1.4 About Labels

---

As in the example above, the label may be used in the sound data text file. The label is a placeholder that denotes the set and location of specific data. To denote the set and location of specific data, first define its label and then use that label in another file.

The format of label names must follow these guidelines: The first character must be an uppercase or lowercase Roman letter or underscore ( `_` ). The following characters can be uppercase or lowercase Roman letters, underscores ( `_` ), and numbers. But because label names that start with an underscore have a special meaning for sequence data, it is best not to use it. For details, see the Sequence Data Manual.

The following is an example of a correct label name.

```
SEQ_TITLE_BGM
loop_start
Track_01
```

Naturally, the same label name cannot be used for other data sets.

## 4.2 Creating Sequence Data

---

Sequence data is created as a standard MIDI file. Therefore, sequence data can be created using most commercial sequencers.

For details about the MIDI events that can be used, see the Sequence Data Manual.

## 5 Sequence Archives

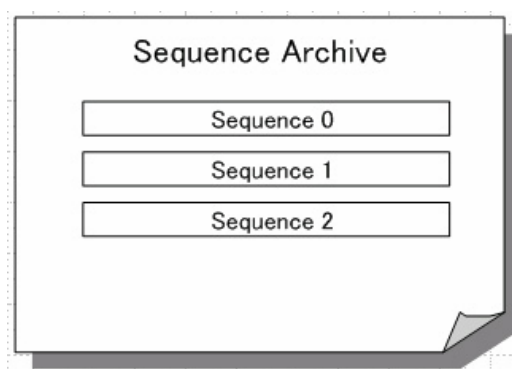
For creating sequence data, a method to create sequence data as a sequence archive is provided. In this chapter, the sequence archive and how to register and create sequence archives are explained.

### 5.1 What is a Sequence Archive?

A sequence archive is a file that groups multiple sets of sequence data together. The sequence data is created and listed in a single file from the start, instead of creating separate files and then combining them, as is the case for a sound archive.

There is no functional difference between sequence data and a sequence archive. The difference is in the way the sequence data is created and how the data is managed.

**Figure 5-1 Conceptual Diagram of Sequence Archive**



#### 5.1.1 Characteristics of Normal Sequence Files

An individual sequence file is special because it can be created from a standard MIDI file. Creating a sequence from a standard MIDI file allows sound designers to use a sequencer, which enables the sound designer to check the sound of the sequence data in real-time.

Usually, this sequence data is loaded into main memory at the start of playback. Thus, the amount of data loaded may increase if the sequence playback is repeated frequently. However, there is an option of loading this kind of sequence data only in advance.

Sequence files are most suitable for creating background music.

### 5.1.2 Characteristics of Sequence Archives

---

Sequence archives are created using a text editor. Therefore, before the sound can be checked, the sequence archive must be converted. However, hundreds or thousands of sequence data can be handled in one file, and the data management becomes easy. Also, the data size of sequence archives can be created to be smaller than normal sequence files.

The data in a sequence archive is normally loaded into main memory during startup or scene changes, and plays back while loading.

Sequence archives are suitable for creating sound effects.

## 5.2 Sequence Archive Registration

---

Sequence archives need to be registered in the sound archive definition file.

The section that begins with @SEQARC in the sound archive definition file is the sequence archive section that registers the sequence archive as shown in Code 5-1.

### Code 5-1 The Sequence Archive Section

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

A line that begins with SEQ\_SE registers one sequence archive. This one entry registers multiple sets of sequence data listed in the sequence archive.

Adding additional lines will register multiple sequence archives.

## 5.3 Creating Sequence Archives

Sequence archives are created with a text editor instead of a sequencer.

\$data/mus/"se.mus" is a sequence archive text file. Look at Code 5-2.

### Code 5-2 Text Sequence Archive

```

////////////////////////////////////
;      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

```

First, the file starts with `#include <../sound_data.sbd1>`. This statement indicates that the sound archive label file (`sound_data.sbd1`) will be included. The sound archive label file is automatically generated during conversion. By including the sound archive label file, the labels defined in the sound archive definition file becomes available. (Using numbers instead of labels makes the description unnecessary.)

The rest can be divided into two sections: the `@SEQ_TABLE` and `@SEQ_DATA` list. They are explained in the following sections.

### 5.3.1 Sequence Tables

Sequence tables are lists of sequence data registered in the sequence archive. Sequence table lists begin with @SEQ\_TABLE as shown in Code 5-3.

#### Code 5-3 Sequence table

```
@SEQ_TABLE
```

```
SE_YOSHI:          yoshi,          BANK_SE, 127, 96, 64, PLAYER_VOICE
SE_WIHAIHO:        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
```

Each line in the sequence table represents a set of sequence data. The order of the list determines the sequence number for each sequence.

### 5.3.2 Registering Sequence Data

To register sequence data, add a new line to the sequence table as shown below.

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

The label SE\_NEW specifies this sequence. To play this sequence from the program, use this label to specify this sequence.

The next label new\_seq is attached to sequence data (explained later). Sequence data with the new\_seq label are in a different location, and the sequence SE\_NEW is played back using the new\_seq sequence data.

The third label BANK\_SE specifies the bank used for playback. Because the sound archive label file is included, specify the bank label defined in the sound archive definition file. Numbers can be used instead of labels to specify banks.

