Confidential

SMAF Specification

Synthetic music Mobile Application Format

Ver.3.09

Yamaha Corporation

< Revision history >

Ver.	Date	Content
3.00	2001.05.24	Revised and released SMAF v 3.00 specification from SMAF v 2.04. The main changed parts from v 2.04 Defined new Format Type as Track for music, and distinguished from the conventional application. Added the function of single-character color change to Track for display.
3.01	2001.06.05	Added a tag "MI" to Contents Info Chunk and Optional Data Chunk. Optional Data Chunk specification change Addsed Sub-Chunk. Corrected the wrong discription of interpretation of the special value of Effective Span of Banner Info of Graphics Track.
3.02	2001.06.28	Moved to Event of explanation of Huffman coding. Changed about Huffman-coding application of Sequence Data Chunk. Corrected Data Size explanation of Exclusive Messege. Specified the difference between Handy Phone Standard and Mobile Standard in End of Sequence. Added the discription to the Note Messege item about Wave control of Stream PCM Data Chunk. 4.3 Added ES and VC to tag. 6.3.2 Added the discription about the interpretation of coordinate assignment. 6.3.6 1) Corrected wrong discription about Event Type of Bmp Tile. 8.3.1.-6 Deleted a part of explanation of rehearsal mark.
3.05	2002.03.13	Changed this document name into "SMAF Specification" from "SMAF". Changed the expression about simultaneous execution of an event to "successive events which is put duration which value is "0" between them…". Added the Optional Data Chunk to Conceptual diagram of SMAF file. Added the explanatary drawing of Body section. Specified the section of implementation dependence to the relation between Note Message, Start Point and Stop Point. Changed the value of standard type to the short type of Expression. Added Mtsp,Mwa* to Chunk ID & TAG List. 4.2.3 Corrected Korean to EUC-KR(KS) from ISO-2022-KR. 4.2.4 Change to fill in 1 from fill in 0 for unused bit of CopyStatus. 6.1 3) Corrected Korean to EUC-KR(KS) from ISO-2022-KR.
3.06	2002.06.25	Added the TAG to Optional Data Chunk. Addition TAG: IC, IR, IP, TR,TP, GP, VE, CE, UI, OW, A0, A1, A2 4.3 1) Changed Korean to EUC-KR(KS) from ISO-2022-KR.
3.07	2003.03.20	4.4.2.7.a) Reserved number of wave type format was changed. 4.5.3) Reserved number of wave type format was changed. 8.2 Explanation of Duration was changed. 6.1.7) Specification of new tag was added to Graphics Track Header Option. Following clerical errors were corrected 4.2.2 Karaoke: Type 0x42 → 0x43 4.2.6 Option: Description of Class and Type was corrected 6.3.5.2) Offset Origine: Reset → Offset 6.3.6.2) Bitmap Tile was added to Banner Info binary image. 9.3 BNF: Clerical error correction, such as ID. Bmp_Chunk was added.
3.08	2003.10.31	4.2.2 The Contents Type value was changed. SMAF/Phrase was added. 6.1.7) The width of drawing area and hight of drawing area were modified. (error) (0~65535) → (Correct) (-32768~32767) 6.2.1 In Display Parameter Definition Chunk, PrmID was corrected. (error) (0x16~0xFF) → (correct) (0x33~0xFF) 6.3.6.2) In Parameter override, PrmID was corrected. (error) (0x16~0xFF) → (correct) (0x33~0xFF) 9 Chunk ID & TAG were added in the list.
3.09	2004.1.30	4.2.6 A0, A1, A2, and SS tags were added. 4.3.2 1) Tag definitions were added. (SS, LC,RF) 4.3.2 2) Tag contents were added. 6.1 7) GL tag was added.

1. Introduction

SMAF is a data format specification intended to express the multimedia contents efficiently in small size for portable terminals (portable telephones).
The data based on SMAF specification is expressed as multiple bundles of one or more sequence data.
Each sequence data is the data corresponding to the playback device.
All sequence shares a time-axis at the playback. That is, all sequence will play in synchronization.
SMAF is the data format for expressing such synchronous playback contents.

The basic portion of SMAF is designed to express musical data for a Yamaha LSI designed to be the tone generator of a portable terminal device. And SMAF has an extremely flexible structure, and is a specification that provides room for future expansion.
This document is the specification for the SMAF data format, and defines the data format and its basic significance. This document is not intended to be an implementation guide for a system to play back SMAF data, or to define how SMAF data should be created.
Implementation of a SMAF playback system, SMAF data authoring guideline and rules for creating SMAF data must be respectively defined by separate documents.

2. Basic concepts

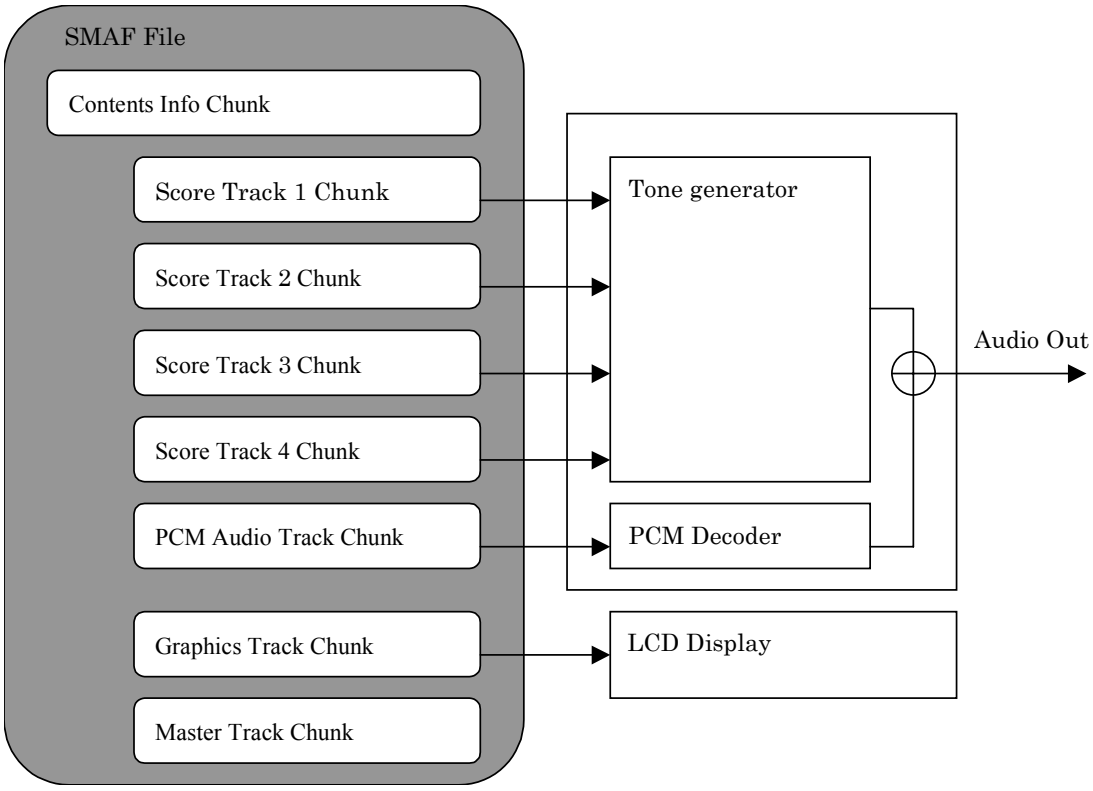
2.1. What the SMAF specification defines

The SMAF specification defines a single-file data format. Files created in compliance with the SMAF specifications are called SMAF files.
The content of an SMAF file can be broadly divided into data for SMAF file management and a collection of sequence data.

The data for SMAF file management is stored in the Contents Info Chunk. Details are explained in the data format definition page.
A collection of sequence data defined for each output device, and includes at least one instance of sequence data. The output devices referred to here are logical devices defined for each instance of sequence data. Normally, corresponding hardware is assumed, but this need not be a one to one correspondence.

Sequence data is a time-stream data format for controlling an output device. All sequence data contained in a single SMAF file is defined as starting playback at time 0. As a result, all sequence data is played back in synchronization.

Output devices defined by the SMAF format include tone generator devices that produce sound in response to MIDI-like control data, PCM tone generator devices that play back PCM data, and display devices that display text or graphics.



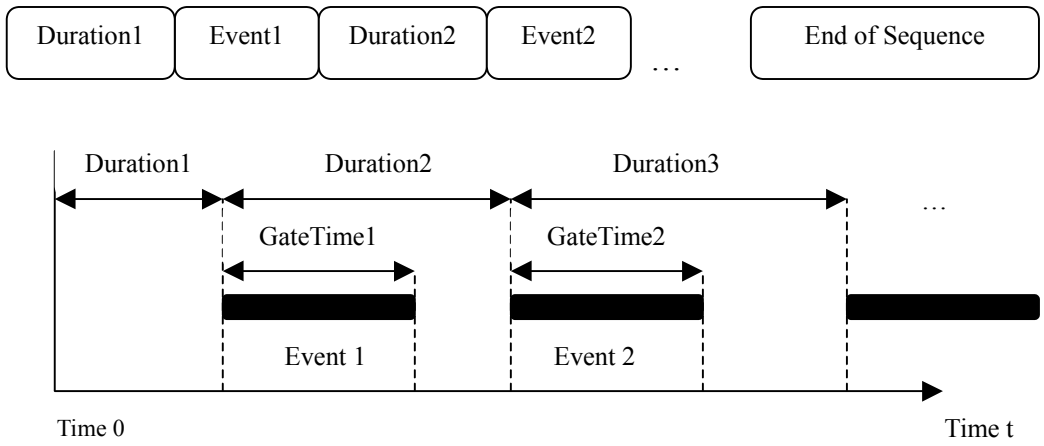
Conceptual diagram of the correspondence between tracks and devices

2.2. Sequence data

Sequence data is expressed as a combination of events and durations.

Events are a form of control data for the output device defined by the sequence data. Durations are the elapsed times between events. In actuality, event processing time is not zero, but for the purposes of SMAF data it is considered to be zero, and all flow of time is expressed as durations. The time at which a certain event is executed can be precisely defined by adding the durations from the beginning of that sequence data. The rule is that event processing time does not affect the time at which processing is begun for the next event. Thus, successive events, which have durations of value "0" between them, are interpreted as executing simultaneously.

Event processing time is always considered to be zero, but in some cases an event will itself contain a sub-sequence. For example, the Gate Time processing of a note event corresponds to this case. The content expressed by a Note event is for Note-on processing to be performed immediately when the event is executed, and Note-off processing to be performed when the Gate Time has elapsed. The Gate Time in this case indicates an elapsed time inside the Note event, and does not affect the execution time of the next event. Some events contain a more complex sub-sequence, but this rule is the same for them as well.



Conceptual diagram of sequence data

2.3. Definition of the time axis

When interpreting sequence data, the flow of time (the time axis) might be defined in one of the following three ways.

A) The time axis used in daily life. It neither expands nor contracts. Units are seconds. Absolute time.

B) The time axis by which a sequencer manages sequence data. By changing the playback speed, it can be expanded or contracted relative to absolute time. Units are seconds. Playback device time.

C) The time axis used mainly to express a musical performance. Corresponds to a musical score,

and can be expanded or contracted relative to playback device time by changing the playback tempo. Units are beats.

All elapsed time in SMAF is expressed as B), playback device time.

Since SMAF is designed mainly for content playback, it has the concept of changing the playback speed, but does not have the concept of changing the tempo. For this reason, the use of time axis concept C) was intentionally avoided.
The length of the time unit is defined for each sequence data, which allows a balance between precision and data size.

2.4. Tracks

Tracks are one of the basic elements in the structure of an SMAF file, and define the sequence data and its operating environment.

The correspondence between tracks and output devices is as follows.

SCORE Track := tone generator device
PCM Track := PCM tone generator device
GRAPHICS Track := display device
MASTER Track := the SMAF sequencer itself

2.5. SMAF playback system implementation guidelines

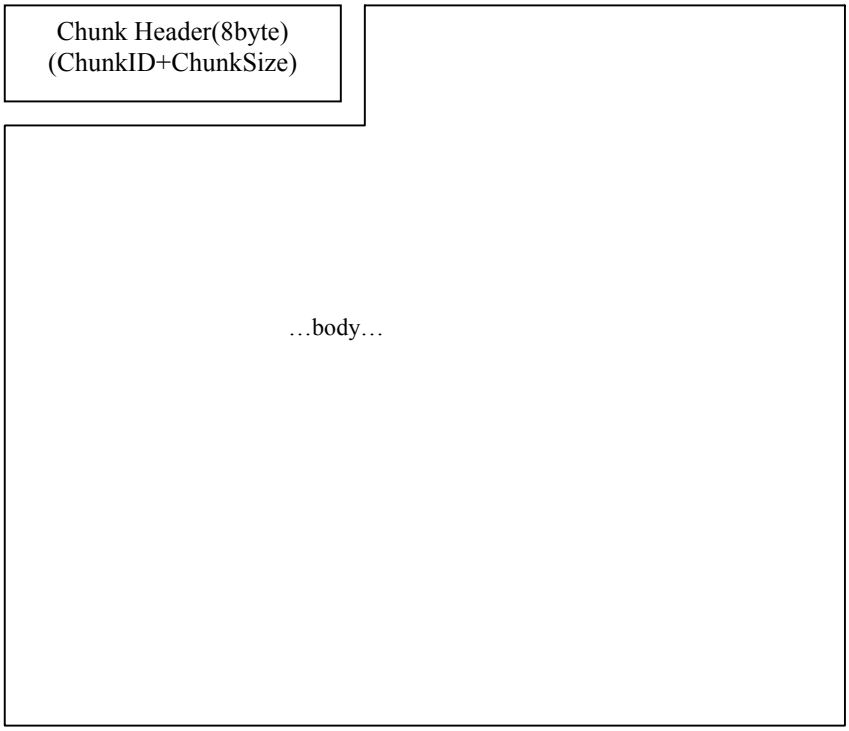
The execution times and processing content of SMAF events are closely defined, and there is virtually no room for ambiguity.
In spite of this, it is impossible for an actual implementation to produce the precise playback expressed in SMAF, and there will always be some discrepancy in execution time or difference in interpretations of events. Also for some events, implementation may not be possible. The permissible range for such discrepancies of execution time, differences in interpretation, and limitations of an implementation is more suitably defined as guidelines for a playback system implementation, and thus will not be discussed in this specification document.

3. Overview of data expressions

3.1. Chunk structure

The basic structure of SMAF data is a collection of data called a Chunk.

