TWL-System NITRO-Composer

Sound Archive Manual

2009/07/08

The content of this document is highly confidential and should be handled accordingly.

TWL-06-0094-001-B
Released: September 11, 2009

Table of Contents

Code

Tables

Figures

Revision History

Revision Date	Description
2009/07/08	Clarified the amount of memory consumed when individually loading wave data.
2008/05/30	Made revisions in line with the NITRO-System name change (from NITRO-System to TWL-System).
2008/04/08	Changed the format of the Revision History. Changed page headers.
2006/05/29	Added explanation about ability to backward-reference subsequently defined labels.
2005/03/28	Added a description about numeric notation. Made revisions corresponding to changes to the sound map file format.
2005/01/31	Added a description related to the feature for individually loading waveform data. Added a description related to the UNC format notation in @PATH.
2004/12/06	Added explanation of stream option flags.
2004/10/12	Revised to reflect that the label can be used in a player specification. Fixed the explanation of label names.
2004/09/16	Revised description of the label files. Name of the SADL files are unified as “sound label files.” Name of the Bank list files (`.sbdl`) were changed to “sound archive label files.”
2004/09/02	Added description for bank list file output.
2004/08/10	Added the explanation for the Stream data section. Added the explanation for the Stream player information section.
2004/07/20	Modified the document structure throughout the document. Made revisions in conjunction with the ability to link banks to multiple waveform archives. Added explanation for sound map files.
2004/06/01	Added sections on the player information section and the group information section. Revised the overall structure. Revised to reflect the ability to use index numbers in place of labels. Renamed “Data block” to “section.”
2004/04/12	Moved the description of the sound archiver to the Sound Tool Manual. Changed the organization of the chapters. Made additions to the description of the file types.
2004/04/01	Made revisions that are associated with changes to directory organization. Added notes to the description of the file types. Fixed the mistake about the voice priority comparison.

2004/03/01	Initial version.

1 Overview

NITRO-Composer handles a type of file format called sound archive. A sound archive file comprises a collection of the various types of sound data, such as sequence and waveform data, into a single file. In the final product, the sound archive file only needs to be stored in the ROM.

To create the sound archive, the sound data set must be pre-specified in text format. This file is called the sound archive definition file. The following chapters describe how to write the sound archive definition file.

2 Organization

This section describes the organization of the sound archive.

2.1 Sound Archive Definition File Example

The following is an example of a sound archive definition file.

Code 2-1 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    : TEXT, "se.bnk",   WAVE_SE
BANK_BGM  : TEXT, "bgm.bnk",     WAVE_BGM

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


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

;;;;;;;;;;;;;;;;;;;;
;; Sequence
```

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

;;;;;;;;;;;;;;;;;;;;;
;; Sequence Archive

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

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

@STRM_PLAYER
 PLAYER_STRM : STEREO, 6, 7

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

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

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

@GROUP
GROUP_STATIC :
 SEQ_SE
 BANK_SE
BANK_BGM
```

2.2 Sections

The sound archive definition file is divided into eight main sections.

- The waveform archive section (starts with `@WAVEARC`)
- The bank data section (starts with `@BANK`)
- The player information section (starts with `@PLAYER`)
- The sequence data section (starts with `@SEQ`)

- The sequence archive section (starts with `@SEQARC`)
- The stream player information section (`@STRM_PLAYER`)
- The stream data section (starts with `@STRM`)
- The group information section (starts with `@GROUP`)

All section types are defined in the example shown in Code 2-1. However, if you do not need a section, there is no need to define the section.

2.2.1 Waveform Archive Section

Registers waveform archives. See Chapter 6 Waveform Archive Section for a description.

2.2.2 Bank Data Section

Registers bank data. This section also specifies how bank data corresponds with waveform archives. See Chapter 5 Bank Data Section for a description.

2.2.3 Player Information Section

Specifies the parameters to set for each player. See Chapter 8 Player Information Section for a description.

2.2.4 Sequence Data Section

Registers sequence data. Also specifies how sequence data corresponds with banks and parameters, such as the player number. See Chapter 3 Sequence Data Section for a description.

2.2.5 Sequence Archive Section

Registers the sequence archive. See Chapter 4 Sequence Archive Section for a description.

2.2.6 Stream Player Information Section

Specifies the parameters to set for each stream player. See Chapter 9 Stream Player Information Section for a description.

2.2.7 Stream Data Section

Registers the stream data. See Chapter 7 Stream Data Section for a description.

2.2.8 Group information Section

Defines the group to load sound data sets at once. See Chapter 10 Group Information Section for a description.

2.3 Comments

Comments start with a semicolon.

```
; Sample
```

The comment is valid until a line break.

A comment can also be written at the end of a text line.

```
BANK_SE   : TEXT, "se.bnk"       ;;; comment
```

2.4 Label

A label is a string that indicates a specific item. In the following example, `WAVE_MK64` is a label.

```
WAVE_MK64 : AUTO, "mariokart64.swls"
```

The first two characters in a label must be uppercase Roman alphabet characters. All characters that follow may include underscores (_) and numbers in addition to uppercase Roman alphabet characters. The following examples are correctly specified labels.

```
SEQ_TITLE
SEQ_MAP01
SEQ_GAME_OVER
```

Because labels indicate a specific item, the same label name cannot be used for another item.

2.4.1 Backward Referencing of Labels

To define certain data in the sound archive definitions file, you sometimes need to describe links to other data. These linked data are specified using labels, and the order in which the descriptions are made does not matter. In other words, labels that are defined later can be used in places before they are defined.

2.4.2 Label File Output

The labels defined in the definition file for the sound archive are exported to the sound archive files (SADL) and the sound archive label files (SBDL).

2.4.2.1 Sound Label Files

By including a sound label file (SADL) from the program source file, data can be specified in the program by using the same label. Supply the sound label file and sound archive file (SDAT) to the programmer. The sound label file (SADL) also includes the sequence labels in the sound archive.