The remaining values are the same as the values set for each sequence data in the sequence data section of the sound archive definition file. For example, the very last label indicates which player is

used. See the Sequence Data Manual for details on the other values.

### 5.3.3 Creating Sequence Data

---

#### 5.3.3.1 Example of Simple Waveform Playback

Describe the sequence data after @SEQ\_DATA.

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

`yoshi:` is the sequence label definition. This label is used in the sequence table to specify the sequence data.

A series of sequence commands follows. Each sequence command is written in this format:

*Command      Argument, Argument, ...*

The number and meaning of the arguments depend on the command.

`prg 0` is the program change command. This command sets the bank that is specified by the sequence table to program number 0.

The program change command is followed by `cn4 127, 0`, which is a note command. The note command will play a note at the pitch of `cn4`, or C4 (natural), with a velocity of 127. The final 0 in the note command is the note length. Surprisingly, a length of 0 corresponds to an infinite length, and the waveform data will play until finished. To playback the entire waveform data, use a length of 0, as shown in this example.

In this case the pitch is set to C4, but the pitch of the playback depends on the original key of the waveform data set in the bank definition file explained later. If a waveform is played at the same pitch as its original key, the waveform data will be played without modification. In this example, the original key is set to C4 in the bank definition file, and therefore, the waveform data will play without modification.

The `fin` command indicates the end of the sequence. If this command is omitted, the next set of sequence data will start processing.

#### 5.3.3.2 Example of Simple Sequence Playback

Let us consider another example.

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

Unlike the previous example, there are two note commands with note lengths that are not zero. What type of sound will be generated?

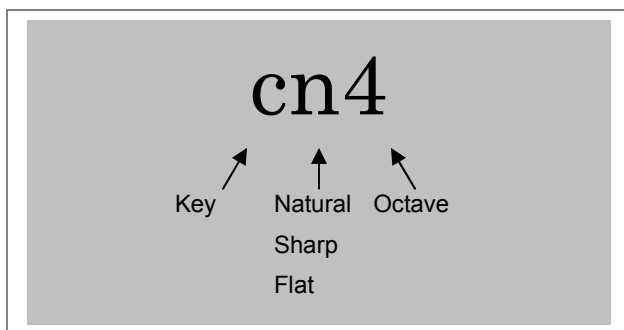


The first note command will play at a pitch of `as5`, or `A#5`. The note length is 6 and is equivalent to a thirty-second note based on the quarter note resolution of 48. Sequence processing stops while sound is generated, and therefore, the second note command will process only after `A#5` is completely generated. Accordingly, `A#5` and then `D#6` will play. (It is possible to play three notes simultaneously without stopping sequence processing. However, this will not be explained here.)

See the Sequence Data Manual for information on other sequence commands.

### 5.3.4 Pitch Notation

This section explains the pitch notation found in sequence note commands. This notation is used to describe pitch using NITRO-Composer.



Standard key names represent each note of the musical scale. Write `c`, `d`, `e`, `f`, `g`, `a`, or `b` for each note.

The second character indicates whether the key is natural, sharp, or flat, and is selected from the letters in Table 5-1 Natural / Sharp / Flat. However, for natural notes, `n` can be omitted and described as `c4`—the same as `cn4`.

**Table 5-1 Natural / Sharp / Flat**

Letter	Description
<code>n</code>	Natural
<code>s</code>	Sharp (semitone higher)
<code>f or b</code>	Flat (semitone lower)

Numbers are used to indicate octaves. Octave 4 is the middle octave, so `an4` plays at 440 Hz. To define a negative number, use `m` instead of a minus sign, as in `cnm1`.

The pitch can be set in the range of `cnm1` to `gn9`.

The pitch notation is not case-sensitive. For example, the description `Cn4` is the same as `cn4`.

## 6 Bank Data

Bank data is required to play sequences. This section will explain how to register and create bank data.

### 6.1 Registering Bank Data

---

First, how to register bank data is explained.

In the sound archive definition file, @BANK starts the bank data section that registers the bank data as shown in Code 6-1.

#### Code 6-1 The Bank Data Section

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

The lines that start with BANK\_SE and BANK\_BGM registers one set of bank data. Bank data is associate with the sequence data using the label defined here.

#### 6.1.1 Association with Waveform Archives

---

WAVE\_SE that follows BANK\_SE specifies the waveform data used by this bank. WAVE\_SE is a label defined in the waveform archive section of the sound archive definition file, and represents a single waveform archive.

The waveform archive section is shown in Code 6-2.

#### Code 6-2 The Waveform Archive Section

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

The waveform archive groups multiple waveforms into one file. Waveform data is much larger than bank data, therefore, bank data and waveform archive data are handled separately. Because they are handled separately, memory can be used more efficiently. However, bank and waveform archive registration are needed to register one bank.

## 6.1.2 Example of Registering Bank Data

---

Code 6-3 is an example of bank data registration.

### Code 6-3 Example of bank data registration

```
;;;;;;;;;;  
;; 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
```

The bank file called `bnk/new.bnk` is registered with the `BANK_NEW` label. The waveform archive used is `WAVE_NEW`.