The structure of a Chunk is shown below.



The structure of a Chunk

A Chunk is divided into an eight-byte Header and a Body of undefined length. The header is further divided into four-byte Chunk ID and a four-byte Chunk Size.
The chunk ID is used to identify the chunk, and the chunk size indicates the length of the body (in bytes).

3.1.1. Chunk ID

The Chunk ID has a unique value within the file. Normally the ID is four bytes of ASCII text. Case is significant. For example a Chunk ID = "ABCD" would be as shown in the following table.

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	0x41 (“A”)							
Data#1	0x42 (“B”)							
Data#2	0x43 (“C”)							
Data#3	0x44 (“D”)							

In the case of the Track Chunk ID, the first three bytes indicate which track this is, and the last byte indicates the track number as a binary value.

3.2. Chunk Size

The Chunk Size is the byte count for the Body, and consists of four bytes.
However, the byte count for the body of the File Chunk includes a CRC. For example if size = 0x12345678, this would be as shown in the following table.

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	0x12							
Data#1	0x34							
Data#2	0x56							
Data#3	0x78							

3.3. File Chunk

The File Chunk is the chunk that indicates the entire SMAF file. The body of the file chunk is a series of chunks, and a two-byte CRC is appended at the end.
The series of chunks must begin with a Contents Info Chunk, and one or more Track Chunks must follow. The series of chunks must not contain two chunks that have the same chunk ID.

The final CRC is used to check whether the file chunk is damaged.

3.4. Track Chunk

The Track Chunk has four groups; Score, PCM, Graphics, and Master.
Since the lowest byte of the Track Chunk chunk ID is the track number, a maximum of 256 tracks can be expressed.

The currently-defined chunk IDs for the Track Chunk are shown below.

Chunk ID "MTR*" :Score Track	* = 0x00 ~ 0xFF
Chunk ID "ATR*" :PCM Audio Track	* = 0x00 ~ 0xFF
Chunk ID "GTR*" :Graphics Track	* = 0x00 ~ 0xFF
Chunk ID "MSTR" :Master Track	

3.4.1. Types of Track Chunk

- Score Track Chunk
  Contains sequence data for playing a tone generator.

- PCM Audio Track Chunk
  Contains PCM-type audio (e.g., ADPCM, MP3, TwinVQ) in event format.

- Graphics Track Chunk
  Contains background images or still images and text data, and sequence data for playing this.

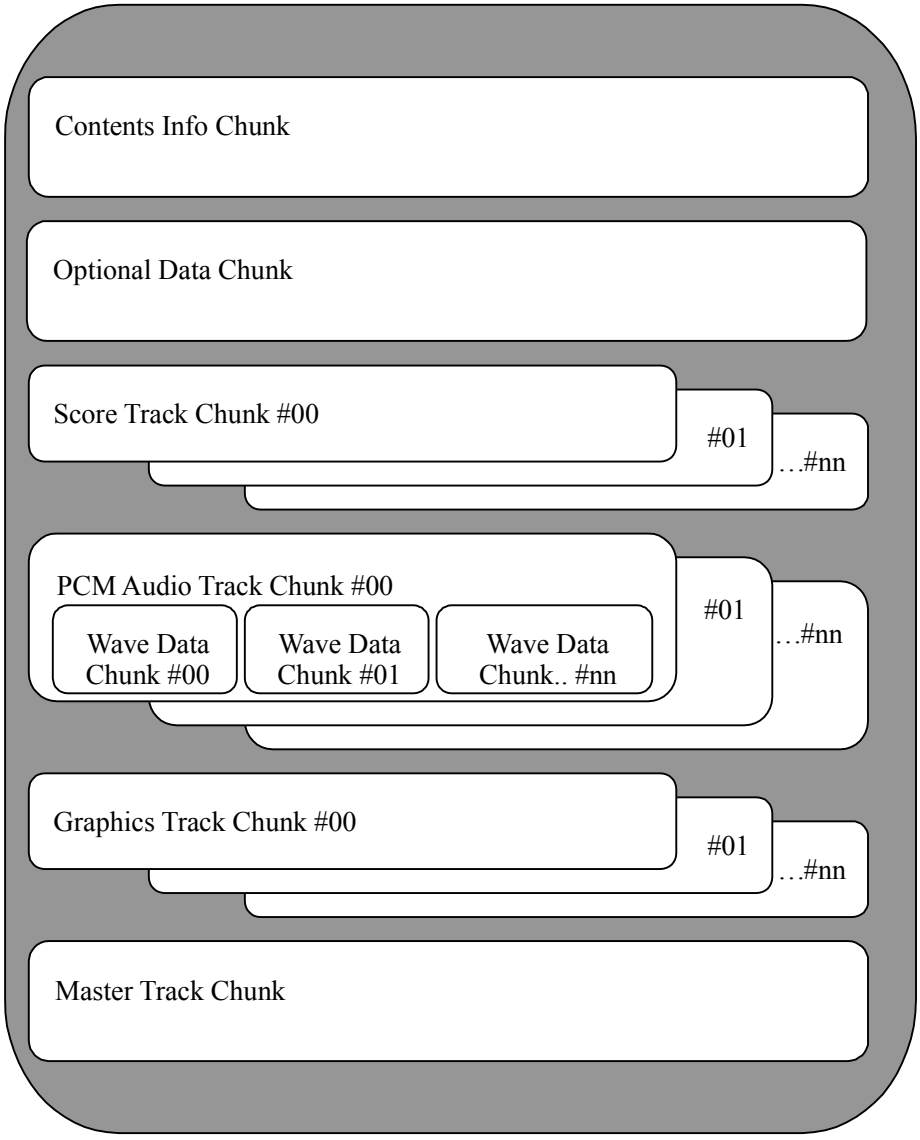
- Master Track Chunk
  Contains sequence data that controls musical data sequences synchronized to the playback sequence (such as the Score Track) or that controls the SMAF playback system.

The track chunk except Master Truck Chunk has a track number within the chunk ID, allowing 256 tracks to be described.
The size of the body can be specified freely. Chunks can be nested by defining an additional chunk within the body. The format of the body is defined for each chunk ID.

The minimum unit of data expressed by SMAF is the byte. In general, the explanations in this document define data structures in byte units. If multiple bytes are used to store integer values, they are Big Endian expressions (upper byte comes first) unless otherwise specified.

3.5. Conceptual diagram of an SMAF file

A conceptual diagram of an SMAF file is shown below.



Conceptual diagram of SMAF file

The SMAF specification does not define the upper limit of the track numbers or the order in which tracks are placed. These matters will be defined separately as SMAF operating rules.

4. Data format (basic functionality)

The basic data format of SMAF is described concretely. "Reserved" in definition item is forbidden to use since the thing is to be defined.

4.1. File Chunk

The File Chunk ID is defined as

Chunk ID	"MMMD"	:Mobile Application Data Chunk

Only in the case of the file chunk, a CRC is added to the end of the body.

4.1.1. CRC

The CRC is the remainder obtained by dividing the following value into the chunk header and body byte string. However, a logical NOT for each bit is used instead of subtraction during division.

Divide value by: 0x11021 (follows CCITT X.25)

(*) Sample code is given in the appendix.

4.2. Contents Info Chunk

Chunk ID	"CNTI"	:Contents	Information	Chunk

The body contains data in the following order.

- Contents Class :1 byte (required)
- Contents Type :1 byte (required)
- Contents Code Type :1 byte (required)
- Copy Status :1 byte (required)
- Copy Counts :1 byte (required)
- Option :n byte (optional)

4.2.1. Contents Class

This indicates the content class.

Contents Class	Description
0x00	YAMAHA
0x01~0xFF	Vender ID

* This is used to make distinctions when the identical data format is used in the future on multi-function terminals such as PDA devices.

4.2.2. Contents Type (for Contents Class 0x00)

Expresses the content type.

Contents Type	Description
0x00~0x0F, 0x30~0x38	Ring melody
0x10~0x1F, 0x40~0x48	Karaoke
0x20~0x2F, 0x50~0x58	Commercials
0xF0	SMAF/Phrase (*)
Unused values except the above	Reserved

(*) The description about SMAF/Phrase is defined in SMAF/Phrase specification.

4.2.3. Contents Code Type

Specifies the character code system used by the data in Option.

Contents Code Type	Description	Language
0x00	Shift-JIS	Japanese
0x01	Latin-1	English, French, German Italian, Spanish, Portuguese
0x02	EUC-KR(KS)	Korean
0x03	HZ-GB-2312	Chinese (simplified)
0x04	Big5	Chinese (traditional)
0x05	KOI8-R	Russian, etc.
0x06	TCVN-5773:1993	Vietnamese
0x07~0x1F	Reserved	Reserved
0x20	USC-2	Unicode
0x21	USC-4	Unicode
0x22	UTF-7	Unicode
0x23	UTF-8	Unicode
0x24	UTF-16	Unicode
0x25	UTF-32	Unicode
0x26~0xFF	Reserved	Reserved

However for any type in Option of Contents Info Chunk, ",(0x2C)" is defined as the Option delimiter, and is therefore used in conjunction with the backslash "\(0x5C)" escape character.

Notation	Function
\,	indicates “,”
\\	indicates \
\	ignored

When set up the character code assumed a delimiter is not correctly discriminable, does not discribe data to Option of Contents Info Chunk, but describes data to Optional Data Chunk.

4.2.4. Copy Status (for Contents Class 0x00, Contents Type 0x00 ~ 0xFF)

Describes the copy definition of the content.

Description	b7	b6	b5	b4	b3	b2	b1	b0
enable/disable bit	Reserved	Reserved	Reserved	Reserved	Reserved	Edit	Save	Transmission

b0 is transmission enable/disable, b1 is save enable/disable, b2 is edit enable/disable. (0: enable, 1: disable)

Reserved bit is filled with “1”.

4.2.5. Copy Counts(for Contents Class 0x00, Contents Type 0x00 ~ 0xFF)

Express the number of copies.

Copy Counts	Description
0x00~0xFE	Copy Counts
0xFF	Copy Counts (255 or more)

* Incremented each time a copy or move occurs.

4.2.6. Option (for Contents Class 0x00, Contents Type 0x00 ~ 0xFF)

This stores the genre name, song name, artist name, and lyricist/composer names. This is not necessarily used for display, but rather to identify the data.

The data length is variable, and is delimited by "tag (2 byte)" + " : (0x3A)" + "data" + " , (0x2C)".

Tag names are given below. The tag is 2-byte fixation.

The escape character for when " , (0x2C)" is used in "Data" is defined in §4.2.3.

Tag name	Title	Hex
VN	Vendor name	0x56 0x4E
CN	Carrier name	0x43 0x4E
CA	Category name	0x43 0x41
ST	Song title	0x53 0x54
AN	Artist name	0x41 0x4E
WW	Lyricist	0x57 0x57
SW	Composer	0x53 0x57
AW	Arranger	0x41 0x57
CR	Copyright©	0x43 0x52
GR	Management group	0x47 0x52
MI	Manegement information	0x4D 0x49
CD	Creation date	0x43 0x44
UD	Revision date	0x55 0x44
A0	Reserved	0x41 0x30
A1	Reserved	0x41 0x31
A2	Reserved	0x41 0x32
SS	Snap shot	0x53 0x53

According to the addition of Unicode, Optional Data Chunk was added since the character code which cannot discriminate the delimiter in Option existed.

4.3. Optional Data Chunk

Chunk ID "OPDA" : Optional Data Chunk

This stores the genre name, song name, artist name, and lyricist/composer names. This is not necessarily used for display, but rather to identify the data. In the Body section of Optional Data Chunk, the Sub Chunk sequence like Data Chunk etc.is stored.

1.) Data Chunk

Chunk ID "Dch*" : Data Chunk * = 0x00 ~ 0xFF

The 4th byte of Chunk ID of Data Chunk chooses the value of Code Type described below according to Data to store. Storing Data shall be shown by the character code system which is specified by Code Type.

{

- ■ Tag : 2 byte (fixed)
- ■ Size : 2 byte (fixed)
- ■ Data : n byte (variable)

}

The data length is variable, and the data is described "Tag (2 byte fixed)" + "Size (2 byte fixed)" + "Data (variable)" as a couple.

Specifies the character code system used by the data in Optional Data Chunk.

Contents Code Type	Description	Language
0x00	Shift-JIS	Japanese
0x01	Latin-1	English, French, German Italian, Spanish, Portuguese
0x02	EUC-KR(KS)	Korean
0x03	HZ-GB-2312	Chinese (simplified)
0x04	Big5	Chinese (traditional)
0x05	KOI8-R	Russian, etc.
0x06	TCVN-5773:1993	Vietnamese
0x07~0x1F	Reserved	Reserved
0x20	USC-2	Unicode
0x21	USC-4	Unicode
0x22	UTF-7	Unicode
0x23	UTF-8	Unicode
0x24	UTF-16	Unicode
0x25	UTF-32	Unicode
0x26~0xFF	Reserved	Reserved
0xFF	Octet Stream	Binary value

When the character code to describe is Unicode, set BOM (byte-order mark) as each letter group head. When no BOM, it interpreted as big endian.

Tag is defined as follows.

Tag name	Title	Hex
VN	Vendor name	0x56 0x4E
CN	Carrier name	0x43 0x4E
CA	Category name	0x43 0x41
ST	Song title	0x53 0x54
AN	Artist name	0x41 0x4E
WW	Lyricist	0x57 0x57
SW	Composer	0x53 0x57
AW	Arranger	0x41 0x57
CR	Copyright©	0x43 0x52
GR	Management group	0x47 0x52
MI	Manegement information	0x4D 0x49
CD	Creation date	0x43 0x44
UD	Revision date	0x55 0x44
ES	Status after edit	0x45 0x53
VC	Virtial card	0x56 0x43
IC	Image creater	0x49 0x43
IR	Image copyright	0x49 0x52
IP	Image editor	0x49 0x50
TR	Text copyright	0x54 0x52
TP	Text editor	0x54 0x50
GP	Contents editor	0x47 0x50
VE	Reserved	0x56 0x45
CE	Reserved	0x43 0x45
UI	Reserved	0x55 0x49
OW	Reserved	0x4F 0x57
A0	Reserved	0x41 0x30
A1	Reserved	0x41 0x31
A2	Reserved	0x41 0x32
SS	Snap shot	0x53 0x53
LC	Locale infomation	0x4C 0x43
RF	Resource infomation	0x52 0x46

2.) Supplement of Tag Data

- LC tag

  When OPDA Data Chunk is described in Unicode, the information of language and assumed service area about character sequence data written in this chunk is described here.