Note that this file cannot be included in the sound data. To use labels with sound data, include the sound archive label file (SBDL).

2.4.2.2 Sound Archive Label File

By including a sound archive label file (SBDL) from sound data in text-format, the labels defined in the sound archive definition file can be used. For example, a bank specified as `@SEQ_TABLE` in a sequence archive can be described with a label rather than a number.

Use the `#include` directive to include the sound archive label file in your project. For details, see the *Sound Tool Manual*.

2.5 @PATH

By default, the initial path is the directory that contains the sound archive definition file. However, `@PATH` can be substituted for the path of the initial directory.

```
@PATH "bnk"
BANK_SE   : TEXT, "se.bnk"  , WAVE_SE
```

In this example, `bnk/se.bnk` is specified. In other words, the statement in the example above is equivalent to the following statement.

```
BANK_SE   : TEXT, "bnk/se.bnk"  , WAVE_SE
```

You can also substitute the path for directories with `@PATH` in the following UNC format notation.

```
@PATH "//server-1/path/bnk"
```

2.6 Numeric Value Notation

In the sound archive definition file, you can enter values in decimal notation directly to specify parameters (for example, volume). The following notations can be used.

2.6.1 Binary and Hexadecimal Notation

Numeric values are most commonly written in decimal notation, but can also be described in binary and hexadecimal notation.

When representing values in binary or hexadecimal notation, numbers must be preceded by `0b` or `0x`, respectively. For example, the decimal number 12 is written in binary and hexadecimal notation, respectively, as follows.

```
0b1100
0xc
```

2.6.2 Bit Notation

Bit notation is effective when a numeric value contains specifically assigned bits, such as a bit flag. Bit notation describes which bits should be set to 1. For example, to set bits 1, 3, and 6 to 8 with a value of 1, write it as follows.

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

This corresponds to `0b111001010`. Note that the least significant will be 0.

2.6.3 Mathematical Notation

Numeric values may also be expressed in mathematical notation. Binary, hexadecimal, and bit notation may be used as terms in each mathematical notation.

For example, the following mathematical notations are possible.

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

Table 6-1 shows the priority of the operators that can be used in mathematical notation.

Table 2-1 Operators

Priority	Operator	Meaning
1	*	Multiplication
	/	Division
2	+	Addition
	-	Subtraction
3	>>	Right shift
	<<	Left shift
4	<	Left side is less than the right side
	<=	Left side is less than or equal to the right side
	>	Left side is greater than the right side
	>=	Left side is greater than or equal to the right side
5	==	Left side is equal to the right side
6	&	Bitwise AND
7	|	Bitwise OR

3 Sequence Data Section

Sequence data is registered in the sequence data section, which begins with `@SEQ`.

Code 3-1 Sequence Data Section

```
;;;;;;;;;;;;;;;;;;;;
;; Sequence

@SEQ
 @PATH "mid"

SEQ_MARIOKART_TITLE : SMF, "kart64_title.mid", BANK_BGM, 127, 64, 64, PLAYER BGM
```

3.1 Format

The following statement registers a single sequence.

```
SEQ_MARIOKART_TITLE : SMF, "kart64_title.mid", BANK_BGM, 127, 64, 64, PLAYER BGM
```

The general statement format is shown below.

```
label : filetype, filename, bank, volume, channelPrio, playerPrio, playerNo
```

Table 3-1 describes these elements.

Table 3-1 Sequence Definition Elements

Element	Description
`label`	Label name or index number
`filetype`	File type
`filename`	Filename
`bank`	Bank label or bank number
`volume`	Volume
`channelPrio`	Voice priority
`playerPrio`	Player priority
`playerNo`	Player label or player number

The following sections give a detailed description of each element.

3.2 Label Name

This defines a label name to specify a sequence. In each sequence, an index number is assigned starting from zero. You can specify an index number, not include a label name, or omit the label name.

```
SEQ_TITLE = 2     : SMF, "title.mid",     BANK_BGM, 127, 64, 64, PLAYER_BGM
SEQ_MENU  : SMF, "menu.mid",      BANK_BGM, 127, 64, 64, PLAYER_BGM
10        : SMF, "main.mid",      BANK_BGM, 127, 64, 64, PLAYER_BGM
```

The first statement directly specifies an index number. The label name is `SEQ_TITLE`, and the index number is 2.

If only a label name is specified, the index number for this statement is determined by incrementing the index number of the previous statement by one. Therefore, in the second statement in the above example, the index number is 3.

The third statement is assigned an index number only, and the label name is undefined. Because this sequence is assigned only an index number, the sequence must be referred to using the index number.

3.3 File Type

Specifies the file type of the sequence data file. The file that is actually stored in the sound archive is a binary sequence file. A method is provided to automatically convert files to a binary sequence file even if a different file type is specified.

Table 3-2 shows the file types that can be specified.

Table 3-2 Sequence File Types

File Type	Description
`SMF`	Standard MIDI file.
`TEXT`	Text sequence file.
`BIN`	Binary sequence file.

See the *Sequence Data Manual* for details on each format.

3.4 Filename

Specifies the filename for the sequence data file. By default, the path is the relative path from the sound archive definition file. However, by using the `@PATH` command, the initial directory can be changed.

Make sure that the actual and specified file types match.

3.5 Bank

Specifies the bank label or number of the bank used to play back sequences. Banks are defined in the bank data section. See Chapter 5 Bank Data Section for details.

3.6 Volume

Specifies the volume of the entire sequence. The value ranges from 0 to 127.

3.7 Voice Priority

Specifies the voice priority for the sequence. Voice priority determines which sound has priority among the16 channels. The value ranges from 0 to 127. The greater value receives higher priority.