`swar/news.swls` is registered with the `WAVE_NEW` label. However, there is no need to create this file. This file is generated automatically using the `AUTO` parameter.

The bank data registration can also be written as in Code 6-4.

**Code 6-4 Example of bank data registration (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` like `BANK_MK64`, specifies the waveform archive `WAVE_BGM`. Therefore, `BANK_NEW` and `BANK_BGM` both share `WAVE_BGM`.

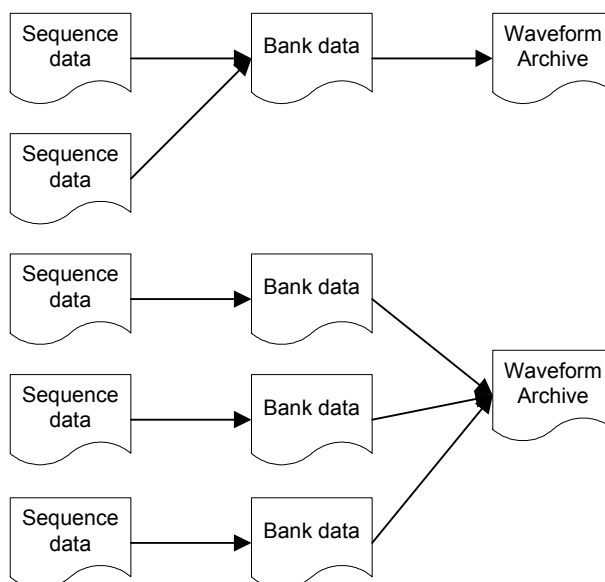
When bank data share a waveform archive, waveform data used by `BANK_NEW` and `BANK_BGM` are both included in `WAVE_BGM`. One waveform archive can be more efficient if the waveform data used by two banks is almost the same.

### 6.1.3 Sound Data Interrelationships

This section reviews the relationships between sequence data, bank data, and waveform archives.

Figure 6-1 is an example of the relationships among sound data.

**Figure 6-1 Relationships Among Various Types of Sound Data**



In the above example, one bank data is associated with one sequence data. However, multiple sequence data can use one bank data.

Furthermore, one waveform archive is associated with one set of bank data. However, multiple bank data can use one waveform archive. For example, if banks share waveform archives, the same waveform files do not consume redundant data if it is used in multiple banks.

Actually, it is also possible to specify more complex associations between banks and the waveform archives. Memory efficiency increases by using this method, but the associations become difficult to manage. Therefore, a description of the implementation is omitted. For details, see the Bank Data Manual and the Sound Archive Manual.

### 6.1.4 Creating Bank Data

---

Bank data is created with a text editor to create a bank definition file.

\$data/bnk/se.bnk is the bank definition file. Code 6-5 is an example of a bank definition file. .

#### Code 6-5 Bank Definition File

```
;*****
;      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
```

The numbers 0, 1, 2, etc. represent program numbers.

In this example, program numbers 0 and 1 contain the parameters `ADPCM` and a specified waveform file. These parameters indicate that the specified waveform files will be used after they are converted to ADPCM format. , AIFF format or WAV format mono waveform files can be specified.

In the bank definition file, describe the original key after the specified waveform file, rather than setting for the waveform data.

The four numbers at the end of each line defines the ADSR of the envelope.

For details on the bank definition file, see Bank Data Manual.

### 6.1.5 Original Key

---

The original key is the unmodified pitch that the waveform data is played. If you properly setting the original key, proper pitch change occurs at note-on from the sequence.

If there is no pitch (as in the voice data), set an arbitrary key and generate the same sequence key for the playback data at the waveform data pitch.

For example, if the original key is set to `cn4`, the note is played at the same pitch when generated from the sequence data at `cn4` –the same as the original key.

The original key setting is unrelated to the sampling rate of the waveform data. Even if the sampling rate changes, there is no need to change the original key setting.

## 7 Memory Management

On the DS, sound data is stored in ROM normally. However, , data must first be loaded into main memory to use sound data. This chapter explains what the sound designer should set up in order to load sound data.

For details on memory management, see Sound System Manual.

### 7.1 Two Types of Heaps

---

In this case, heap refers to the memory region for loading sound data. NITRO-Composer offers two types of heaps.

- Sound heap
- Player heap

The following is a simple explanation for each heap.

#### 7.1.1 Sound Heap

---

The sound heap is used to load sound data needed throughout game play and for upcoming game scenes. The sound heap loads data during game startup and between scenes..

The sound heap is a stack-based heap, and the first data in is the last data to be removed . For example, if data 1, 2, and 3 are loaded onto the stack in order, 2 or 1 cannot be removed until 3 is removed.

#### 7.1.2 Player Heap

---

The player heap is used to load data as needed. An instance of the player heap must be created for each player, and is automatically deleted at the end of sequence playback.

The player heap loads data while the game executes; therefore, loading large amounts of data or frequently needed data is unsuitable for the player heap.. Also, data for sequence archive playback cannot be used.

## 7.2 Creating Groups

Creating groups to load data into the sound heap is convenient. By creating groups, multiple sets of data can be loaded at once.

Create the groups in the group information section of the sound archive definition file.