  LC tag Data : 2 byte (Option)

  <Data> = <Language> < Carrier etc.>
  BYTE BYTE

Language	Description
0x00	Japanese
0x01	Latin-1
0x02	Korean
0x03	Chinese (simplified)
0x04~0xFE	Reserved
0xFF	ASCII

Carrier etc.	Description
0x00	Generic (Global)
0x01	Reserved
0x02~0xFF	Reserved

LC tag is described as Octet Stream, when creating Unicode contents.
Only OPDA Chunk is the object of this tag.

- RF tag

RF tag enumerates playable things in the sequence and resource data which exist in the SMAF file.

RF tag Data : 3 byte (Option)

<Data> = <MTR> <GTR><MSTR, others >
BYTE BYTE BYTE

1st BYTE
<MTR> 0bit : FM/PCM seq. Presence: 1, Absence: 0
1bit : Stream seq. Presence: 1, Absence: 0
2bit : 0

2nd BYTE
<GTR> 0bit : GTR0 seq. Presence: 1, Absence: 0

3rd BYTE
<MSTR> 0bit : MSTR seq. Presence: 1, Absence: 0

Other bits are taken as Reserved. Reserved bit is filled with 0.

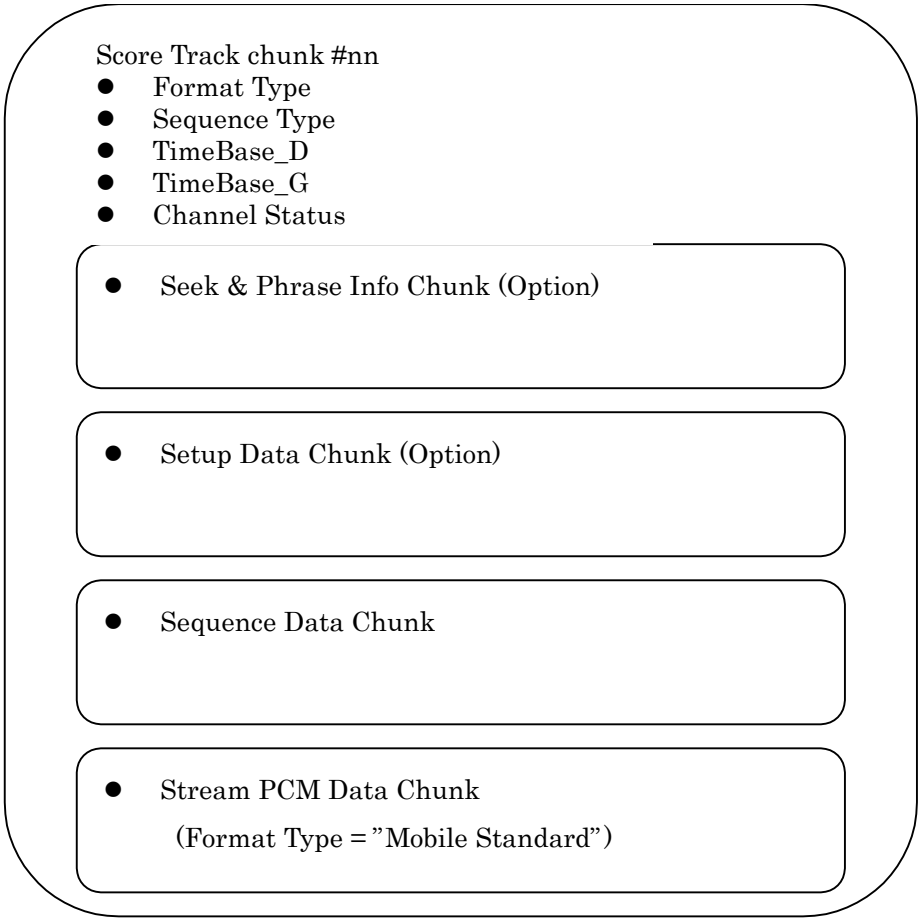
RF tag is described as Octet Stream, when creating SMAF contents.
Only OPDA Chunk is the object of this tag.

4.4. Score Track Chunk

Chunk ID "MTR"	: Score Track Chunk

This chunk contains the musical sequence tracks that are sent to the tone generator. It consists of one byte of data to distinguish the format, track-dependant information, and three sub-chunks. Only one instance of each sub-chunk may exist in a score track chunk.

- Format Type :1 byte (required)
- Sequence Type :1 byte (required)
- TimeBase_D :1 byte (required)
- TimeBase_G :1 byte (required)
- Channel Status :n byte (required)(depends on Format Type)
- Seek & Phrase Info Chunk :n byte (optional)
- Setup Data Chunk :n byte (optional)
- Sequence Data Chunk :n byte (required)
- Stream PCM Data Chunk :n byte (optional)(In the case of Format Type="Mobile Standard")



Format Type

This status defines the actual format of this track chunk. To minimize the data size, we allow for other sequence formants, such as data in the native format of a LSI, or powerful control CPU's of the future. Compress determines that compression by Huffman-coding-izing is performed.
(4.4.2 Mobile Standard Format explains Compress.)

Format Type	Description
0x00	Handy Phone Standard
0x01	Moble Standard(Compress)
0x02	Moble Standard(No Compress)
0x03~0xFF	Reserved

4.4.1. Handy Phone Standard Format(Format Type=0x00)

This scalable data format supports several generations of Handy Phone. Data production is limited (as specified in the production guidelines) by the processing capabilities of the implementation. Similarly, implementation guidelines are also required for the implementation, and these two sets of guidelines and this format comprise the three legs of a tripod required for guaranteeing actual operation.

1.) Sequence Type

Two types of sequence data listing are considered.

- 0x00 :Stream Sequence
  The sequence data is in one continuous stream. Seek Points and Phrase Lists are used to allow external references to meaningful points within the sequence.
- 0x01 :Sub-sequence
  The sequence data is given as a succession of two or more instances of phrase data. Phrase Lists are used to identify individual phrases from outside.
- 0x02 - 0xFF(reserved)

2.) TimeBase_D, TimeBase_G

This specifies the time base used within the sequence data.
TimeBase_D is the time base used for Duration, and TimeBase_G is the time base for GateTime.

Timebase_D,G	Description
0x00	1 msec
0x01	2 msec
0x02	4 msec
0x03	5 msec
0x04 - 0x0F	Reserved
0x10	10 msec
0x11	20 msec
0x12	40 msec
0x13	50 msec
0x14~0xFF	Reserved

3.) Channel Status

This contains status information for each of the four Channels of sequence data. Four bits

contains the status information for one Channel, and the Channels are defined as shown in the following table.

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	Channel 0				Channel 1			
	KCS	VS	Ch Type		KCS	VS	Ch Type	
Data#1	Channel 2				Channel 3			
	KCS	VS	Ch Type		KCS	VS	Ch Type	

- Key Control Status (KCS)

  Specify whether Key Control will be performed for the corresponding channel when a Key Control request is received. If this is ON, Key Control will be enabled.

KCS	Description
0x0	OFF
0x1	ON

- Vibration Status (VS)

  Specify whether Vibration will be performed for the corresponding channel when a Vibration Control request is received. If this is ON, Vibration will be enabled.

VS	Description
0x0	OFF
0x1	ON

- Ch Type

  Specify the Channel Type for the corresponding channel.

Ch Type	Description
0x0	No Care
0x1	Melody
0x2	No Melody
0x3	Rhythm

4.) Seek & Phrase Info Chunk (optional)

This contains information that allows starting or stopping at a desired location in the sequence data, or allows regions to be repeated. Use and functionality are optional, and depends on the model and on the intentions when creating data.

Chunk ID	"MspI"	:Seek & Phrase Info Chunk

The body contains data in the following order.

- Start Point :1 or 4 byte (optional)
- Stop Point :1 or 4 byte (optional)
- Phrase List :n byte (optional)
- Sub-sequence List :n byte (optional)

As long as this chunk exists, one of the data items must exist. The above data sizes do not include the tag, colon, or comma. Since each data is variable in length, they are delimited by "tag (2 bytes)" + ": (0x3A)" +"Data"+ ", (0x2C)".

a) Start / Stop Point

These specify a Start or Stop point at the desired location within sequence data or a sub-sequence list. The tags are shown below, and are in ASCII characters.

Title	Tag name	Hex
Start	st	0x73 0x74
Stop	sp	0x73 0x70

- If the Sequence Type = 0x00(Stream Sequence)

  Start Address : 4 byte
  Stop Address : 4 byte

  is listed following the tag.

  Each addressing sets the head of Body section of Sequence Data chunk to "0", and describes by Offset from there.

  Body

0	1	2	・・・

  Sequence Data Chunk. Also, the addressing value must be a value that points to the beginning of the Duration.

- If the Sequence Type = 0x01(Sub-sequence)

  Start Address : 1 byte
  Stop Address : 1 byte

  And describes after TAG.

  Each addressing is the byte count number from Sub-sequence List head.

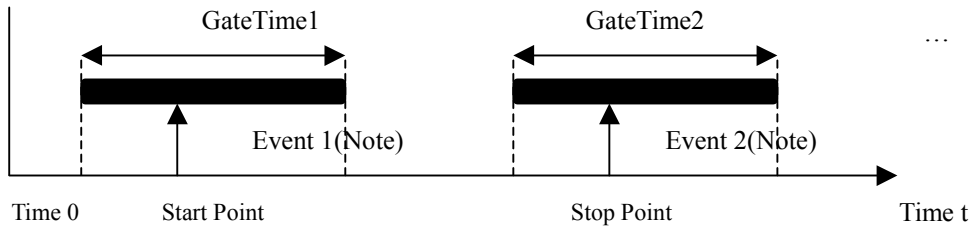
  If omitted, it considers as Start = 0, Stop = "File End" or "Sub-sequence List End".

(*) The relation between Note Message, Start Point, and Stop Point

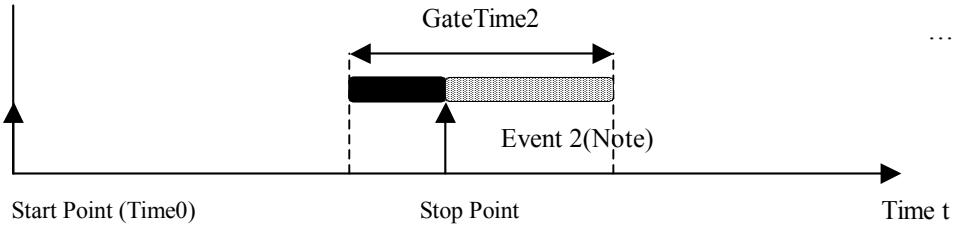
The start time of the Note Message is compared with the Start Point and Stop Point times.
For the Start Point, if
(Start Point time) ≤ (start time of Note Message)
then the corresponding Note Message will be sounded.
For the Stop Point, if
(start time of Note Message) < (Stop Point time)
then the corresponding Note Message will be sounded.
(The play guarantee of gatetime is considered as implementation dependence.)

When looping between the Start Point and Stop Point, you must guarantee that the times of the Stop Point being looped and the next Start Point are identical.

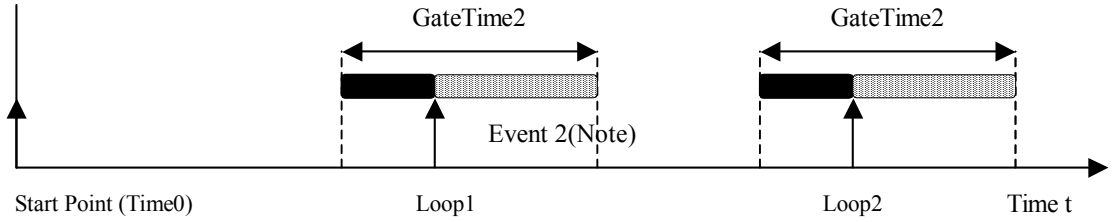
As an example, let us consider the case of the following data in a Sequence Data Chunk.



When playing from Start to Stop, interpretation of the Stream Sequence will be as follows.
<Single Play>



<Loop play>



loop1 = Stop Point – Start Point
loop2 =loop1 + (Stop Point – Start Point)

Play guarantee of here depends on implementation.

b) Phrase List

Specify the desired location (region) in the sequence data as a phrase region. Usage differs, depending on the Sequence Type.

- If Sequence Type = 0x00 (Stream Sequence)
  Used to assign significance's to portions (regions) of the sequence data such as melody-A, melody-B, bridge, etc.). The way in which this is used is up to the application.
- If Sequence Type = 0x01(Sub-sequence)
  Specifies a phrase region used in the "sub-sequence list" (discussed later).

The data is expressed by following the tag name with

Start Address : 4 bytes
Stop Address : 4 bytes

Each address is specified as a byte count from the beginning of the body of the Sequence Data Chunk.

The Start Address and Stop Address of each region can be set freely.
The tag names indicating each phrase region are given below. Tags are ASCII characters.

Title	Tag name	Hex
a	Pa	0x50 0x61
b	Pb	0x50 0x62
c	Pc	0x50 0x63
d	Pd	0x50 0x64
e	Pe	0x50 0x65
:	:	:
z	Pz	0x50 0x7A
A (melody A)	PA	0x50 0x41
B (melody B)	PB	0x50 0x42
E (ending)	PE	0x50 0x45
I (intro)	PI	0x50 0x49
K (break)	PK	0x50 0x4B
S (bridge)	PS	0x50 0x53
R (refrain)	PR	0x50 0x52

c) Sub-sequence List

If the Sequence Type is 0x01 (Sub-sequence), this lists the order of the phrase regions specified in the Phrase List. If the Sequence Type is 0x00(Stream Sequence), this is ignored.

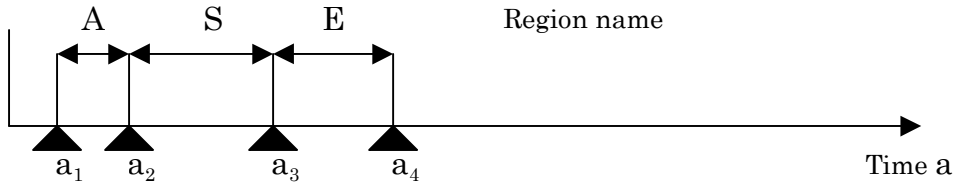
The region names (a--z, A, B, E, I, K, S) are ASCII code, and are listed sequentially in byte units. The tag is shown below, and is ASCII text.