See the *NITRO-Composer Sound System Manual* for details on voice priority.

3.8 Player Priority

Specifies the player priority. Player priority determines which sequence has priority to play among the maximum number of sequences that can be played. The value ranges from 0 to 127. The greater value receives higher priority.

See the *NITRO-Composer Sound System Manual* for details on player priority.

3.9 Player Number

Specifies the player number of the player that should be used for sequence playback. If a label for the player is defined in the player information section first, the player can be specified with its label. When you specify the player with the player number, specify a value from 0 to 31. When specifying player number, consider that a single player can play only a limited number of sequences.

4 Sequence Archive Section

The sequence archive section is a section for registering sequence archives. This section begins with `@SEQARC`.

Code 4-1 Sequence Archive Section

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

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

4.1 Format

The following statement registers a single sequence archive.

```
SEQ_SE  : TEXT, "se.mus"
```

The general statement format is shown below.

```
label : filetype, filename
```

Table 4-1 describes these elements.

Table 4-1 Sequence Archive Definition Elements

Element	Description
`label`	Label name or index number.
`filetype`	File type.
`filename`	File name.

The following sections describe each element.

4.2 Label

Defines a label name to specify a sequence archive. An index number is assigned to each sequence archive starting from zero. It is also possible to specify an index number directly or omit the definition of a label name.

```
SEQ_SE = 2    : TEXT, "se.mus"
SEQ_ENEMY_SE  : TEXT, "enemy_se.mus"
10            : TEXT, "common.mus"
```

The first statement specifies an index number. The label name is `SEQ_SE`, and the index number is 2.

If only a label name is specified, the index number for this statement is determined by incrementing the index number of the previous statement by one. Therefore, in the second statement in the above example, the index number is 3.

The third statement is assigned an index number only, and the label name is undefined. Because this sequence archive is assigned only an index number, the sequence archive must be referred to using the index number.

4.3 File Type

The file that is actually stored in the sound archive is a binary sequence archive file. A method is provided to automatically convert files to a binary sequence archive file even if a different file type is specified.

Table 4-2 shows the file types that can be specified.

Table 4-2 Sequence Archive File Types

File Type	Description
`TEXT`	Text sequence archive file.
`BIN`	Binary sequence archive file.

See the *NITRO-Composer Sequence Data Manual* for details on each format.

4.4 Filename

Specifies the filename for the sequence archive. By default, the path is a relative path from the sound archive definition file. However, by using the `@PATH` command, you can change the initial directory.

Make sure that the file type of the actual and specified files match.

5 Bank Data Section

The bank data section is a section for registering the bank data. It begins with `@BANK`.

Code 5-1 Bank Data Section

```
;;;;;;;;;;;;;;;;;;;;;
;; Bank

@BANK
@PATH "bnk"
BANK_SE=0 : TEXT, "se.bnk"        WAVE SE
BANK_BGM  : TEXT, "bgm.bnk",      WAVE_BGM
```

5.1 Format

The following statement registers a single bank.

```
BANK_BGM  : TEXT, "bgm.bnk",      WAVE_BGM
```

The general statement format is shown below.

```
label : filetype, filename, wavearc0, wavearc1, wavearc2, wavearc3
```

Table 5-1 describes these elements.

Table 5-1 Bank Definition Elements

Element	Description
`label`	Label name or index number.
`filetype`	File type.
`filename`	Filename.
`wavearc0`	Number 0 waveform archive. (Can be omitted.)
`wavearc1`	Number 1 waveform archive. (Can be omitted.)
`wavearc2`	Number 2 waveform archive. (Can be omitted.)
`wavearc3`	Number 3 waveform archive. (Can be omitted.)

The following is the detailed explanation of each element.

5.2 Label

Defines a label name to specify a bank. An index number is assigned to each bank starting from zero. It is also possible to directly specify an index number or omit the definition of a label name.

```
BANK_SE = 2       : TEXT, "se.bnk", WAVE_SE
BANK_BGM  : TEXT, "bgm.bnk", WAVE_BGM
10        : TEXT, "common.bnk", WAVE_COMMON
```

The first statement specifies an index number. The label name is BANK_SE, and the index number is 2.

If only a label name is specified, the index number for this statement is determined by incrementing the index number of the previous statement by one. Therefore, in the second statement in the above example, the index number is 3.

The third statement is assigned an index number only, and the label name is undefined. Because this bank is assigned only an index number, the bank must be referred to using the index number.

5.3 File Type

Specifies the file type of the bank file. The file that is actually stored in the sound archive is a binary sequence bank file. A method is provided to automatically convert files to a binary bank file even if a different file type is specified.

Table 5-2 shows the file types that can be specified.

Table 5-2 Bank File Types

File Type	Description
TEXT	Text bank file.
BIN	Binary bank file.

See the *NITRO-Composer Bank Data Manual* for details regarding the text bank format.

Be aware that if a binary bank file is specified, the feature that automatically generates the waveform list file explained in section 6.3.1 Automatic Creation of Waveform List Files cannot be used.

5.4 Filename

Specifies the filename for the bank data. By default, the path is a relative path from the sound archive definition file. However, by the using @PATH command, the initial directory can be changed.

Be sure that the file type of the specified and actual files match.

5.5 Waveform Archive

Using a label or an index number, specify the waveform archive that stores the waveform data used by the bank. How the waveform archive is specified depends on how the waveform file is defined in the bank definition file.

5.5.1 When Waveform Data Is Not Used

If waveform data is not used in the bank definition file, the waveform archive does not need to be specified.

```
BANK_SE   : TEXT, "se.bnk"
```

5.5.2 When @WGROUP Is Not Used

If @WGROUP is not used for specifying a waveform group in the bank definition file, specify one waveform archive.