### Code 7-1 Group Information Section

```

////////////////////////////////////
;; Group

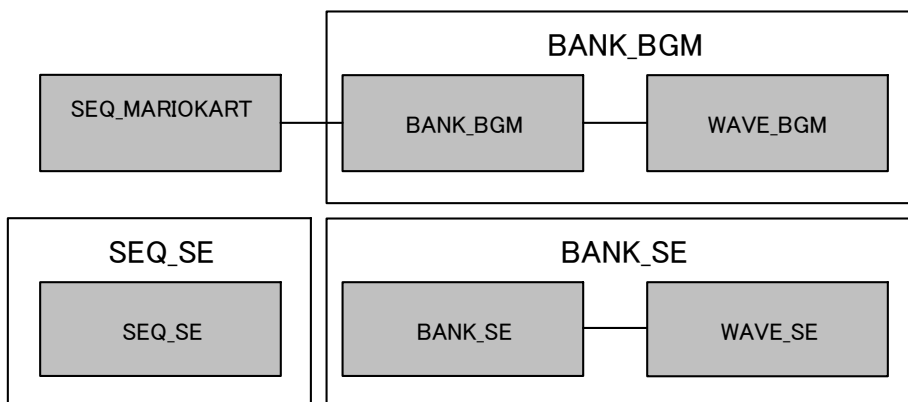
@GROUP

GROUP_STATIC = :
    SEQ_SE
    BANK_SE
    BANK_BGM

```

In Code 7-1, the group `GROUP_STATIC` is defined. The group contains `SEQ_SE`, `BANK_SE`, and `BANK_BGM`. Specifically, data sets like those in Figure 7-1 are included in groups.

**Figure 7-1 Relationship between Group Settings and Sound Data**



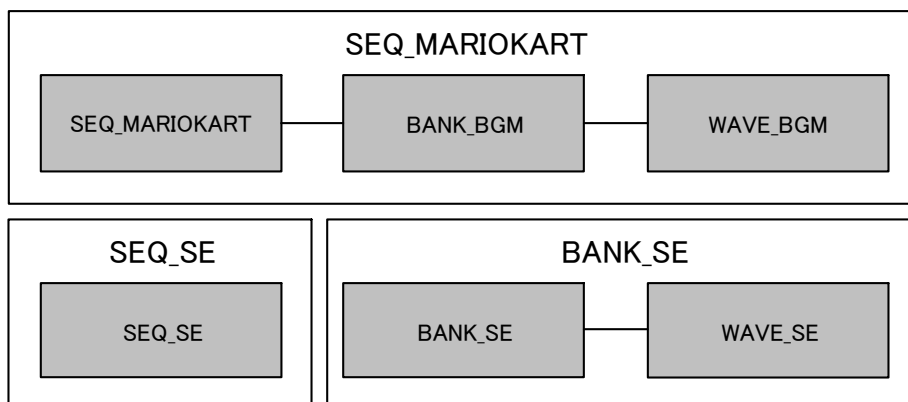
`BANK_SE` and `BANK_BGM` load not only the bank data but also the corresponding waveform archive at the same time. Because the sequence for the sound effects is a sequence archive, the sequence is not directly associated with the bank. Therefore, even if `SEQ_SE` is specified, the bank and waveform data for the sound effects are not loaded at the same time.



On the other hand, because the sequence data is associated with the bank, BANK\_BGM and WAVE\_BGM are loaded at the same time when SEQ\_MARIOKART is specified.

```
GROUP ALL
SEQ_SE
BANK_SE
SEQ_MARIOKART
```

**Figure 7-2 Relationship between Group Settings and Sound Data**



## 7.3 Loading Groups

Once the group is created, that group can be loaded at any time during the game. The sound programmer is responsible for loading the groups. The sound designer must tell the sound programmer which groups to load and at what time.

## 7.4 Creating Player Heaps

To load to the player heap, create a player heap.

Create player heaps in the player information section of the sound archive definition file.

### Code 7-2 Player Information Section

```
;;;;;;;;;;;;;;
;; Player

@PLAYER
0 : 1, 8000
```

Code 4-1 specifies how to create an 8000-byte player heap for player 0. The parameter 1 sets the maximum number of simultaneous plays for the sequence. However, the description about that parameter is omitted. For details, see Sound Archive Manual.

To decide on a player heap size, determine which sound data set is loaded. How to determine the size

of the player heap is explained in a later section.

## 7.5 Loading to Player Heap

---

The data is loaded to the player heap automatically. Normally, data is loaded when a sequence attempts to play back.

Playing sequences require sequence data, bank data, and the waveform archive. However, if these sets are not loaded in the sound heap, they are loaded in the player heap. For example, if both the bank data and waveform archive are loaded already in the sound heap, sequence data is loaded in the player heap.

## 7.6 Deciding the Player Heap Size

---

As stated earlier, the sound data that will be loaded must be known to determine the size of the player heap.. Consider the case where only the sequence data is loaded to the player heap.

First remember that a player heap must be created for each player. Only sequence data that plays on this player is loaded onto the corresponding player heap.. Match the player heap size with the size of the largest sequence data to play back on this player.

The best way to check sequence data size is by using the NITRO-Player. NITRO-Player displays a list of sequence data sizes so the sequence data size can be easily found. For details about NITRO-Player, refer to the documents in `$NitroSystem/docs/NitroPlayer`.