Title	Tag name	Hex
Sub-sequence List	SL	0x53 0x4C

(*) Interpreting the Subsequence List

An example of using a Stream Sequence to express a Subsequence List is shown in the following diagram.

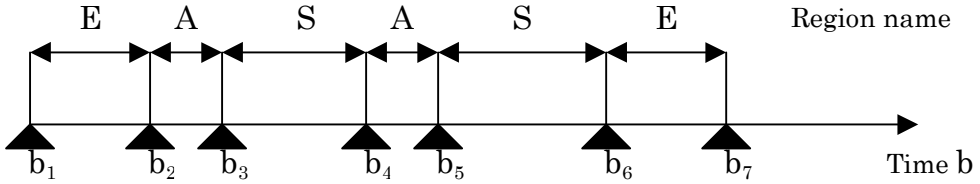
Example 1)



Region names A, S, and E are respectively defined as shown above, and if the sub-sequence list is

Subsequence List = " SL : E A S A S E "

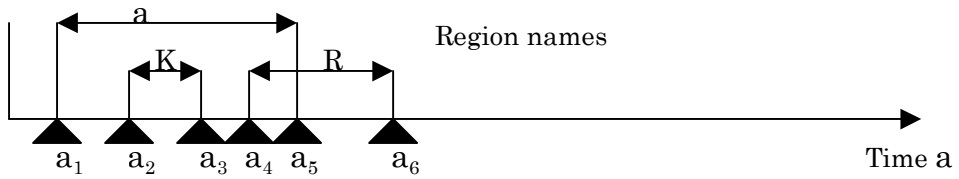
then the interpretation as a Stream Sequence would be as shown in the following diagram.



Thus we have

$b_1 = 0$, $b_2 = a_4 - a_3$, $b_3 = b_2 + (a_2 - a_1)$, $b_4 = b_3 + (a_3 - a_2)$,
$b_5 = b_4 + (a_2 - a_1)$, $b_6 = b_5 + (a_3 - a_2)$, $b_7 = b_6 + (a_4 - a_3)$

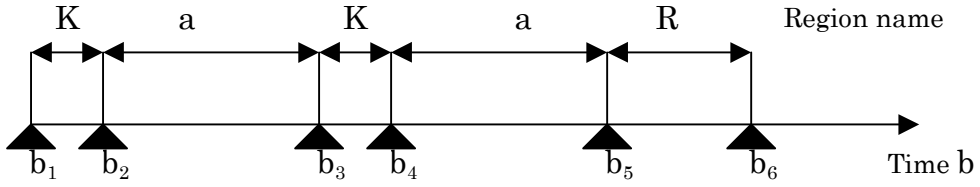
Example 2)



Region names a, K, and R are respectively defined as shown in the above diagram. For a sub-sequence list of

Subsequence List = " SL : K a K a R "

the interpretation as a Stream Sequence would be as shown in the following diagram.



Thus we have

$b_1 = 0$, $b_2 = a_3 - a_2$, $b_3 = b_2 + (a_5 - a_1)$, $b_4 = b_3 + (a_3 - a_2)$,
$b_5 = b_4 + (a_5 - a_1)$, $b_6 = b_5 + (a_6 - a_4)$

(*) The relation between Note Message and the Start Point and Stop Point of a sub-sequence

This is the same as for a Start Point and Stop Point.

If the gatetime of a Note Message extends beyond the Stop Point of the last phrase in the Stream Sequence, the sound will continue until the end of the gatetime.

5.) Setup Data Chunk

This contains sound data and effect settings for the tone generator. In most cases, this data can also be listed in the Sequence Data Chunk, but for convenient data bundling it can be placed in a separate Data Chunk.

Chunk ID "Mtsu" :Score Track Setup Data Chunk

The body contains a succession of exclusive messages used in the Sequence Data Chunk.

Setup Data Chunk body = (Exclusive Message)+

This does not include the duration defined in the Sequence Data Chunk. In the idealized model, this setup time is zero. Message spacing and interpretation time at the time of setup depends on the system, and is a question of implementation.

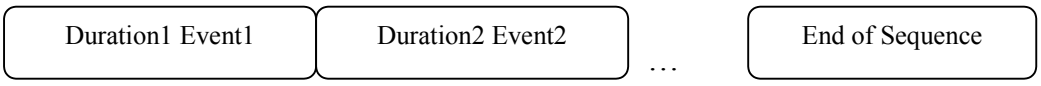
6.) Sequence Data Chunk

Contains the actual playback data. This Sequence Data Chunk is required as long as there is a Track Chunk.

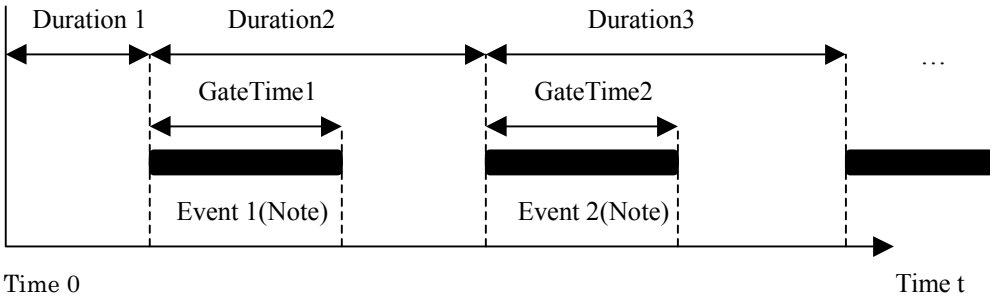
Chunk ID "Mtsq" :Score Track Sequence Data Chunk

The Sequence Data Chunk consists of sequential data pairs of Duration and Event, plus an End of Sequence.

Sequence Data Chunk body = ((Duration Event) | (End of Sequence))+



The duration is the time from the start time of the preceding event to the start time of the paired event. If, for example, the event is a Note Message, the relation between Duration and the GateTime (see (3) item b)) of the Note Message will be as shown in the following diagram. If the event is at the identical time, the duration is 0 (0x00).



One Sequence Data Chunk can have up to four channels. Five or more channels can be expressed by using two or more Sequence Data Chunks in the Score Track Chunk.
This format does not define whether each channel is monophonic or polyphonic. For the way in which the system will interpret and process, refer to the separate data production guidelines and the implementation guidelines.

a) End of Sequence(EOS)

Four consecutive \x00 indicate End of Sequence. EOS is not absolutely necessary, but its inclusion is desirable in order to explicitly indicate the end of the sequence.

End of Sequence = \x00 \x00 \x00 \x00

Since EOS indicates the end, it is desirable for the system implementation to process the sequence in time-order, and then when EOS is detected, terminate the sequence and mute the sound. Even if a Sub-sequence List is used, it is expanded into a Stream Sequence and processed in order of time, and when EOS is detected it is considered to be the end of the sequence. If EOS does not exist, the ending location is calculated from the Chunk Size.

b) Duration

This is expressed by 1 byte or 2 bytes. If the MSB of the 1st byte is 0, then the data is 1 byte. If the MSB of the 1st byte is 1, then the data is 2 bytes. If the 2nd byte exists, its MSB is always zero.

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	S	Data1						
Data#1	0	Data2						

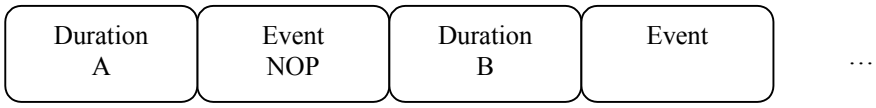
Depending on the MSB(=S) of the 1st byte, the range of values is as shown below.

S	Step
0x0	0~127
0x1	128~16511

If S is 1, the expressed value is 128 plus the 14 bit data consisting of data1 as the MSB and data2 as the LSB.

The resolution of one step is specified by Timebase_D in the Score Track Chunk. You may also use NOP Events to divide one duration into two or more. In this case, the sum of the two divided durations shall be equal to the original duration. Use this method of the number of steps exceeds 16,511.

Ex) If the Duration is 16511 + α



A + B = 16511 + α

c) Event

Events are defined below. The frequently-used events Expression/ Pitch Bend/ Modulation have normal types and short types, and the data size can be reduced by using these two types appropriately.

- Note Message

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	Channel		Octave		Note Number			
Data #1	Gatetime (1st byte)							
Data #2	Gatetime (2nd byte) [optional]							

This message plays sound on the specified channel. Gatetime is expressed in the same way as Duration. However, a GateTime of 0 is prohibited.

Channel	Description
0x0	Channel #0
0x1	Channel #1
0x2	Channel #2
0x3	Channel #3

Octave	Description
0x0	Low
0x1	Mid Low
0x2	Mid High
0x3	High

A key scale of up to four octaves can be expressed simultaneously.
Pitches that exceed this range can be expressed using the Octave Shift control change.

Voice Number	Description
0x0	Prohibited
0x1	C#
0x2	D
0x3	D#
0x4	E
0x5	F
0x6	F#
0x7	G
0x8	G#
0x9	A
0xA	A#
0xB	B
0xC	C
0xD - 0xF	Prohibited

Mid High A is 440 (Hz).

- Program change

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x0			
Data #2	Value							

Specify the voice for the corresponding channel. The correspondence between voice number and voice is defined by the Standard Voice Map in the appendix. (Banks only 0x00 and 0x80)

Value	Description
0x00-0x7F	Program Change Number
0x80-0xFF	Reserved

If this message is not specified, the voice number of the corresponding channel will be 0x00.

- Bank Select

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x1			
Data #2	Value							

Specify the bank of the corresponding channel. Use this message to specify the bank, and then use a Program Change to specify the voice of the corresponding bank. Only banks 0x00 and 0x80 are defined by the Standard Voice Map in the appendix.

Value	Description
0x00-0x7F	Bank Number(Normal)
0x80-0xFF	Bank Number(Drum)

If this message is not specified, the bank number for the corresponding channel will be 0x00.

- Octave Shift

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x2			
Data #2	Value							

Specify the octave shift amount for the corresponding channel. The value that specifies the amount of shift is an absolute value. (It is not added to the previous shift value.)

Value	Description
0x00	No Shift(Original)
0x01	+1 Octave
0x02	+2 Octave
0x03	+3 Octave
0x04	+4 Octave
0x05~0x80	Reserved
0x81	-1 Octave
0x82	-2 Octave
0x83	-3 Octave
0x84	-4 Octave
0x85~0xFF	Reserved

- Modulation (normal type)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x3			
Data #2	Value							

This modifies the vibrato depth of the corresponding channel.

Value	Description
0x00~0x7F	Data(default 0x00)
0x80~0xFF	Reserved

- Modulation (short type)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	0	Value			

This modifies the vibrato depth of the corresponding channel.

Value	Description
0x1~0xE	Data(default 0x1)
0x0,0xF	Reserved

The relation of Data of Normal type and Short type in Modulation is shown in the following table.

Short	Normal
0x1	0x00
0x2	0x08
0x3	0x10
0x4	0x18
0x5	0x20
0x6	0x28
0x7	0x30

Short	Normal
0x8	0x38
0x9	0x40
0xA	0x48
0xB	0x50
0xC	0x60
0xD	0x70
0xE	0x7F

- Pitch Bend (normal type)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel			1	1	0x4		
Data #2	Value							

Raise or lower the pitch of the corresponding channel.

Value	Description
0x00-0x7F	Data(default 0x40)
0x80-0xFF	Reserved

- Pitch Bend (short type)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel			0	1	Value		

Raise or lower the pitch of the corresponding channel.

Value	Description
0x1-0xE	Data(default 0x8)
0x0,0xF	Reserved

The relation of Data of Normal type and Short type in Pitch Bend is shown in following table.

Short	Normal
0x1	0x08
0x2	0x10
0x3	0x18
0x4	0x20
0x5	0x28
0x6	0x30
0x7	0x38

Short	Normal
0x8	0x40
0x9	0x48
0xA	0x50
0xB	0x58
0xC	0x60
0xD	0x68
0xE	0x70

- Volume

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x7			
Data #2	Value							

Adjust the volume of the corresponding channel.

Value	Description
0x00~0x7F	Data
0x80~0xFF	Reserved

The recommended equation is given below.
Gain[dB] = MUTE , Data = 0
Gain[dB] = $20*\log(Data^2/127^2)$, Data > 0

- Pan

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0xA			
Data #2	Value							

Specify the stereo position of the corresponding channel. The sound of the channel will be panned to the specified position between L (0x01) and R (0x7F). 0x00 is L.

Value	Description
0x00~0x7F	Data(default 0x40)
0x80~0xFF	Reserved

The recommended equation is given below.
Left Channel Gain[dB] = $20*\log(\cos(\pi/2*Data/127))$
Right Channel Gain[dB] = $20*\log(\sin(\pi/2*Data/127))$

- Expression (normal type)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel			1	1	0xB		
Data #2	Value							

Adjust the volume of the corresponding channel during the song, relative to the Volume setting.

Value	Description
0x00~0x7F	Data(default 0x7F)
0x7F~0xFF	Reserved

The recommended equation is given below.
Gain[dB] = MUTE , Data = 0
Gain[dB] = $20*\log(Data^2/127^2)$, Data > 0

- Expression(short type)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel			0	0	Value		

Adjust the volume of the corresponding channel during the song.

Value	Description
0x1~0xE	Data(default 0xE)
0x0,0xF	Reserved

In Expression, the Data 0x1-0xE of Short type corresponds to Data 0x00-0x7F of Normal type as shown in the following table.

Short	Normal
0x1	0x00
0x2	0x1F
0x3	0x27
0x4	0x2F
0x5	0x37
0x6	0x3F
0x7	0x47

Short	Normal
0x8	0x4F
0x9	0x57
0xA	0x5F
0xB	0x67
0xC	0x6F
0xD	0x77
0xE	0x7F

- Exclusive Message

This message allows settings to be made for various parameters specific to a device. This message is variable length. Data format is shown below.

	b7	b6	b5	b4	b3	b2	b1	b0
Status	1	1	1	1	1	1	1	1
Data #0	1	1	1	1	0	0	0	0
Data #1	Message Size							
Data #2	Maker ID							
Data #3	Format ID							
:	:							
Data End	1	1	1	1	0	1	1	1

Message Size is the byte count from Data_#2 to Data_End.
According to the Maker ID, the manufacturer-specific data format is defined for Data_#3 and following.
According to the Format ID, the model- (device-) specific data format for the corresponding manufacturer is defined for Data_#4 and following.

- NOP

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0xFF							
Data #1	0x00							

This performs no operation.
Data #1values other than 0x00 and 0xF0 are reserved.