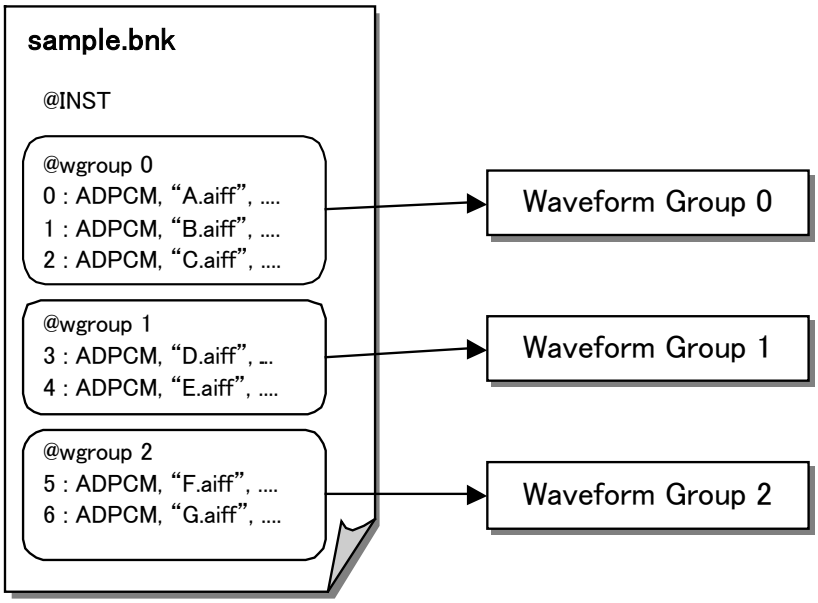
```
BANK_MK64 : TEXT, "mariokart64.bnk",     WAVE_MK64
```

WAVE_MK64 is the waveform archive label defined in the waveform archive section. WAVE_MK64 must contain all of the waveform data sets used by BANK_MK64. However, if the waveform archive generation feature is used, containing waveform data sets in WAVE_MK64 that are used by BANK_MK64 shouldn't be an issue. See Chapter 6 Waveform Archive Section for details.

5.5.3 When @WGROUP Is Used

Waveform data sets can be divided into a maximum of four groups using @WGROUP in the bank definition file. When the waveform data are grouped, each group must be associated with a waveform archive. For example, assume that the waveform data sets were grouped in the bank file shown in Figure 5-1.

Figure 5-1 Example of Waveform Data Grouping



As shown below, three waveform archives must be specified to register this bank.

```
BANK_SAMPLE       : TEXT, "sample.bnk",  WAVE_0, WAVE_1, WAVE_2
```

In this example, WAVE_0, WAVE_1, and WAVE_2 correspond to Waveform Group 0, Waveform Group 1, and Waveform Group 2, respectively. In other words, WAVE_0 must contain the converted A.aiff, B.aiff, and C.aiff files. If the waveform archive automatic generation feature is used, each group does not have to be associated with a waveform archive. See Chapter 6 Waveform Archive Section for details.

See the *NITRO-Composer Bank Data Manual* for details on grouping waveform data sets in the bank definition file.

6 Waveform Archive Section

The Waveform Archive section is a section for registering waveform archives. The waveform archive section begins with `@WAVEARC`.

Code 6-1 Waveform Archive Section

```
;;;;;;;;;;;;;;;;;;;;
;; Wave Archive

@WAVEARC
 @PATH "swar"
WAVE_SE   : AUTO, "se.swls"
WAVE BGM  : AUTO, “bgm.swls”
```

6.1 Format

The following statement registers a single waveform archive.

```
WAVE_BGM  : AUTO, "bgm.swls"
```

The general statement format is shown below.

```
label : filetype, filename
```

Table 6-1 describes these elements.

Table 6-1 Waveform Archive Definition Elements

Element	Description
`label`	Label name or index number.
`filetype`	File type.
`filename`	Filename.
`flags`	Option flags. (Can be omitted.)

The following is the detailed explanation of each element.

6.2 Label

Defines a label name to specify a waveform archive. An index number is assigned to each waveform archive starting with zero. It is also possible to directly specify an index number or omit the label name.

```
WAVE_SE = 2   : AUTO, "se.swlsmus"
WAVE_BGM      : AUTO, "bgm.swls"
10            : AUTO, "common.swls"
```

The first statement specifies an index number. The label name is `WAVE_SE`, and the index number is 2.

If only a label name is specified, the index number for this statement is determined by incrementing the index number of the previous statement by one. Therefore, in the second statement in the above example, the index number is 3.

The third statement is assigned an index number only, and the label name is undefined. Because this bank is assigned only an index number, the bank must be referred to using the index number.

6.3 File Type

Specifies the file type of the waveform archive. The file stored in the sound archive is a waveform archive file. A method is provided to automatically convert files to a binary bank file even if a different file type is specified. Table 6-2 shows the file types that can be specified.

Table 6-2 Waveform Archive File Types

File Type	Description
AUTO	Automatic generation of waveform list file
TEXT	Waveform list file
BIN	Waveform archive file