There is also a way to view the sound map file, but a description is omitted from this section. For details, see the Sound Archive Manual.

### 7.6.1 Note on Size Determination

---

Even if you load sound data that has the same size as the player heap, failures due to memory shortages are possible, Specify the size with 100 byte boundaries to play it safe.

## 8 Stream Playback

NITRO-Composer can playback streams as well as sequences. Stream playback is a method that plays back waveform data while the data for the waveform data loads from ROM. The differences between sequence and stream playback are explained in the Quick Start Guide.

This chapter explains how to register and create stream data.

### 8.1 Registering Stream Data

---

First, how to register stream data is explained. In the sound archive definition file, the stream data section to register stream data starts with @STRM as shown below in Code 8-1.

#### Code 8-1 Stream Data Section

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

Using Code 8-1 as an example, a line that begins with STRM\_MARIOKART registers a single sequence data.

STRM\_MARIOKART is a label that can be used to select the specified stream data. To play this stream from a program, use this label to specify the stream.

#### 8.1.1 Specifying a Player Number

---

The stream data is played on a stream player. Use the player number to specify which stream player to use for playback. In Code 8-1, the 0 at the end of each statement is the player number; In this case, stream Player 0 will be used for playback.

Stream players must be defined before use in the stream player information section, in 8.2- Stream Player Definitions.. To register stream data, specify the player numbers of the stream players that were defined in the stream player information section..

A single stream player can play only one stream at a time. Therefore, streams that are played simultaneously with other streams must play back on separate stream players. However, playing more than one stream at once increases processing time dramatically.

#### 8.1.2 Other Parameters

---

Other parameters include settings such as volume and priority. A detailed description of these

parameters is omitted here. For details, see the Sound Archive Manual.

## 8.2 Stream Player Definitions

---

A stream player is needed to play a stream. In the sound archive definition file, define the stream player in the stream player information section, which starts with @STRM\_PLAYER as shown in Code 8-2 Stream Player Information Section.

### Code 8-2 Stream Player Information Section

```

////////////////////////////////////
;; Stream Player

@STRM_PLAYER
0 : STEREO, 6, 7

```

In Code 8-2 Stream Player Information Section, a stereo stream player is assigned player number of 0. To define a mono player, follow the example below.

```

////////////////////////////////////
;; Stream Player

@STRM_PLAYER
0 : MONO, 6

```

A stereo player can play either stereo or mono data, but a mono player can play only mono data. The mono player, however, uses approximately half of the memory of the used by the stereo player.

The number after `STEREO` or `MONO` is the channel number of the channel to be used for stream playback. Because `STEREO` uses two channels, write two channel numbers.

Values from 0 to 15 can be specified for the channel numbers, but there are restrictions on various channels. For details, see the Sound System Manual. In most cases, however, it using channels around 6 and 7 is best.

## 8.3 Creating Stream Data

---

Stream data is created in AIFF or WAV format. Stream data is basically the same as the waveform data registered in banks. However, in stream playback, mono as well as stereo data can be played back.

## 9 SoundPlayer

SoundPlayer is used to check sounds on the Nintendo DS using Sound Archives generated by the sound archiver. This section explains how to use SoundPlayer.

The display on the Nintendo DS when using NITRO\_Player is almost identical to that of SoundPlayer; so it is recommended that you look over how to use SoundPlayer.

### 9.1 How to Use SoundPlayer

To use the SoundPlayer, create a SoundPlayer executable file and execute this file in the following order:

- Create sound data inside the \$data directory.
- Execute \$data/MakeSound.srl.
- \$data/SoundPlayer.srl is generated.
- Load SoundPlayer.srl to the IS-NITRO-DEBUGGER and execute the file.

If the file executes normally, Figure 9-1 SoundPlayer screens will appear on the Nintendo DS screens.

**Figure 9-1 SoundPlayer screens**



### 9.1.1 MakeSound.bat

When `MakeSound.bat` is executed, the following also occurs:

1. The sound archiver `sndarc.exe` starts and the `*.sdat` file is generated.
2. `MakeSoundPlayer.bat` starts and `SoundPlayer.srl` is generated.

If the preceding operations carry out properly, a window will be displayed briefly and then automatically disappear. If an error occurs, the window will remain open and display an error message. Refer to “10 - Error Handling” for information on troubleshooting errors.

The batch file `$data/MakeSound.bat` starts the two processes described above. Modify this file to suit your needs.

**Note:** If the data file has not been updated, nothing will happen even if `MakeSound.bat` is executed. To reconvert all files and data, execute `ReMakeSound.bat` instead of `MakeSound.bat`.

## 9.2 Sound Playback

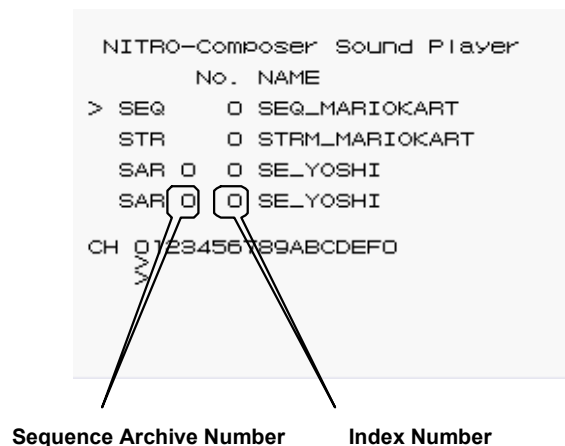
Use the +Control Pad Up/Down keys to select a line that starts with `SEQ` (sequence), `STR` (stream), or `SAR` (sound archive). Point the cursor to the corresponding playback line and press the A Button.