4.4.2. Mobile Standard Format (Format Type = 0x01~02)

The data format is possible to support to the expression of near MIDI data.

About the variable length notation

It is expressed by the variable length of 1 - 4 byte. When MSB of the 1st-3rd byte are 0, it is shown that data do not exist henceforth, and when MSB is 1, it is shown that data exist henceforth. MSB of the 4th byte is always zero.

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	S	Data0						
Data#1	S	Data1						
Data#2	S	Data2						
Data#3	0	Data3						

actual value (Hex)	Expression by variable length (Hex)
00000000	00
00000040	40
0000007F	7F
00000080	81 00
00002000	C0 00
00003FFF	FF 7F
00004000	81 80 00
00100000	C0 80 00
001FFFFF	FF FF 7F
00200000	81 80 80 00
08000000	C0 80 80 00
0FFFFFFF	FF FF FF 7F

Table Example of the notation of variable format

1.) Sequence Type

Same as Handy Phone Standard Format.

2.) TimeBase_D, TimeBase_G

Same as Handy Phone Standard Format.

3.) Channel Status

The status information on each Channel of Sequence data 16 Channels is stored.

Count	Channel	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	1	KCS		VS	LED	Reserved		Ch Type	
Data #1	2	KCS		VS	LED	Reserved		Ch Type	
Data #2	3	KCS		VS	LED	Reserved		Ch Type	
Data #3	4	KCS		VS	LED	Reserved		Ch Type	
Data #4	5	KCS		VS	LED	Reserved		Ch Type	
Data #5	6	KCS		VS	LED	Reserved		Ch Type	
Data #6	7	KCS		VS	LED	Reserved		Ch Type	
Data #7	8	KCS		VS	LED	Reserved		Ch Type	
Data #8	9	KCS		VS	LED	Reserved		Ch Type	
Data #9	10	KCS		VS	LED	Reserved		Ch Type	
Data #10	11	KCS		VS	LED	Reserved		Ch Type	
Data #11	12	KCS		VS	LED	Reserved		Ch Type	
Data #12	13	KCS		VS	LED	Reserved		Ch Type	
Data #13	14	KCS		VS	LED	Reserved		Ch Type	
Data #14	15	KCS		VS	LED	Reserved		Ch Type	
Data #15	16	KCS		VS	LED	Reserved		Ch Type	

- Key Control Status(KCS)

When the request of Key Control is received, specifies whether Key Control is performed for the corresponding Channel.Key Control becomes effective by ON.

KCS	Description
0x0	Nondesignation
0x1	OFF
0x2	ON
0x3	Reserved

The interpretation with nondesignation depends on the implementation of handset.

- Vibration Status(VS)

  When the request of Vibration Control is received, specifies whether Vibration is performed synchronizing with the data of the corresponding Channel. Vibration becomes effective by ON.

VS	Description
0x0	OFF
0x1	ON

- LED Status(LED)

  When the request of LED Control is received, specifies whether LED is controlled synchronizing with the data of the corresponding Channel. LED becomes effective by ON.

LED	Description
0x0	OFF
0x1	ON

- Ch Type

  Channel Type is specified to the corresponding Channel.

Ch Type	Description
0x0	No Care
0x1	Melody
0x2	No Melody
0x3	Rhythm

4.) Seek & Phrase Info Chunk

Same as Handy Phone Standard Format.

5.) Setup Data Chunk

Chunk ID "Mtsu" : Score Track Setup Data Chunk

Setup Data Chunk consists of Exclusive Event rows.

Setup Data Chunk body = (Exclusive Event)+

<Exclusive Event> = F0 <Length> <Data Bytes> F7

Length is regarded as variable format. Moreover, F7 of the last is included in Length.

Exclusive Event which is not specified by this specification needs to be disregarded when creating the program which interprets this format.

6.) Sequence Data Chunk

Chunk ID "Mtsq" : Score Track Sequence Data Chunk

The Body of Sequence Data Chunk consists of data sequence which made Duration and Event the pair. Duration means as the time to the start time of Event which it has as a pair from the start time of Event in front of one. Refer to the c) for the format of Huffman code. It is arbitrary whether compression by Huffman coding is performed. However, sets as follows by Format Type.

When Format Type is 0x01 (Compress),
Sequence Data Chunk body = Huffman(((Duration Event) | (End of Sequence))+)

When Format Type is 0x02(No Compress),
Sequence Data Chunk body = ((Duration Event) | (End of Sequence))+

a) Duration

It is described by the variable length notation.

b) Event

The status byte is placed in the head byte of each event and shows the case of MSB="1".

Status Byte	Description
0x80~8F	Note Messege (No Velocity)
0x90~9F	Note Messege (With Velocity)
0xA0~AF	Reserved (3byte including Status Byte)
0xB0~BF	Control Change
0xC0~CF	Program Change
0xD0~DF	Reserved (2byte including Status Byte)
0xE0~EF	Pitch Bend
0xF0	System Exclusive
0xF1~FE	Reserved
0xFF	EOS or NOP

- Note Message

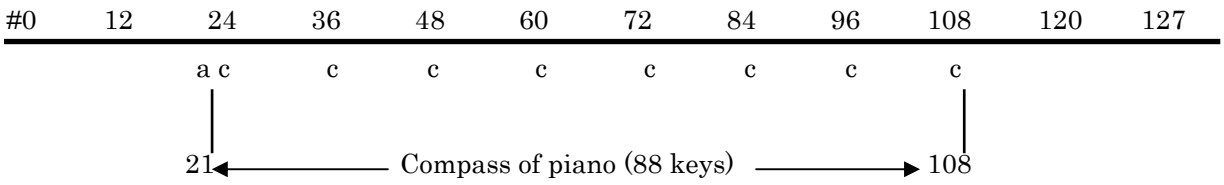
No Velocity

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	1	0	0	0	Channel			
Data #1	0	Note Number						
Data#2-5	S	Gatetime(Variable)						

Sound is pronounced by the key of Note Number for the relevent Channel. Velocity is set to 64 by default. Gatetime is described by the variable length notation.

Memorizes the value of velocity per Channel when pronouncing by Note Message (With velocity), and pronounces with the value of memorized velocity by the pronunciation of Note Message (No velocity). This storage is cleared by reset all control (it becomes a default value).

Channel is assigned to 1-16 channel (0-15) and each key is assigned as Note Number, and interprets note on / note off message. The range of Note Number is 0-127 and the central C of 88 key piano can be specified as Note Number =60. (Note Number =69 is set to A =440Hz)



With Velocity

Data Count	B7	b6	b5	b4	b3	b2	b1	b0
Data #0	1	0	0	1	Channel			
Data #1	0	Note Number						
Data #2	0	Key Velocity						
Data#3-6	S	Gatetime(Variable)						

The key of Note Number is pronounced by the strength of Key Velocity for the corresponding Channel. Gatetime is described by the variable length notation.

Channel can be 1-16 channels (0-15), and Note Number can be the value of 0-127 (sets 69 to A= 440Hz), and Key Velocity 0-127.

Stream PCM

Note Messege can be used as pronunciation directions of Wave Data stored in Stream Wave Data Chunk. Assigns to Wave Number of each Stream Wave Data Chunk for one Note Messege.

The assignment methods of Control Change Messege, Program Change Messege and Note Number etc., are depended on the installation system.

- Control Change Message

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	1	0	1	1	Channel			
Data #1	0	Control Number						
Data #2	0	Control Value						

The controller specified by Control Number is modified for the corresponding Channel.

- Program Change Message

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	1	1	0	0	Channel			
Data #1	0	Program Number						

The voice of assigned Channel is set up.

When the corresponding channel is set as the Normal channel, a voice is choosed from the bank specified by the bank selection.

A drum set is choosed, when the corresponding channel is set as the drum.

- Pitch Bend Message

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	1	1	1	0	Channel			
Data #1	0	Pitch Bend Change LSB						
Data #2	0	Pitch Bend Change MSB						

The pitch of the corresponding channel is changed up and down. The initial value of change width (pitch bend range) is ±2 semitone. The down pitch bend becomes the maximum by 0x00 / 0x00. The upper pitch bend becomes the maximum by 0x7F/0x7F. The pitch bend range can be set up by 0x00/0x00 of RPN.

- Exclusive Message

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0xF0							
Data #x	S	Size(Variable)						
Data #y	0	Data(Variable)						
Data #end	1	1	1	1	0	1	1	1

Sets up the Exclusive Message which exists as an event in music data. Size and Data is described by the variable length notation. Size is taken as from Data#y to Data#end.

- NOP

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0xFF							
Data #1	0x00							

This performs no operation.

- End of Sequence(EOS)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0xFF							
Data #1	0x2F							
Data #2	0x00							

It indicats the end of Sequence. EOS is not absolutely necessary, but its inclusion is desirable in order to explicitly indicate the end of the sequence. (Since End of Sequence in Mobile Standard Format is treated as Event unlike the case of Handy Phone Standard Format, Duration is required.)

Since EOS indicates the end, it is desirable for the system implementation to process the sequence in time-order, and then when EOS is detected, terminate the sequence and mute the sound. Even if a Sub-sequence List is used, it is expanded into a Stream Sequence and processed in order of time, and when EOS is detected it is considered to be the end of the sequence. If EOS does not exist, the ending location is calculated from the Chunk Size.

c) Huffman coding

The data format of Huffman coding is as the following.

Huffman(Data) = (Size) (Huffman Table) (Data)+
(Huffman Table) = ((Node/Leaf)(Code))+

Size is 4 bytes and expresses the byte count before encoding.
Node/Leaf of the Huffman table is set as Leaf by the bit 0, as Node by the bit 1. And set the Code of 0x00 -FF in case of Leaf. Data is arranged in the variable length bit according to the Huffman table.

7.) Stream PCM Data Chunk

Chunk ID "Mtsp" : Score Track Stream PCM data Chunk

Wave data for Stream PCM playback is stored. This consists of one or more Stream wave Data chunk.

Stream PCM Data Chunk Body = (Stream wave Data chunk)+

a) Stream Wave Data Chunk

Chunk ID "Mwa*" : Score Track Stream Wave Data Chunk

* = 0x00 ~ 0xFF is Chunk Number.

Chunk Number means Wave Number.

Wave Number	Description
0x00	Prohibition
0x01 - 0x3E	Wave ID
0x3F - 0xFF	Prohibition

Body internal representation is dependent on each Wave Format.

Stream Wave Data Chunk attaches Wave Type of the wave data to the head. Wave Data is stored after that. This internal representation of Wave Data is dependent on Wave Type.

Stream Wave Data Chunk body = Wave Type (Wave Data)+

The format of Wave Type is defined as below.

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	M/S	Format			Base Bit			
Data #1	Sampling Freq(MSB)							
Data #2	Sampling Freq.(LSB)							

Channel	Description
0x0	Mono
0x1	Stereo

Format	Description
0x0	2's complement PCM
0x1	Offset Binary PCM
0x2	ADPCM(YAMAHA)

Base Bit	Description
0x0	4 bit
0x1	8 bit
0x2	12 bit
0x3	16 bit
0x4 ~ 0x15	Reserved

In Sampling Freq, the frequency, which reproduces a wave, is specified by Hz.

4.5. PCM Audio Track Chunk

Chunk ID "ATR*" : PCM Audio Track Chunk

This is a type of Sound Track, and is the track chunk for PCM Audio tone generators. The data for Format perception and plural Sub Chunks are stored.

- Format Type : 1 byte (required)
- Sequence Type : 1 byte (required)
- Wave Type : 2 byte (required)
- TimeBase_D : 1 byte (required)
- TimeBase_G : 1 byte (required)
- Seek & Phrase Info Chunk : n byte (optional)
- Setup Data Chunk : n byte (optional)
- Sequence Data Chunk : n byte (required)
- Wave Data Chunk : n byte (required)

1.) Format type

This status defines the actual format of this track chunk. In order to minimize the data size, it is possible to use the native format of an LSI, or other sequence formats for powerful control CPU's in the future.

Format Type	Description
0x00	Handy Phone Standard
0x01~0xFF	Reserved

2.) Sequence Type

Sequence Data can be given in two different forms.

- 0x00 :Stream Sequence
  The sequence data consists of a single continuous stream. Seek Points and Phrase List can be used to reference desired locations in the sequence from outside.

- 0x01 :Sub-sequence
  The sequence data consists of a succession of multiple instances of phrase data. Phrase List can be used to reference individual phrases from outside.

  0x02 - 0xFF(reserved)

3.) Wave Type

Defines the format of Wave Data Chunk.

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	M/S	Format			Sampling Freq			
Data #1	Base bit				Reserved			

Channel	Description
0x0	Mono
0x1	Stereo

Format	Description
0x0	Signed(2's complement) PCM
0x1	ADPCM
0x2	TwinVQ
0x3	MP3
0x4~0x7	Reserved

Sampling Freq	Description
0x0	4 kHz
0x1	8 kHz
0x2	11 kHz
0x3	22.05 kHz
0x4	44.1 kHz
0x5~0x15	Reserved

Base Bit	Description
0x0	4 bit
0x1	8 bit
0x2	12 bit
0x3	16 bit
0x4~0x15	Reserved

* Reserved is intended for future use in recognizing the encoder table, etc.

4.) TimeBase_D, TimeBase_G

This expresses the time bases used within the sequence data.
TimeBase_D is the time base for Duration, and TimeBase_G is the time base for GateTime.

TimeBase_D,G	Description
0x00	1 msec
0x01	2 msec
0x02	4 msec
0x03	5 msec
0x04~0x0F	Reserved
0x10	10 msec
0x11	20 msec
0x12	40 msec
0x13	50 msec
0x14~0xFF	Reserved

5.) Seek & Phrase Info Chunk (optional)

Chunk ID "AspI*" : PCM Audio Track Seek & Info Chunk

This contains information that allows starting/stopping at a desired location (or repeating a desired region) within the sequence data. Its use and function are optional, and will depend on the type of device and on the intentions with which the data was created. (Details are given in chapter 4.4.1.)

6.) Setup Data Chunk (optional)

Chunk ID "Atsu*" : PCM Audio Track Setup Data Chunk

This contains parameters and effect settings etc. for the PCM encoder section. In most cases, this data can also be listed in the Sequence Data Chunk, but data bundling can be facilitated by making it a separate data chunk.

7.) Sequence Data Chunk

Chunk ID "Atsq*" : PCM Audio Track Sequence Data Chunk

This contains audio event data. If there is a Track Chunk, this Sequence Data Chunk is also required.