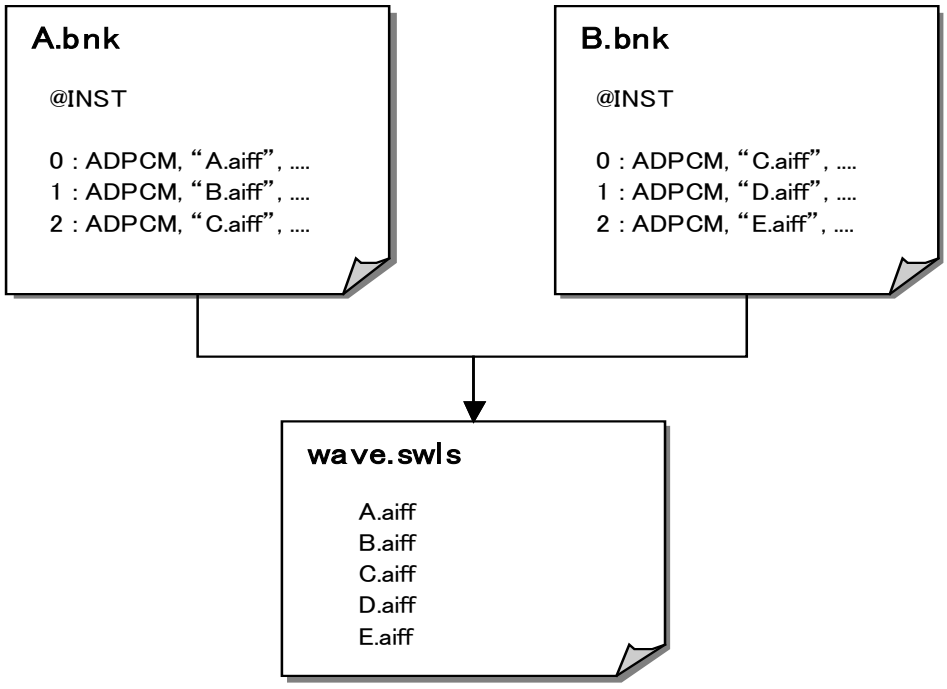
The waveform list file lists a path per line for each waveform file. Based on this list, a single waveform archive file is created by combining multiple waveform files.

6.3.1 Automatic Creation of Waveform List Files

The waveform archive that stores the waveform files that are used by a bank must be specified for each bank. However, adding waveform files to the waveform list whenever the number of waveform files in the bank changes is cumbersome. Therefore, a waveform file specified as an AUTO file type can be set to generate automatically.

When the file type is set to AUTO, the associated banks are searched and a waveform list file is automatically generated from the collected waveform file information. If there are multiple associated banks, all waveform files used by the banks are included in the waveform list file. If a waveform file is used by more than one bank, the waveform file information is combined into one waveform list file.

Figure 6-1 Automatic Generation of Waveform List Files



Specifically, `wave.swls` lists filenames such as `A.aiff` after conversion.

The file of the associated bank must be a text file. The waveform file information cannot be obtained from non-text files. Therefore, the waveform list file cannot be automatically generated.

6.4 Filename

Specifies the filename for the waveform archive. By default, the path is a relative path from the sound archive definition file. However, by using `@PATH` command, the initial directory can be changed.

Make sure that the specified and actual file type match. When the file type is set to `AUTO`, specify a file name that will not conflict with an existing file.

6.5 Option Flags

Specifies the option flag that configures supplemental settings related to a waveform archive. If this option is not specified, the statement can be omitted.

```
WAVE_COMMON      : AUTO, "common.swls", s
```

The following can be designated as an option flag. The specified option is enabled when the flag is set.

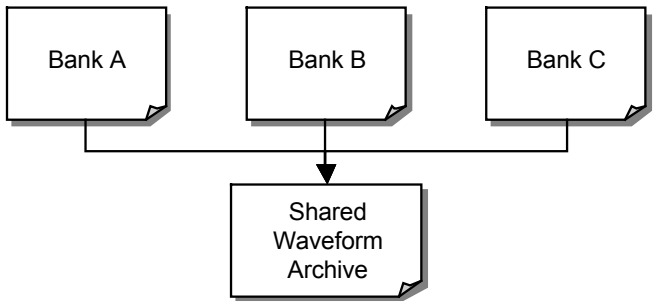
Table 6-1 List of Waveform Archive Option Flags

Flag	Description
`-s`	When loading waveform data, this option loads only the necessary waveform data from the waveform archive instead of the entire waveform archive. (A detailed explanation follows.)

6.5.1 Loading Only Waveform Data

When loading waveform data, a waveform archive with the `-s` option loads only the necessary waveform data in the waveform archive instead of loading the entire archive. This option works well when a single waveform archive is shared over multiple banks, as in Figure 6-2.

Figure 6-2 Shared Waveform Archive



Waveform data used only in Bank A may be included in the shared waveform archive. When using Bank B, if you load waveform data for Bank A that Bank B does not use, the memory efficiency decreases.

Therefore, if the `-s` option flag is specified for this shared waveform archive, only the waveform data used by Bank B loads even if there is waveform data information for Bank A and C in the shared waveform archive.

The following sections discuss precautions to keep in mind when specifying the `-s` option flag.

6.5.1.1 Overhead on Memory Consumption and Load Time

The waveform data sets are handled individually instead of at once, causing a greater increase in overhead for memory consumption and data set loading time. (Specifically, 32 bytes of memory are consumed for management purposes for every four wave files.) In most cases, the overhead issue can be ignored because unnecessary data sets are not loaded, which reduces memory consumption and shortens data set loading time. However, the overhead increases if there is little or no unnecessary data.

In such cases, consider using a procedure for waveform data group management. For more details about waveform archive group management, see section 5.5.3 When @WGROUP Is Used or the *NITRO-Composer Bank Data Manual*.

6.5.1.2 Caution About Loading Waveform Data and Banks

Because the `-s` option flag loads waveform data individually into the specified bank, you must specify that the waveform data and the bank need to load at the same time.

For example, if you do not specify the option flag to load the waveform data or you specify the waveform archive to load the waveform data, the necessary waveform data does not load, and the sounds does not generate properly.

7 Stream Data Section

The stream data section is a section for registering the stream data. The section starts with `@STRM`.

Code 7-1 Stream Data Section

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

7.1 Format

The following statement registers a single stream.

```
STRM_MARIOKART : PCM8, "kart_title.32.aiff",  127, 64, PLAYER STRM
```

The general statement format is as follows.

```
label : format, filename, volume, playerPrio, playerNo[, flags]
```

Table 7-1 describes these elements.