You can play four lines, but each line plays only one sound. Therefore, to play two sounds simultaneously, play the sounds on separate lines.. However, even if the sounds are played on separate lines, simultaneous playback may not be possible; because of restriction on the number of sequences that can play simultaneously on a single player.

Select a sound number using the +Control Pad Left/Right keys. Use the +Control Pad Left/Right keys to select an index number in a Sequence Archive. To select the Sequence Archive itself, hold down the L Button and use the +Control Pad Up/Down keys.

The Sequence Archive number is displayed immediately to the right of `SAR`.

**Figure 9-2 SoundPlayer Top Screen (Sequence Archive number)**



To stop a sequence, point the cursor and press the B Button. This stops the sound playing on the corresponding line.

**Note:** In Figure 9-2, the four lines are arranged in order of SEQ, STR, SAR, and SAR. If there is no stream data, the STR line may be replaced with SEQ.

## 9.3 Selecting the Sound Type

Hold down the L Button and press +Control Pad Up/Down to select the sound type or the sequence archive number. Each time you push the Up key while holding down the L Button, the selection changes as follows:

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

## 9.4 Screen Displays

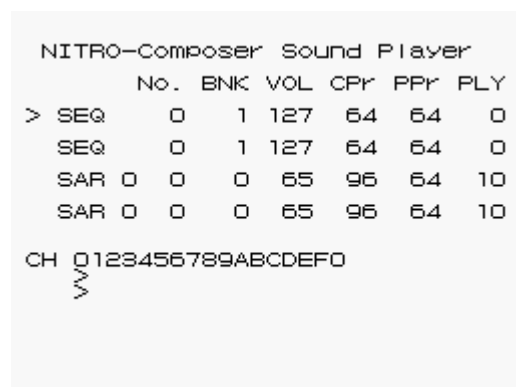
### 9.4.1 Displaying Label Names

The label name of the selected sound appears in NAME as shown in Figure 9-2. If no label is specified, “no name” is displayed.

### 9.4.2 Displaying Parameters

Press the X Button, and the parameters will be displayed instead of the label names as shown in Figure 9-3. Pressing the X Button again displays the label names.

**Figure 9-3 SoundPlayer Upper Screen (parameter information)**



The screenshot shows the NITRO-Composer Sound Player interface. At the top, it says "NITRO-Composer Sound Player". Below that is a header row: "No. BNK VOL CPr PPr PLY". There are four rows of data, each starting with a greater-than sign (>) followed by a label. The first two rows are labeled "SEQ" and the last two are labeled "SAR". Each row contains seven numerical values corresponding to the header. At the bottom, there is a label "CH" followed by a row of numbers "0123456789ABCDEF0" and a small squiggly line below it.

	No.	BNK	VOL	CPr	PPr	PLY
> SEQ	0	1	127	64	64	0
SEQ	0	1	127	64	64	0
SAR 0	0	0	65	96	64	10
SAR 0	0	0	65	96	64	10

CH 0123456789ABCDEF0  
~

Each line in Figure 9-3 has a series of numbers.

No. BNK VOL CPr PPr PLY

No. is the sound number. Use the +Control Pad Left/Right keys to change the value. The values that follow are parameters set for each sound in the sound archives. These values cannot be changed and are only referenced. Starting from the left, the numbers represent the bank number, the master volume, the channel priority, the player priority, and the player number.

For more information about these parameters, see the Sound Archive Manual and the Sequence Data Manual.

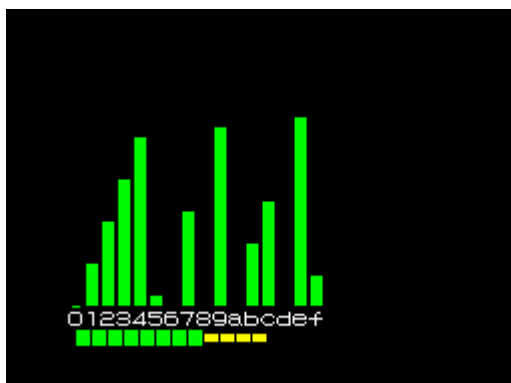
### 9.4.3 Sequence Archive Number

The number that is displayed immediately to the right of `SAR` in Figure 9-3 is the Sequence Archive number. Change the number by holding down the L Button and pushing the +Control Pad Up/Down keys.

## 9.5 Channel Meter

The channel meter is displayed on the lower screen. This meter shows the condition of sound generation on each channel.