8.) Wave Data Chunk

Chunk ID "Awa*" : PCM Audio Track Wave Data Chunk

This contains wave data. If there is a Track Chunk, here must be at least one of these Audio Data Chunks. The last byte of the Chunk ID is the Audio Data Number, allowing a maximum of 255 waves to be expressed.

4.5.1. Handy Phone Standard Format

1.) Seek & Phrase Info Chunk

Chunk ID	"AspI"	: PCM Audio Track Seek & Phrase Info Chunk

The body contains data in the following order.

- Start Point :1 - 4 byte (optional)
- Stop Point :1 - 4 byte (optional)
- Phrase List :n byte (optional)
- Sub-sequence List :n byte (optional)

Content and use are the same as for Score Track.

2.) Setup Data Chunk

Chunk ID	"Atsu"	: PCM Audio Track Setup Data Chunk

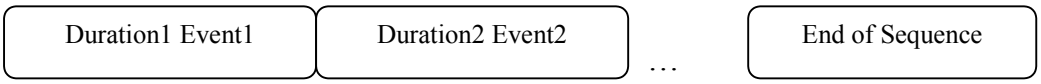
Content and use are the same as for Score Track.

3.) Sequence Data Chunk

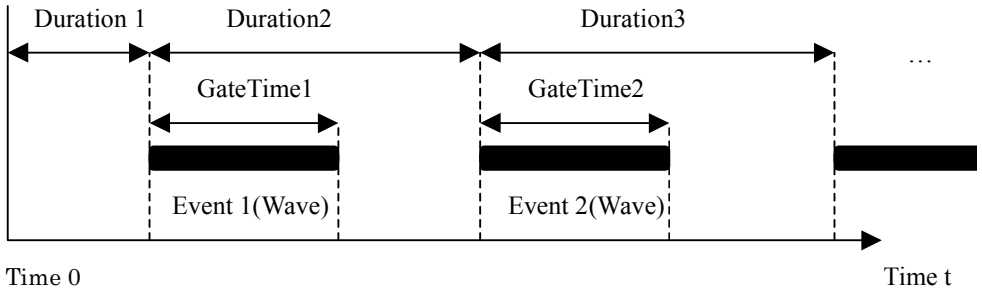
Chunk ID	"Atsq"	: PCM Audio Track Sequence Data Chunk

The Sequence Data Chunk consists of sequential pairs of Duration and Event, and End of Sequence.

Sequence Data Chunk body = ((Duration Event) | (End of Sequence))+



Duration is the time from the start time of the preceding event until the start time of the paired event. As an example, the following diagram shows the relation between the Duration and the GateTime (refer to (4) c)) of the Note Message, in the case of the Wave Message event. The Events are performed at the same time if the Duration is 0 (0x00).



a) End of Sequence

Four consecutive \x00 signify End of Sequence. EOS is not required, but it is desirable as an explicit indication of the end of the sequence.

End of Sequence = \x00 \x00 \x00 \x00

Since EOS indicates the end, the implementation will process the sequence in time-order, and will end the sequence and mute the sound when EOS is detected. Even if a sub-sequence list is used, the sub-sequence is developed as a time-order list and then processed in time-order, so that the point where EOS is detected will be the end of the sequence. If EOS does not exist, the end location is calculated from the chunk size.

b) Duration

This is expressed by a maximum of 2 bytes.

87654321	87654321	Type	Step
0xxxxxxx		Short	0 - 127
1xxxxxxx	0xxxxxxx	Medium	128 - 16511

The resolution of 1 step is defined by TimeBase_D in the Seek & Phrase Info Chunk.
If the Step number exceeds 16511, insert a NOP Event.

c) Event

Events are defined below. The frequently-used events Expression/ Pitch Bend/ Modulation have standard types and short types, and the data size can be reduced by using these two types appropriately.

- Wave Message

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	Channel		Wave Number					
Data #1	Gatetime (1st byte)							
Data #2	Gatetime (2nd byte) [optional]							

This message performs Wave playback on the specified channel. Gatetime is expressed in the same way as Duration. If GateTime = 0, the wave will be sounded for the wave data length of the Wave ID.

Channel	Description
0x0	Channel #0
0x1	Channel #1
0x2	Channel #2
0x3	Channel #3

Wave Number	Description
0x01~0x3E	Wave ID
0x00,0x3F	Prohibited

- Pitch Bend (normal type)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x4			
Data #2	Value							

Raise or lower the pitch of the corresponding channel.

Value	Description
0x00-0x7F	Value(default 0x40)
0x80-0xFF	Reserved

- Pitch Bend (short type)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		0	1	Value			

Raise or lower the pitch of the corresponding channel.

Value	Description
0x1~0xE	Value(default 0x8)
0x0,0xF	Reserved

Data 0x1-0x0E of the short type corresponds to steps of value 0x0E relative to normal type data 0x00-0x7F. The correspondence between normal type and short type Pitch Bend data is shown in the following table.

Short	Normal
0x1	0x08
0x2	0x10
0x3	0x18
0x4	0x20
0x5	0x28
0x6	0x30
0x7	0x38

Short	Normal
0x8	0x40
0x9	0x48
0xA	0x50
0xB	0x58
0xC	0x60
0xD	0x68
0xE	0x70

- Volume

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0x7			
Data #2	Value							

Adjust the volume of the corresponding channel.

Value	Description
0x00-0x7F	Value
0x80-0xFF	Reserved

The recommended equation is given below.
Gain[dB] = MUTE , Value = 0
Gain[dB] = $20*\log(Data^2/127^2)$, Value > 0

- Pan

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel		1	1	0xA			
Data #2	Value							

Specify the stereo position of the corresponding channel. The sound of the channel will be panned to the specified position between L (0x01) and R (0x7F). 0x00 is L.

Value	Description
0x00~0x7F	Value(default 0x40)
0x80~0xFF	Reserved

The recommended equation is given below.
Left Channel Gain[dB] = $20*\log(\cos(\pi/2*Data127)$
Right Channel Gain[dB] = $20*\log(\sin(\pi/2*Data/127)$

- Expression (normal type)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel			1	1	0xB		
Data #2	Value							

Adjust the volume of the corresponding channel during the song, relative to the Volume setting.

Value	Description
0x00~0x7F	Value(default 0x7F)
0x7F~0xFF	Reserved

The recommended equation is given below.
Gain[dB] = MUTE , Value = 0
Gain[dB] = $20*\log(Data^2/127^2)$, Value > 0

- Expression (short type)

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0x00							
Data #1	Channel			0	0	Value		

Adjust the volume of the corresponding channel during the song.

Value	Description
0x1-0xE	Value(default 0xE)
0x0,0xF	Reserved

In Expression, the Data 0x1-0x0E of Short type corresponds to Data 0x00-0x7F of Normal type as shown in the following table.

Short	Normal
0x1	0x00
0x2	0x1F
0x3	0x27
0x4	0x2F
0x5	0x37
0x6	0x3F
0x7	0x47

Short	Normal
0x8	0x4F
0x9	0x57
0xA	0x5F
0xB	0x67
0xC	0x6F
0xD	0x77
0xE	0x7F

- Exclusive Message

This message allows settings to be made for various parameters specific to a device.
This message is variable length. Data format is shown below.

	b7	b6	b5	b4	b3	b2	b1	b0
Status	1	1	1	1	1	1	1	1
Data #0	1	1	1	1	0	0	0	0
Data #1	Message Size							
Data #2	Maker ID							
Data #3	Format ID							
:	:							
Data End	1	1	1	1	0	1	1	1

Message Size is the byte count from Data_#2 to Data_End.
According to the Maker ID, the manufacturer-specific data format is defined for Data_#3 and following.
According to the Format ID, the model- (device-) specific data format for the corresponding manufacturer is defined for Data_#4 and following.

- NOP

	b7	b6	b5	b4	b3	b2	b1	b0
Data #0	0xFF							
Data #1	0x00							

This performs no operation.
Data #1 values other than 0x00 and 0xF0 are reserved.

4.) Wave Data Chunk

Chunk ID	"Awa*"	: PCM Audio Track Wave Data Chunk

* = 0x00 - 0xFF is the Chunk Number

The Chunk Number is used by the Wave Message of the Sequence Data Chunk..
It signifies the Wave Number.

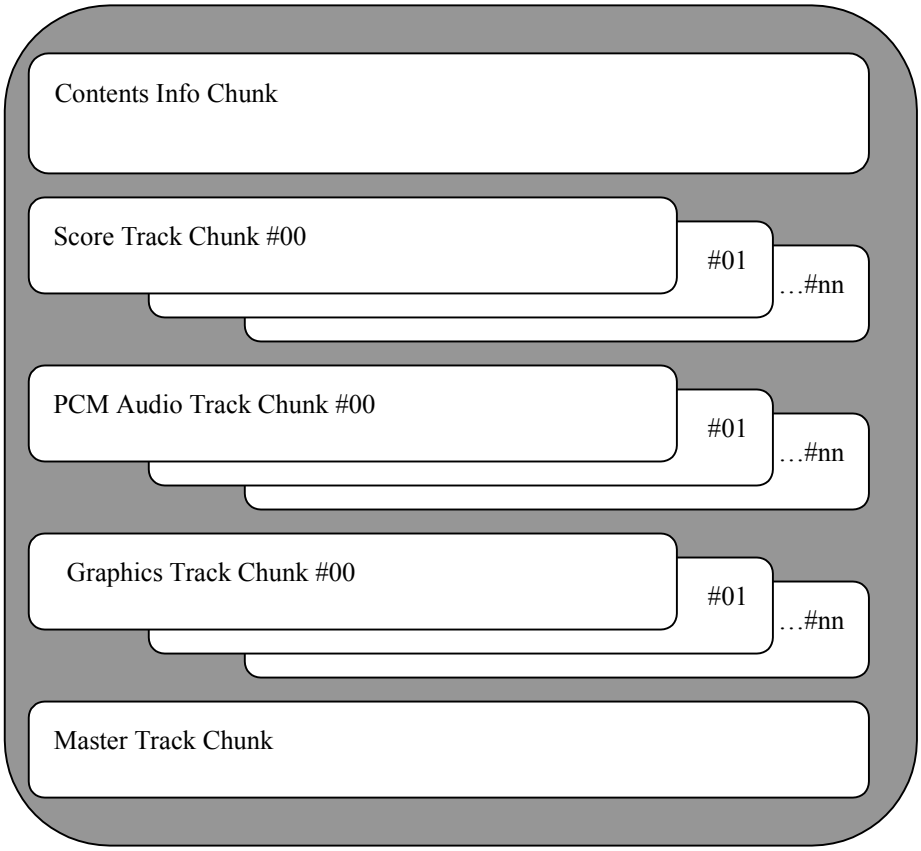
Wave Number	Description
0x00	Prohibited
0x01-0x3E	Wave ID
0x3F-0xFF	Prohibited

Expression within the Body depends on the applicable wave format.

5. Basic concepts of the Graphics Track

5.1. The position of the Graphics Track

Within the SMAF specification, the Graphics Track is data for expressing display sequences.



Conceptual diagram of SMAF Tracks

As shown by the above diagram, the Graphics Track is one of the tracks that constitute SMAF.

Each track is a data expression used to define a sequence for the corresponding output device.
A Score Track is a performance sequence for a tone generator device.
A PCM Track is a sound sequence for a PCM playback device.
The Graphicstrack describes a display sequence for a display device.

In the SMAF specification, all tracks are defined as starting simultaneously at time 0.
Since the SMAF playback system carries out playback processing of each track in parallel according to the time expressions of each track, the result is that playback processing for multiple output devices is synchronized.

As far as data expression goes, any desired number of each track (maximum 256 tracks) can be expressed. However the actual number of tracks is restricted by limitations of the SMAF playback system or limitations of content production. In the explanation that follows, it is assumed that there is a maximum of one graphics track. This is because SMAF playback systems foreseeable at the

time that this specification was designed have only one type of display device.

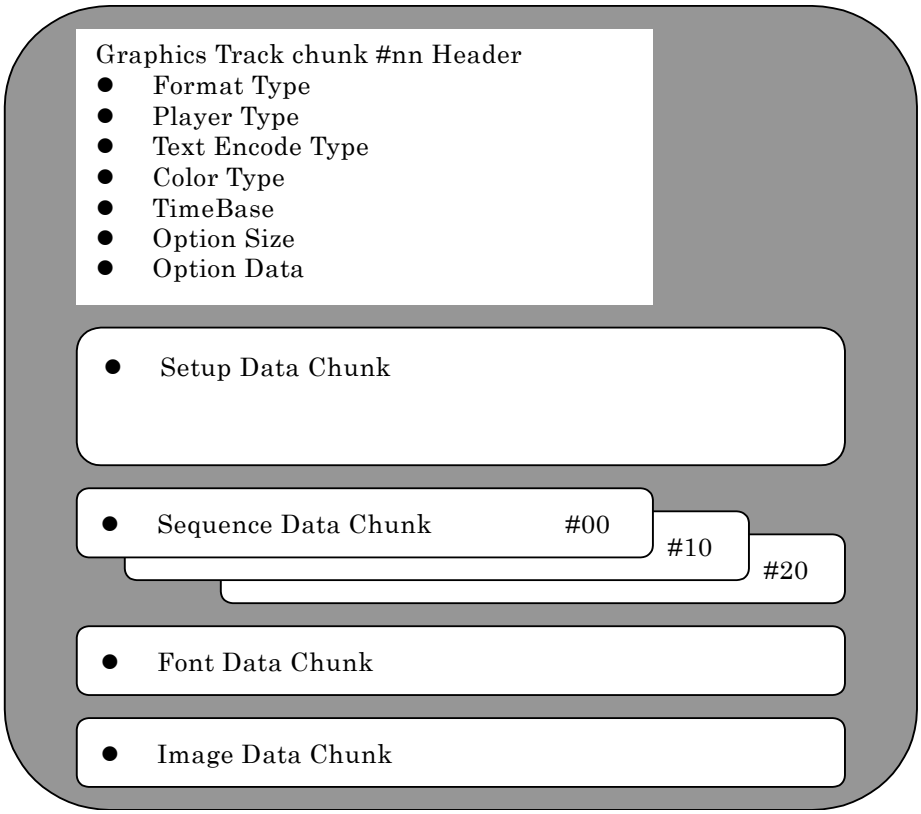
The display device assumed by the graphics track has multiple virtual planes. The graphics track allows the description of drawing sequences for each virtual plane. These drawing sequences are referred to as sub-sequences of the graphics track. Each sub-sequence defines procedures for displaying and erasing text and image data in synchronization with time.