Table 7-1 Sequence Definition Elements

Element	Description
`label`	Label name or index number.
`format`	Data Format.
`filename`	File name.
`volume`	Volume.
`playerPrio`	Player priority.
`playerNo`	Player label or number.
`flags`	Option flags. (Can be omitted.)

The following sections provide a detailed description of each element.

7.2 Label

Defines a label name to specify a stream. An index number is assigned to each stream starting from zero. It is possible to directly specify an index number or omit the label name.

```
STRM_TITLE = 2    : PCM8, "title.aiff,    127, 64, PLAYER STRM
STRM_MENU         : PCM8, "menu.aiff",    127, 64, PLAYER STRM
10                : PCM8, "main.aiff",    127, 64, PLAYER STRM
```

The first statement specifies an index number. The label name is `STRM_TITLE`, and the index number is 2.

If only a label name is specified, the index number for this statement is determined by incrementing the index number of the previous statement by one. Therefore, in the second statement in the above example, the index number is 3.

The third statement is assigned an index number only, and the label name is undefined. Because this bank is assigned only an index number, the bank must be referred to using the index number.

7.3 Data Format

Specifies which data format to convert the waveform data to. Table 7-2 lists the data formats that can be used.

Table 7-2 Stream Data Formats

Format	Description
PCM16	Converts to 16-bit PCM data.
PCM8	Converts to 8-bit PCM data.
ADPCM	Converts to ADPCM data.
STRM	No conversion. (Specifies a converted file.)

Each data format has its own advantages and disadvantages. Table 7-3 shows the characteristics of each data format.

Table 7-3 Characteristics of the Stream Data Format

Format	Size	Sound Quality	Processing Time
PCM16	large	best	medium
PCM8	small	fair	least
ADPCM	smallest	good	most

Size compares the data size of the formats. PCM16 uses the most ROM capacity. ADPCM is the most compact format. The ratio of the sizes of the three formats is shown below.

```
PCM16 : PCM8 : ADPCM = 4 : 2 : 1
```

PCM16 has the best sound quality, but there is only a slight difference in sound quality between PCM16 and ADPCM. PCM8 has somewhat noticeable quantization noise.

Processing time is the processing time required to stream playback. ADPCM requires a decoding process. Therefore, it takes a longer time to process. With PCM8, less data is loaded than with PCM16. Therefore, the processing time is relatively short.

7.4 Filename

Specifies the filename for the stream data file. If the specified format is something other than STRM, specify an AIFF or WAV file. If the specified format is STRM, specify a converted STRM file.

Waveform files can be played in stereo or mono. However, note that stereo data cannot be played unless the stream player is configured for stereo playback. By default, the path is a relative path from the sound archive definition file. But the initial directory can be changed using the `@PATH` command.

7.5 Volume

Specifies the volume for the entire stream. The range of the value is from 0 to 127.

7.6 Player Priority

Specifies the player priority. Player priority determines if the currently playing stream has priority over a new stream set to play. The value ranges from 0 to 127. The greater value receives higher priority.

7.7 Player Number

Specifies which stream player to use for playback. When the stream player information section is defined beforehand, the player can be specified by using a label. When specifying the stream player by using a number, specify a value of between 0 and 3. Only the stream player defined in the stream player information section (explained in Chapter 9 Stream Player Information Section) can be used. Additionally, only one stream can be played back with one stream player, and this condition must be taken into consideration for specifying the player number.

7.8 Option Flag

Specifies the option flag that configures supplemental settings related to a stream. If this option is not specified, the description can be omitted.

```
STRM_MARIOKART : PCM8, "kart_title.32.aiff", 127, 64, PLAYER_STRM, s
```

Table 7-4 shows the option flag that can be specified. The specified option is enabled when the flag is set.

Table 7-4 List of Stream Option Flags

Flag	Description
`s`	Use two channels to play at a louder volume when playing mono data on a stereo player. (The default is one channel.)

8 Player Information Section

The player information section sets parameters for each of the 32 players. The section begins with `@PLAYER`.

Code 8-1 Player Information Section

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

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

8.1 Format

The following statement configures a single player. Sequences can still be played back with unconfigured players. In that case, the default element values are used.

```
PLAYER_BGM  : 1, 8000
```

The format is as follows.

```
PlayerNo.: seqMax, heapSize, chBitFlag
```

Table 8-1 describes these elements.

Table 8-1 Player Information Definition Elements

Element	Description
`playerNo`	Player number or label.
`seqMax`	Maximum number of sequences to play simultaneously. (Default is 1.)
`heapSize`	Player heap size. (Can be omitted.)
`chBitFlag`	Allocatable channel bit flag. (Can be omitted.)

The following sections describe each element in detail.

8.2 Player Number

Assigns a player number from 0 to 31 for a specific player, as shown below.

```
PLAYER_BGM = 1  : 1
PLAYER_ENEMY    : 4
10          : 1
```

The first statement shows that the player number is 1, and the label name is `PLAYER_BGM`.

If only a label name is specified, the player number for this statement is determined by incrementing the player number of the previous statement by one. Therefore, in the second statement in the above example, the player number is 2.

The third statement is assigned a player number only, and the label name is undefined. Because this player is assigned only a player number, the player must be referred to using the player number.

8.3 Maximum Number of Concurrent Sequences

Specifies the maximum number of sequences that can play at the same time on each player. The default value is 1. The values can be set from 0 to 16. If 0 is specified, the player does not play back sequences. If 16 is specified, there is no limit on the number of sequences that can play at the same time on a player.

For details on the maximum number of concurrent sequences, see the *NITRO-Composer Sound System Manual*.

8.4 Player Heap Size

Specifies the size of the player heap. Player heaps will be created for the maximum number of sequences that can play at the same time.

For example, when the following specification is made, three 1000-byte player heaps are created for the player that has the player number 5.

```
5 : 3, 1000
```

Three player heaps are required because up to three sequences can be played at the same time.