**Figure 9-4 The SoundPlayer Channel Meter**



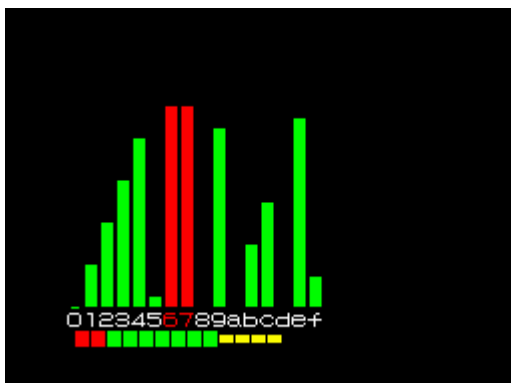
The bars indicate the output volume level for each channel. Below the bars, the numeric values (in hexadecimal) below the bars represent the channel numbers on the Nintendo DS sound hardware.

The very bottom row represents the number of channels generating sound, and the yellow blocks indicate the number of generated sounds being outputted.

When a channel is used for stream playback or anything besides sequence playback, the channels are shown in red as shown in Figure 9-5.



Figure 9-5 The SoundPlayer Channel Meter (Red Display)



## 9.6 Real-time MIDI playback

---

### 9.6.1 What is Real-Time MIDI Playback?

---

Using IS-NITRO-UIC with MIDI connectors or IS-AGB-MIDI, you can use MIDI signals to play sounds from the SoundPlayer. MIDI sequence data can be played without converting sequence data by using this feature.

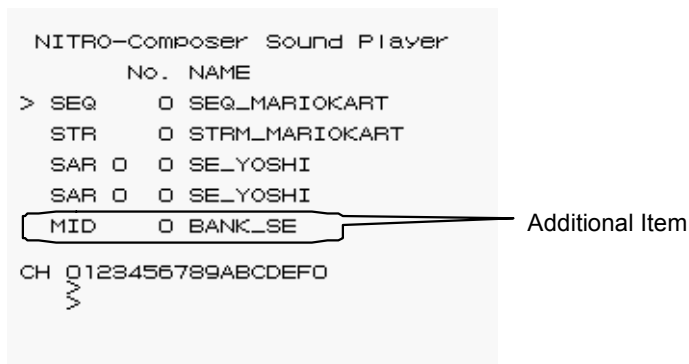
### 9.6.2 Preparation

---

Enabling real-time MIDI playback requires IS-NITRO-UIC with MIDI connectors or IS-AGB-MIDI. For more information on how to connect the system, see the IS-NITRO-UIC or IS-AGB-MIDI manual. The following is an example of how to connect the system:

1. Insert the IS-NITRO-UIC (IS-AGB-MIDI) cartridge into the IS-NITRO-EMULATOR cartridge slot.
2. In the project setting of IST-NITRO-DEBUGGER, turn the cartridge power ON.
3. Start up the SoundPlayer.

If connected properly, `MID` appears in the SoundPlayer display as shown in Figure 9-6.

**Figure 9-6 SoundPlayer display (real-time MIDI)**

### 9.6.3 Bank Selection

Select a bank for real-time MIDI playback.

Use the +Control Pad Up/Down buttons to move to `MID` and the +Control Pad Left/Right buttons to select a bank. After making selection, press the A button to load the bank to use for playing MIDI.

### 9.6.4 MIDI Signal Transmission

By transmitting MIDI signals when a MIDI cable is connected to the IS-NITRO-UIC (IS-AGB-MIDI), the sound can be played using selected banks. By default, Program Number 0 is set, therefore, send the program change message as needed.

## 9.7 Manual Load Mode

### 9.7.1 What is Manual Load Mode?

If you attempt to play the sequence on SoundPlayer, the required data is loaded and played automatically. This feature is very useful for checking sounds. However, automatically loading data while a sequence plays back is rare during actual game play. Usually, loads are grouped and uploaded when scenes change. Therefore, the groups must be set properly to have the sequences play.

In manual load mode the data is loaded manually. Manual load mode allows you to verify if the sequence can be played properly during actual game play. In addition, how much memory is consumed by loading a group is displayed.

The mode that automatically loads data for sound verification is called the automatic load mode to distinguish it from the manual load mode.

### 9.7.2 Operation in Manual Load Mode

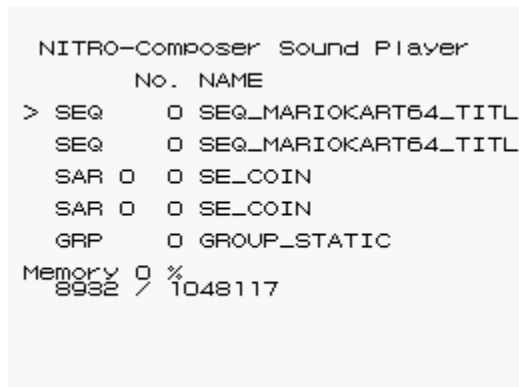
In manual load mode, the data is not loaded automatically. If the necessary data is not loaded, the sequence playback will fail. However, if the player heap settings are configured and has sufficient free memory, the data is loaded to the heap and played.

### 9.7.3 Mode Switching

---

Change from automatic and manual load mode by pushing the SELECT button. Every time the SELECT button is pressed, these two modes toggle.

**Figure 9-7 SoundPlayer Screen (Manual Load Mode)**



Changing modes clears all previously loaded data.

### 9.7.4 Loading Data

---

In the manual load mode, the load execution item and memory use status are displayed instead of the Channel Meter.