The physical display device (such as the LCD display panel of a portable telephone) displays the result of combining the virtual planes.
This document also defines the method for combining the virtual planes. As many as 256 virtual planes can be defined as far as data expression goes, but the explanation in this document assumes an implementation of three planes. (Of these, one plane allows only the background color to be specified.)

5.2. Basic structure of the Graphics Track

The Graphics Track consists of

- Header (required)
- Setup Data Chunk (required)
- Sequence Data Chunk (required; 1 or more)
- Font Data Chunk (optional; maximum 1)
- Image Data Chunk (optional; maximum 1)



Conceptual diagram of the Graphics Track

5.3. Virtual display devices

From the viewpoint of the playback system, the display device is assumed to have three levels of virtual planes, and software is used to combine these planes to generate the final display image. Each plane is the same size as the physical display device, and allows 256 colors to be specified.

Definition of a virtual plane

A virtual plane is a virtual display device allocated in memory space.

For convenience in creating content, these virtual planes are referred to as follows.

- Plane 0 : backdrop plane (used to specify the background color; drawing is not possible)
- Plane 1 : background image plane (used mainly to display images)
- Plane 2 : text plane (used mainly to display text)

As many virtual planes can be defined as there are Sequence Data chunks (the number of sub-sequences) in the Graphics track. However, the explanation here assumes an implementation with the above three planes.

Virtual plane combining algorithm

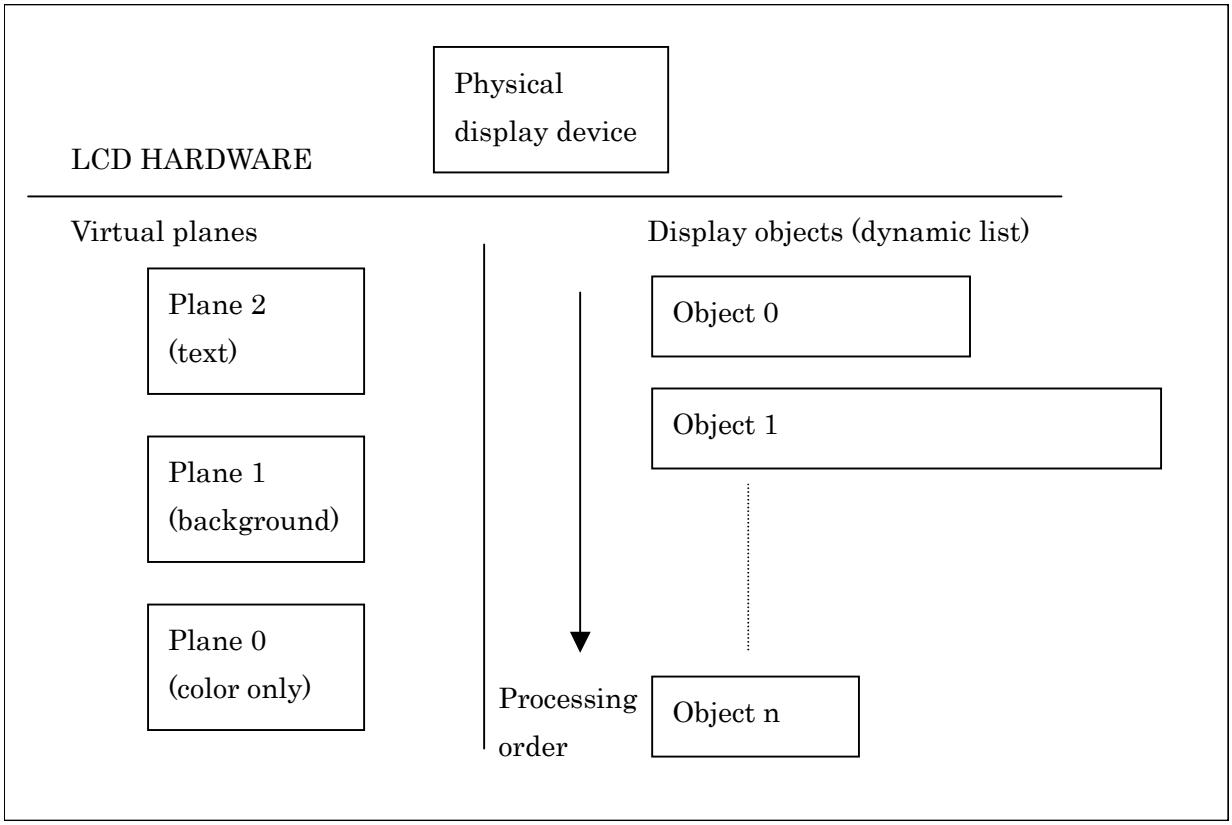
The lower-numbered plane is at a lower level.

Display objects displays in a higher plane will overlap display objects of a lower plane (described later).

A region in which there is no display object is interpreted as "clear," and allows the lower plane(s) to be seen.

When a display object is erased, the erased region will in general become "clear." (The case in which display objects overlap is defined separately.)

Display objects are rectangles, but clear pixels can be defined within the rectangle.



Conceptual diagram of the display device and virtual display devices

5.4. Sequence Data Chunks and virtual display devices

The correspondence between Sequence Data Chunks and virtual planes is as follows.

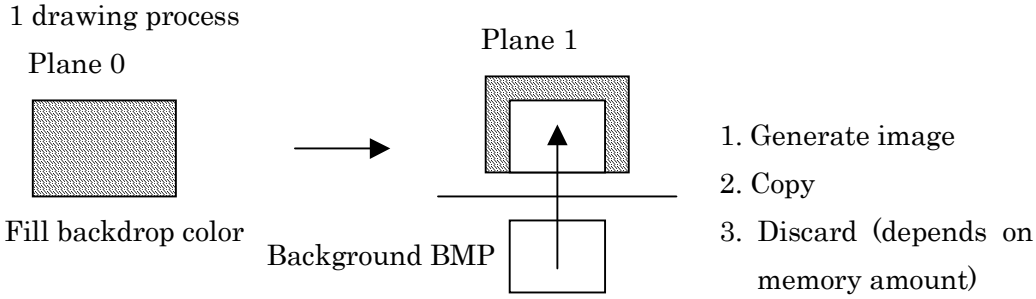
- Sequence Data 0x00 is a sequence for Plane 0 (only the BackDrop Color can be changed)
- Sequence Data 0x10 is a sequence for Plane 1
- Sequence Date 0x20 is a sequence for Plane 2

For multiple sequence data within the same graphic track, the data with the lower sequence data number is considered to be for a lower level.

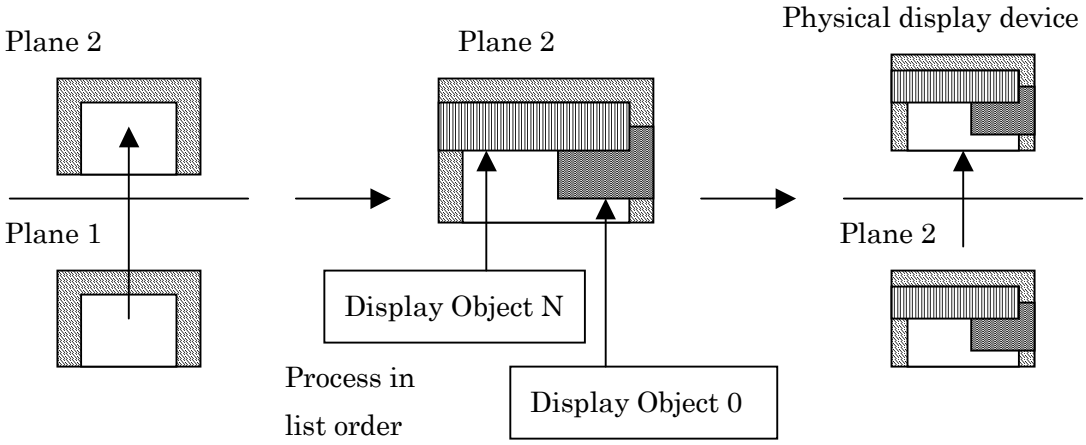
5.5. Virtual plane combining algorithm

A display object is an image that occupies a rectangular area (display area) in a virtual plane.
Display objects have a lifetime, and will always be erased when their lifetime ends.
For plane 1 and plane 2, the display objects described in the corresponding sequence data are managed as a list.

(1) Plane 1 drawing process



(2) Plane 2 drawing process and VRAM transmission



Conceptual diagram: Typical order of drawing process

For Plane 0

Not required for the virtual plane area; fill Plane 1 with the backdrop color.

For Plane 1 and Plane 2

The last-written display object will be displayed in front of previously-written display objects.

When a higher object is erased, lower objects will become visible.

The displayed content must be maintained while a display object is being displayed.

Display objects must be managed by a drawing history list, and the displayed content must be re-combined each time the display is updated.

Display objects can be moved within a plane.

Movement is created by repeatedly erasing -> re-displaying the object.

Operation when movement causes the display area to overlap with another display object is as described above.

Both plane 1 and 2 will return to their prior state when standard drawing processes are performed.

Display objects can be moved and effects created by associating a subsequence (§6.3.6 6) with the auxiliary sub-block of the display object.

5.6. Coordinate system

This section explains the coordinate system used to express locations on a virtual device or virtual plane, and the local coordinate system used to specify pixel locations within a display object.

5.6.1. Specifying display locations in a virtual plane

The sequence data in an SMAF file describes the display location of a display object on a virtual plane.

The base coordinate for the displayed location of a display object can be expressed using one of the following methods.

- Standard coordinates

  The coordinate origin is at the upper left. For the x-axis, left is the positive direction. For the y-axis, down is the positive direction.

  Standard coordinates specify the coordinates for the upper left of the display object.

- Symmetrical coordinates

  These coordinates are measured in the negative direction from the right edge or lower edge of the display screen.

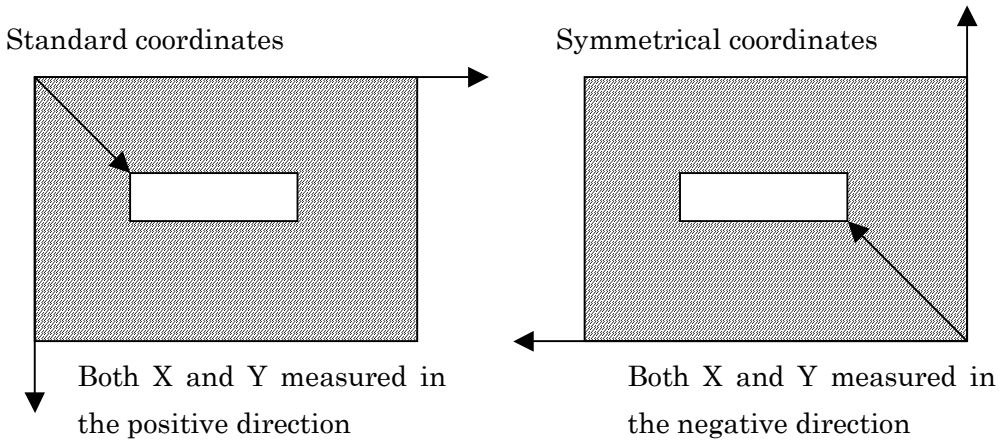
  The coordinate origin is the lower right. For the x-axis, left is the positive direction. For the y-axis, up is the positive direction.

  Symmetrical coordinates specify the coordinates for the lower right of the display object.

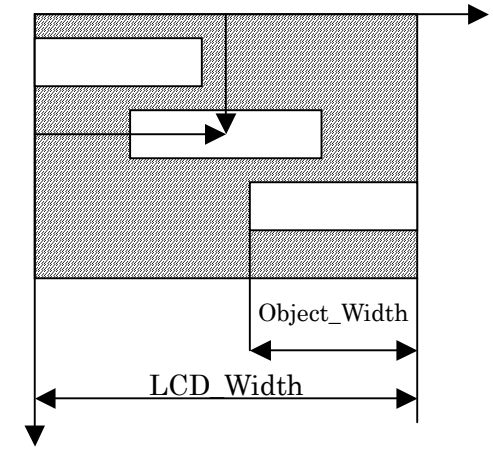
- Layout data specification

  This method specifies how the display coordinates will be calculated.

  Specify in 100ths.
  In the x-direction, 0% is flush left, 50% is centered, and 100% is flush right
  In the y-direction, 0% is flush to top, 50% is centered, and 100% is flush to bottom



Layout data



Specify in 100ths

X-direction 0: flush left
50: centered
100: flush right

Y-direction 0: flush to top
50: centered
100: flush to bottom

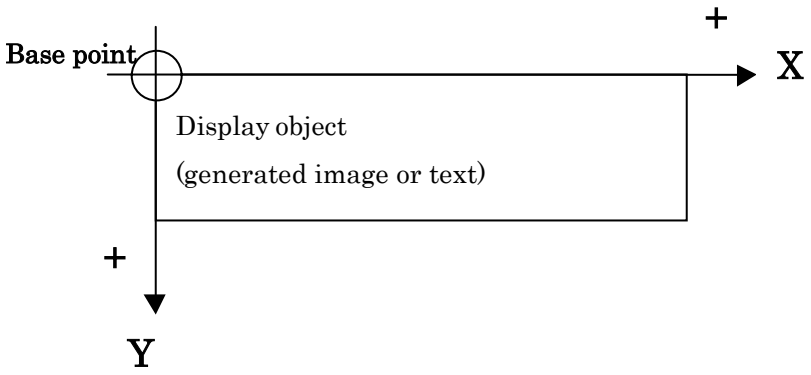
Standard coordinates Specified value= (LCD_Width - Object_Width) * Layout_Ratio / 100
If LCD_Width < Object Width, then a value of
0 will cause the right side
50 will cause both sides
100 will cause the left side
of the display object to protrude beyond the screen.

- Mixed specification

For the x-coordinate and y-coordinate, you can independently select whether to use coordinate specification or layout data specification.

5.6.2. Local coordinate system of a display object

The pixel locations on a display object can be specified using a coordinate system in which the base is at the upper left, and the x-axis is positive toward the right and the y-axis is positive toward the bottom.



6. Graphics Track data format

6.1. Graphics Track Chunk

In the Track Chunk ID, the first three bytes indicate the track, and the last byte indicates the track number in binary form.

The Chunk ID of the Graphics Track Chunk is given below.

Chunk ID "GTR*" :Graphics Track * = 0x00 - 0xFF

The internal contents of the Graphics Track are as follows.