The player heap size does not need to be specified. If the player heap size is not specified, the player heap is not created.

For more details about the player heap, see the *NITRO-Composer Sound System Manual*.

8.5 Allocating Channel Bit Flags

Specifies the bit flag for the sequence playback channel available for allocation on the configured player. Specifying 0 will enable bit flag allocation from all channels. The default setting is 0.

The channel bit flag expresses channel numbers 1, 2,…, 15 in order from the least significant bit. The channel number is the channel number of the TWL and Nintendo DS hardware. Channels with enabled bits can be allocated.

For example, the following statement uses only channels 0-7.

```
0 : 1, 0, 0x00ff
```

For more details about the limitations on sequences and channels, see the *NITRO-Composer Sound System Manual*.

9 Stream Player Information Section

The stream player information section defines stream players. The stream player information section starts with `@STRM_PLAYER`.

Code 9-1 Stream Player Information Section

```
;;;;;;;;;;;;;;;;;;;;
;; Stream Player

@STRM_PLAYER
PLAYER STRM : STEREO, 6, 7
```

9.1 Format

The following statement configures a single stream player. Only configured players can be used.

```
PLAYER STRM : STEREO, 6, 7
```

The general statement format is shown below.

```
playerNo : type, chNo[, chNo]
```

Table 9-1 describes these elements.

Table 9-1 Stream Player Information Definition Elements

Element	Description
`playerNo`	Player number or label.
`type`	Player type.
`chNo`	Channel number.

The following section describes each element in detail.

9.2 Player Number

Specifies the player number to set for the player using a value between 0 to 3. The player number can be specified by using a label as shown below.

```
PLAYER_BGM = 1 : STEREO 4, 5
PLAYER_FANFARE : MONO, 6
3          : MONO, 7
```

The first statement shows that the player number is 1 and the label name is `PLAYER_BGM`.

If only a label name is specified, the player number for this statement is determined by incrementing the player number of the previous statement by one. Therefore, in the second statement in the above example, the player number is 2.

The third statement is assigned a player number only, and the label name is undefined. Because this player is assigned only a player number, the player must be referred to using the player number.

9.3 Player Type

Specifies whether the player has stereo or mono sound.

Table 9-2 Stream Player Types

Player Type	Description
MONO	Mono player.
STEREO	Stereo player.

A stereo player can play both stereo and mono sound, but a stereo player requires a stream buffer that is twice the size of the stream buffer that is required by a mono player.

A mono player can play only mono data.

9.4 Channel Number

Specifies the channel number for a channel that plays stream data. Specify two channel numbers for the stereo player and one for the mono player.

Channel numbers can be specified in the range from 0 through 15. Because the functionality of each channel is different, coordinate with other channel usage. For details on the functionality of each channel, see the *NITRO-Composer Sound System Manual*.

10 Group Information Section

The group information section defines the groups that load multiple sound data sets at the same time. The group information section begins with `@GROUP`.

Code 10-1 Group Information Section

```
;;;;;;;;;;;;;;;;;;;;
;; Group

@GROUP
GROUP_STATIC:
 SEQ_SE
 BANK_SE
 BANK_MK64
```

10.1 Format

The following statement configures a single group.

```
GROUP_STATIC:
 SEQ_SE
 BANK_SE
 BANK_MK64
```

The general statement format is shown below.

```
groupLabel :

   dataLabel, option

   dataLabel, option

   dataLabel, option
```

Table 10-1 describes these elements.

Table 10-1 Group Information Definition Elements

Element	Description
`groupLabel`	Group label
`dataLabel`	Sound data label
`option`	Option (can be omitted)

First, define the group label. In the following statement, specify the sound data that should be included in that group. Any number of sound data sets can be included.

The following explains each element in detail.

10.2 Group Label

Defines the label name for the group. An index number is assigned to each group starting with zero. It is possible to directly specify the index number or omit the definition of a label name.

```
GROUP_COMMON = 2 :
GROUP_TITLE      :
10               :
```

The first statement above directly specifies an index number. The label name is `GROUP_COMMON`, and the index number is 2.

If only a label name is specified, the index number for this statement is determined by incrementing the player number of the previous statement by one. Therefore, in the second statement in the above example, the index number is 3.

The third statement is assigned an index number only, and the label name is undefined. Because this group is assigned only an index number, the group must be referred to using the index number.

10.3 Sound Data Label

Specifies the sound data to be included in the group using labels. Sequences, sequence archives, banks, and waveform archives can be included as sound data.

When a sequence is specified, the banks and waveform archives used by the sequence are also included in the group. In addition, when a bank is specified, the waveform archives used by the bank are also included. However, if a sequence archive is specified, no banks or other data are automatically included.

Table 10-2 summarizes the above text.

Table 10-2 Data Included in Group with Specified Data Types

	SEQ	BANK	WAVEARC	SEQARC
SEQ	X	X	X	
BANK		X	X	
WAVEARC			X	
SEARC				X

The left column is the specified data type. The items indicated with an “X” are also included in the group. Finer control is possible by using the options described in the following sections.

The sound data will not be duplicated in a group even if the same sound data is registered in a single group more than once.