The data can be loaded in units of group, sequence, sequence archive, bank, or waveform archive. However, with sequence or bank units, the bank and waveform archive that will be used are loaded at the same time.

GRP (group units) is the default load unit. Hold down the L button and use the +Control Pad Up/Down key to select the load unit.

To load data, select a number with the +Control Pad Left/Right key and press the A Button.

To clear all the loaded data, hold down the L and B button and press the A button.

### 9.7.5 Memory Display

---

The currently used memory is displayed at the very bottom of the screen. Memory is consumed every time data is loaded.

However, even initially a certain amount of memory is used for the sound archive header information and creation of the player heap.

## 9.8 Key Operations

---

Table 9-1 shows the key operations of SoundPlayer.

**Table 9-1 Key Operations in SoundPlayer**

Key	Description
+Control Pad Up/Down	Move cursor
+Control Pad Left/Right	Change value
A Button	Start playback or load data
B Button	Stop playback
L Button and +Control Pad Left/Right	Add 10 to the value while pressed
L Button and +Control Pad Up/Down	Select sequence archive number, sound type, or load units
SELECT	Switch modes
L Button + B Button + A Button	Clear memory

## 9.9 Error Messages

Error messages may appear in the very bottom row of the screen. For example, if the sequence playback fails, error messages may explain the cause.

Error messages are shown in Table 9-2.

**Table 9-2 SoundPlayer Error Messages**

Error Message	Description
Low Priority	A higher priority sequence is being played so playback is impossible.
Invalid Seq No	An undefined sequence number.
Invalid SeqArc No	An undefined sequence archive number.
Invalid Bank No	An undefined bank number.
Invalid WaveArc No	An undefined waveform archive number.
Invalid Group No	An undefined group number.
Invalid SeqArc Index	An undefined sequence archive index.
Invalid Stream No	An undefined stream number.
Invalid Stream Player No	An undefined stream player number.
Memory Over	Loading is impossible because of insufficient memory.
Too Large Priority	Size of the required data is too large so playback is impossible.
Not Found Wave Data	Cannot find the required waveform data (manual load mode).
Not Found Bank Data	Cannot find the required bank data (manual load mode).
Not Found Seq Data	Cannot find the required sequence data (manual load mode).
Not Found SeqArc Data	Cannot find the required sequence archive data (manual load mode).
Not Enough Player Heap for Wave	Player heap is too small to load the waveform data.
Not Enough Player Heap for Bank	Player heap is too small to load the bank data.
Not Enough Player Heap for Seq	Player heap is too small to load the sequence data.

## 10 Error Handling

This section explains how to handle errors that occur while converting sound data.

In this example, `MakeSound.bat` is double clicked to execute the process, but this section can be used as a reference even when `MakeSound.bat` is executed using a different method.

### 10.1 Error Generation

---

If the window remains open after double clicking `MakeSound.bat`, an error has occurred. An error message similar to the one shown below will appear at the bottom of the window.

```
smfconv: BgmSeq/kart64_title.mid: Cannot open file
sndarc: smfconv: Failed
Press any key to continue...
```

### 10.2 Determine the Cause of the Error

---

It can be difficult to pinpoint the causes of errors because a number of error messages may be displayed; in general, troubleshoot starting with the last displayed message and work down the list.

The last line displays a message to close the window, so start troubleshooting from the second to last line.

```
sndarc: smfconv: Failed
```

The line starts with `sndarc:`, which means that all subsequent character strings are output by the **sndarc** tool. In other words, it was the **sndarc** tool that output the `smfconv: failed` message. It seems that **smfconv** has failed, but it does not tell the exact cause. Now, move to the next line up:

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

The **smfconv** tool has output the message `BgmSeq/kart64_title.mid: Cannot open file`. The error message seems to indicate that the file `BgmSeq/Kart64_title.mid` could not be opened. The cause of this error can be determined from this message. **Smfconv** failed because the file `BgmSeq/kart64_title.mid` could not be opened.

### 10.3 Error Handling

---

Chances are that `BgmSeq/kart64_title.mid` could not be opened because this file does not exist. Therefore, check if that file exists.

The error may have possibly occurred because the filename was specified incorrectly. If this is the case, correct the description in the sound archive definition file.

## 10.4 Confirm that an Error Has Been Fixed

---

After troubleshooting the error, double click `MakeSound.bat` again to try and convert the data.

If the window closes immediately, the error was fixed. If the window remains open, go through the suggested troubleshooting procedure again.

# 11 The Next Step

Up to this point, we've learned about the composition of sound data and its broad specifications. To find details about these specifications, refer to the following manuals:

- Learn more about the sound archiver by reading the Sound Archive Manual:

`NITRO_Composer_SoundArchiveManual.pdf`

- Learn more about sequence data by reading the Sequence Data Manual:

`NITRO_Composer_SequenceDataManual.pdf`

- Learn more about bank data by reading the Bank Data Manual:

`NITRO_Composer_BankDataManual.pdf`

For anyone who wants to use NITRO-Player, the Nitro Player User Manual is provided:

`$NitroSystem/docs/NitroPlayer/NITRO_Player_UserManual.pdf`.





© 2004-2005 Nintendo

The contents of this document cannot be duplicated, copied, reprinted, transferred, distributed or loaned in whole or in part without the prior approval of Nintendo.