10.3.1 Specifying Index Numbers

Instead of labels, index numbers can be specified. Write the index number in place of the data label as shown below.

```
Data type [ index number ]
Specify SEQ, SEQARC, BANK, or WAVEARC for data types.
GROUP_1 :
SEQ[ 2 ]
BANK[ 1 ]
```

10.4 Options

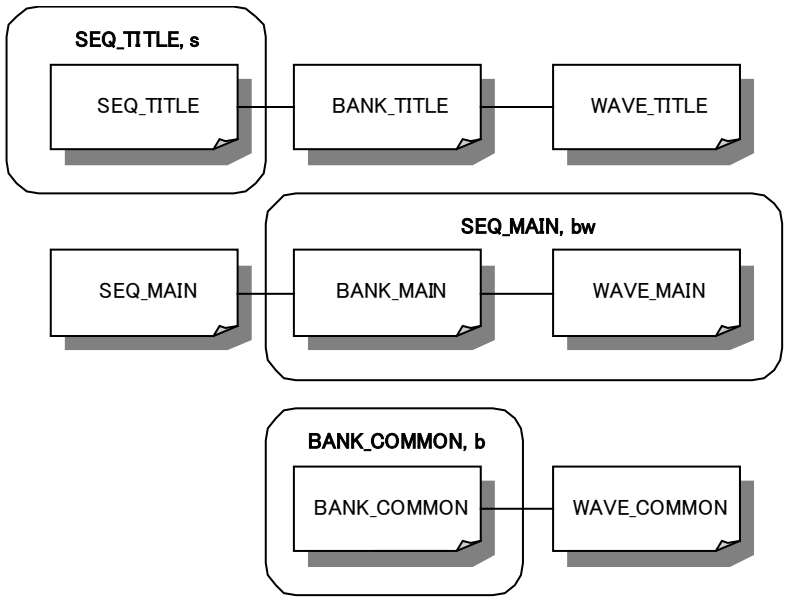
Options specify the data to include. Options can be omitted. Specify any combination of `s`, `b`, and `w`. Examples are shown below.

```
GROUP_2 :
SEQ_TITLE, s
SEQ_MAIN, bw
BANK_COMMON, b
```

`s`, `b`, and `w` refer to sequences, banks, and waveforms respectively.

When using `SEQ_TITLE` alone, all associated sequences, banks, and waveform archives are included in the group. When option `s` is used, only the sequence data is included. With option `bw`, sequences are not included, but banks and waveform archives are included. Figure 10-1 maps out the examples discussed above.

Figure 10-1 Example of Options



Options cannot be specified for sequence archives. Option `s` cannot be used with banks.

11 Sound Map Files

11.1 Sound Map File

The sound map file is a text file that contains information about the kind of data that is stored in the sound archive. By looking at the sound map file, you can easily verify the size of the group and sets of sound data.

This file is created automatically when a sound archive is created. If the name of the sound archive definition file is `sound_data.sarc`, the sound map file is output as `sound_data.smap`.

11.2 Sound Map File Contents

The sound map file generated by the sound data conversion is shown below.

Code 11-1 Sound Map File

```
# GROUP:
# label                  number     hsize  type    number     hsize
  GROUP_STATIC                0    208352
                                       SEQARC      0       736
                                       BANK        0     42592
                                       BANK        1    165024
# SEQ:
# label                  number fileID bnk vol cpr ppr ply     hsize      size name
  SEQ_MARIOKART               0      0   1 127  64  64   0    172032      6928
mid/kart64_title.sseq
# SEQARC:
# label                  number fileID                                 size name
  SEQ_SE                      0     1                                 652 mus/se.ssar
# BANK:
# label                  number fileID wa0 wa1 wa2 wa3        hsize      size name
  BANK_SE                     0     2   0                   42592       132 bnk/se.sbnk
  BANK_BGM                    1     3   1                  165024      1156
bnk/bgm.sbnk
# WAVEARC:
# label                  number fileID                                 size name
  WAVE_SE                     0     4                               42304
swar/se.swar
  WAVE_BGM                    1     5                              163684
swar/bgm.swar
# STRM:
# label                  number fileID vol pri ply                      size name
```

```
  STRM_MARIOKART               0     6 127  64   0                    6073912
strm/kart_title.32.pcm8.strm
  STRM_FANFARE                 1     7 127  64   0                     132908
strm/fanfare.32.pcm8.strm
# FAT:
# fileID    offset      size name
      0 0x00000440      6928 mid/kart64_title.sseq
      1 0x00001f60       652 mus/se.ssar
      2 0x00002200       132 bnk/se.sbnk
      3 0x000022a0      1156 bnk/bgm.sbnk
      4 0x00002740     42304 swar/se.swar
      5 0x0000cc80    163684 swar/bgm.swar
      6 0x00034c00   6073912 strm/kart_title.32.pcm8.strm
      7 0x005ffa40    132908 strm/fanfare.32.pcm8.strm
```

Group, sequence, sequence archive, and other information are shown starting at the top of the file. The last item, `FAT`, describes the order of individual files and where the files are stored in the sound data archive.

The sound map file contents can be understood by reading the comments. The content that is difficult to understand are described in the following sections.

11.2.1 Actual Data Size and Consumed Heap Size

The sound map file contains the `size` and `hsize` items. `size` indicates the actual data size, and `hsize` indicates the consumed heap size.

The actual data size is the standalone size of the sound data. It corresponds to the file size.

The consumed heap size is the file size when all the data is loaded onto the heap. For example, because a waveform archive and bank are loaded at the same time, the waveform archive size is added to the bank size.

11.2.2 Group

The first line under `GROUP` indicates group size. The following lines indicate the type of data that makes up the group.

Code 11-2 Group

```
# GROUP:
# label                  number     hsize  type    number     hsize
  GROUP_STATIC                0    208352
                                          SEQARC       0       736
                                          BANK         0     42592
                                          BANK         1    165024
```

In the example above, the group size is 208352 bytes. This group comprises three items and the size of each item disclosed. (Each size is the consumed heap size.)

The total size of all the items in the group may not be equal to the group size. The group size takes duplicate data items and counts them as a single data item. Therefore, the group size is less than the total size of all the items in the group when a common waveform archive is used for each bank.

11.2.3 File ID

`fileID` is an identification number assigned to all sound data and is used inside the sound archive.

If `fileID` is identical for two files, the sound data sets are identical. For example, when separate sequences use the same sequence data, the same `fileID` is retained by each sequence.