- Header (required)
- Setup Data Chunk (required)
- Sequence Data Chunk (1 or more required)
- Font Data Chunk (maximum 1, optional)
- Image Data Chunk (maximum 1, optional)

Header contents

- Format Type : 1 byte
- Player Type : 1 byte
- Text Encode Type : 1 byte
- Color Type : 1 byte
- TimeBase : 1 byte
- Option Size : 1 byte
- Option Data : size specified by Option Size(0-255b)

1.) Format Type

Specify the data format of the graphic data to be used.

Format Type	Description
0x00	Handy Phone Standard
0x01-0xFF	Reserved

2.) Player Type

Indicate the class of assumed playback environment.

Player Type	Description
0x00	Handy Phone Standard
0x01-0xFF	Reserved

3.) Text Encode Type

Define the character code set that is used.

Text Encode Type	Description	Language
0x00	Shift-JIS	Japanese
0x01	Latin-1	English, French, German Italian, Spanish, Portuguese
0x02	EUC-KR(KS)	Korean
0x03	HZ-GB-2312	Chinese (simplified)
0x04	Big5	Chinese (traditional)
0x05	KOI8-R	Russian, etc.
0x06	TCVN-5773:1993	Vietnamese
0x07~0x1F	Reserved	Reserved
0x20	USC-2	Unicode
0x21	USC-4	Unicode
0x22	UTF-7	Unicode
0x23	UTF-8	Unicode
0x24	UTF-16	Unicode
0x25	UTF-32	Unicode
0x26-0xFF	Reserved	Reserved

4.) Color Type

Define the color code system that is used.

Color Type	Description
0x00	Direct RGB:=3:3:2
0x01	Index Color
0x02-FF	Reserved

Color specification for text or backdrop in SMAF data supports only 256 color display.
Color is specified by an integer between 0 and 255.
Color expressions of image data contained within the SMAF are not limited to 256 colors. However, the SMAF playback system will reduce this to 256 colors when the image is generated.

5.) TimeBase

Define the time base used internally.

The TimeBase is used as the reference time in the Graphics track, such as for Duration and LifeTime.

The actual time is calculated by multiplying the time data with the TimeBase.

TimeBase	Description
0x00	1 msec
0x01	2 msec
0x02	4 msec
0x03	5 msec
0x04~0x0F	Reserved
0x10	10 msec
0x11	20 msec
0x12	40 msec
0x13	50 msec
0x14	60 msec
0x15	70 msec
0x16	80 msec
0x17	90 msec
0x18	100 msec
0x19-0xFF	Reserved

6.) Option Size

Specify the size of the extension data that follows Option.

7.) Option Data

Specify the extension data for the amount of the Option Size(0-255b).

Specification of description in Option Data

Tag is defined for every data. Data is variable length for every tag.

[Tag (2byte)] + [Size (1byte)] + [Data (Variable length byte designated by Size)]

Name	Tag name	Hex
Rendering Size	RS	0x52 0x53
GTR Locale	GL	0x47 0x4C

- RS tag (Dsplayed screen size)

  The size of the drawing area for display system contents is designated.

  Display system player determines actual drawing size as compared with the drawing area which can be displayed.

  Data

  RS tag 0x52 0x53 : 2byte

  Size 0x04 : 1byte

  Data xx xx yy yy : 4byte

  xx xx : 2byte : Width of the drawing area for SMAF contents (-32768~32767)

  yy yy : 2byte : Hight of the drawing area for SMAF contents (-32768~32767)

- GL tag

When GTR is created in Unicode, the information of language and assumed service area about character sequence data written in this chunk is described here.

GL tag Data : 2 byte (Option)

<Data> = < Language > < Carrier etc>
BYTE BYTE

Language	Description
0x00	Japanese
0x01	Latin-1
0x02	Korean
0x03	Chinese (simplified)
0x04~0xFE	Reserved
0xFF	ASCII

Carrier etc.	Description
0x00	Generic (Global)
0x01	Reserved
0x02~0xFF	Reserved

When the special pictorial symbol and the extended character are defined as Unicode, information about career etc. may be used in order to avoid duplication and incorrect operation.

GL tag is described as Octet Stream, when creating Unicode contents.

6.2. Setup Data Chunk

The Setup Data Chunk is used to define various parameters required for display. (1 is required)

Chunk ID "Gtsu" :Graphics Track Setup Data Chunk

Contents of the Setup Data Chunk

Display Parameter Definition Chunk (1 is required)
Color Palette Definition Chunk (optional)

6.2.1. Display Parameter Definition Chunk

The Display Parameter Definition Chunk is used to define display parameters specific to each display event type.

Chunk ID "Gdpd" :Display Parameter Definition Chunk

The Default display parameters must be defined.
Parameter definitions for each Display Object Event Type are optional.
The data structure is an array of variable length records, in which definition data is repeated for each event type.

Display Parameter Definition Chunk

```
Repeated for each Event Type {
   cord Size    :1-2 bytes   Data length for each Event Type; use Duration expression.
   Event Type :1 byte        Specify the Event Type that will use the following display parameters
         repeated {
            PrmID          :1 byte
            Value          :1byte
         }
      }
```

Specify Event Type = 0x00 and define default parameters.
This event type is used only within the Setup Data Chunk.
Valid event types are in the range of 0x40...0x7f (listed below). Definitions for other event types are ignored.

PrmID	Description	
0x01	Font Type	font type
0x02	Font Size	font size
0x03	Direction	font direction
0x04	Attribute	font attribute (future use)
0x05-0x0F	Reserved	
0x10	Font Color 0	font color
0x11	Font Color 1	font color after color change
0x12	Edge Color 0	font edge color (future use)
0x13	Edge Color 1	font edge color after color change (future use)
0x14	Back Color 0	font background color
0x15	Back Color 1	font background color after color change
0x16-0x1F	Reserved	
0x20	Coordinates	default coordinate method
0x21-0x2F	Reserved	
0x30	BackDropColor	color for Plane 0 backdrop
0x31	Transparent Color	color used as transparent
0x32	Transparent Enable	enabling flag for transparency processing
0x33-0xFF	Reserved	

Font attribute specifies a Bold, Italic, Outline, or Shadow font

Coordinates

For the possible methods, refer to §6.3.2

BackDropColor

This is valid only as the Type = 0x00 default setting in the Setup Definition Chunk.

Changes in backdrop color in sequence data for other than plane 0 will be ignored.

Transparent Enable

Transparency processing is disabled for 0x00, and enabled otherwise.

6.2.2. Color Palette Definition Chunk

Chunk ID "Gcpd" :Color Palette Definition Chunk

Details of the Color Palette Definition Chunk are currently undefined.

If the Color Type of the header is 0x01 and there is no Color Palette Definition Chunk, this will rely on the system color palette.

6.3. Graphics Track Sequence Data Chunk

6.3.1. Graphics Track Sequence Data Chunk

The Graphics Track Sequence Data Chunk expresses a display sequence.

Chunk ID "Gsq*" :Graphics Track Sequence Data Chunk * = 0x00 - 0xFF

The first three bytes of the ChunkID in the Sequence Data Chunk indicate what type of chunk it is, and the final byte indicates the Sequence Data Number as a binary value.

Sequence Data Number = *	Description
0x00	Plane 0
0x01-0x0F	Reserved
0x10	Plane 1
0x11-0x1F	Reserved
0x20	Plane 2
0x21-0xFF	Reserved

The playback system has multiple virtual display planes which it uses to combine the display data, and relates these planes to the sequence data. Lower numbers are defined as being at a lower level.

Sequence Data syntax

Sequence data is a variable length byte stream.

Sequence data consists of Durations, Events, and EOS.

The first element in sequence data is a Duration, and Durations and Events must occur alternately.

EOS indicates End Of Sequence. It is expressed by four or more consecutive bytes of 0x00.

When interpreting sequence data, EOS is checked before interpreting events and duration.

The data format guarantees that there will be no events or durations in which four or more bytes of 0x00 occur.

If successive durations of 0x00 and NOP events of 0x00 occur so that there are four or more consecutive 0x00 bytes, this will be interpreted as EOS. If there is subsequent sequence data, it will be skipped by referring to the Size data.

Numerical expressions used to express sequence data can use either Coordinates or Duration.

The numerical expressions used to specify Coordinates and the Duration expressions used to specify time or length in SMAF are explained below.

Values noted as using "Coordval expressions" are specified by a one or two-byte Coordval expression.

Similarly, values noted as using "Duration expressions" are specified by a one or two-byte

Duration expression.

6.3.2. Coordinates and Coordval

Coordinate expression details (Coordinates)

Coordinates := [Format specification+] Position+Position

Coordinates	b7	b6	b5	b4	B3	b2	b1	b0
Formatspec.	1	1	1	1	xx		yy	
Position	first byte of x-coordinate							
[optional]	second byte of x-coordinate [optional]							
Position	first byte of y-coordinate							
[optional]	second byte of y-coordinate [optional]							

The first Position indicates the x-coordinate, and the second indicates the y-coordinate.

Format specification := '1111xxyy'

xx is the format of the x-coordinate

yy is the format of the y-coordinate

xx or yy	Description
0x0	Standard coordinate
0x1	Symmetrical coordinate
0x2	Layout
0x3	Not specified

In some cases, the format specification may be omitted.

If the format specification is omitted, the default format will be used.
By default, Standard coordinates will be used for both X and Y.

About the interpretation sequence of the coordinate system assignment when it is omitted

1. The default coordinate method is specified by Type = 00 in the Display Parameter Definition Chunk.
2. Moreover, it is also possible to define the coordinate assignment format for each Event Type to override the assignment.
3. Furthermore, it is possible to set to override the Format assignment for each Type of Display Parameter Definition Chunk with addition of Parameter Override Sub-block also in Event of Sequence Data Chunk.

A special interpretation is made as below if the coordinate assignment is contained in the same Event (From Primary to Auxiliary end).
When individual Format assignment for each Data is done in Event (From Primary to Auxiliary end), adopts as a default value which overrides the assignment by three upper methods.

And the default assignment will remain valid until the next format specification or until the end of Auxiliary Sub-block if no other assignment.

Position := Standard coordinate specification | Symmetrical coordinate specification | Layout data

Standard coordinate specification :1-2byte := uses Coordval expression (described later)

Symmetrical coordinate specification:1-2byte := uses Coordval expression (described later)

Layout data :1byte := 0...127 indicates 0...100%

Coordval:= variable-length data expression indicating a coordinate value (expressed as 1-2 bytes)

The range that can be expressed is-2048 to +2047.

1-byte notation (expresses 0-223)

0=>0x00

1=>0x01

...

222=>0xDE

223=>0xDF

2-byte notation(expresses -2048 through 2047)

In 0xExxx, the "xxx" is interpreted as a 12-bit signed integer.

-2048=>0xE800

-2047=>0xE801

...

-1 => 0xEFFF

0 => 0xE000

1 => 0xE001

...

2047=>0xE7FF

6.3.3. Duration

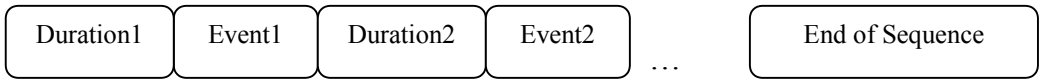
Duration expression

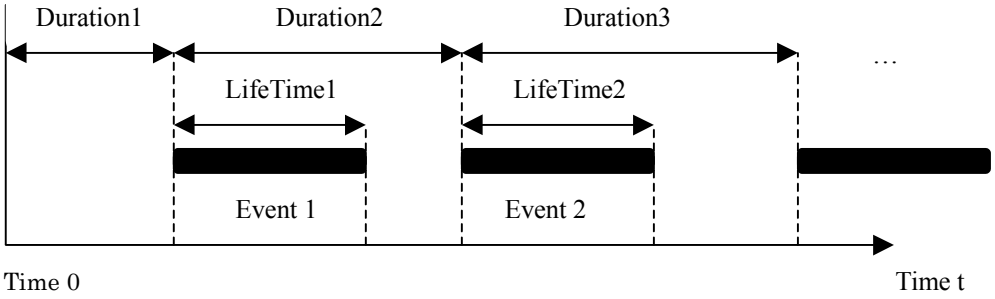
Duration specifies a time interval between events.

Duration is expressed in units of "the time specified by TimeBase"

The starting time of the subsequent event can be determined by accumulating the Duration values from the beginning of the data.

Event start time = (sum of all prior durations) * TimeBase





Conceptual diagram of Sequence Data

This is expressed either as 1 byte or 2 bytes. If the first byte has an MSB of 0, it is a one-byte expression. If the MSB is 1, it is a two-byte expression. If the second byte exists, the MSB of the second byte is always zero.

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#1	S	Data1						
Data#2	0	Data2						

The range of values is as follows, depending on the MSB (=S) of the first byte.

S	Step
0x0	0~127
0x1	128~16511

If S is 1, the expression is a value equal to 128 plus the 14-bit data of which data1 is the MSB and data2 is the LSB.

1-byte expression

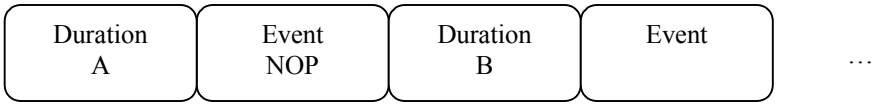
0x00~0x7F := 0x00 through 0x7F (0~127)

2-byte expression

0x80,0x00 := 0x0080(128)
0x80,0x01 := 0x0081(129)
0x81,0x00 := 0x0100(256)
...
0xff,0x7F := 0x407F(16511)

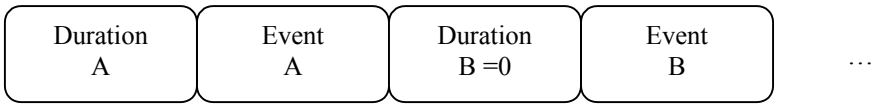
The resolution of one step is specified by the TimeBase in the Graphics Track Chunk. You may use NOP events to divide a single duration into two or more durations. In this case, the sum of the two divided durations is equal to the original duration. Use this method if the number of steps exceeds 16,511.

Ex) If the required Duration is 16511+ α = A + B



In order to define multiple events that occur at the same time location, insert Duration = 0 (for example) between Event A and Event B.

Ex) Defining Event A and B at the same time location



6.3.4. End definition

The length of the Graphics track is defined as the length of the longest sub-sequence (Sequence Data Chunk) included in the Graphics track.

The length of a sub-sequence is defined as the location of EOS (End Of Sequence). However, if two or more sub-sequences in the Graphics track have different lengths, the SMAF playback system will understand the EOS of all sub-sequences as being at the same time location as the EOS of the longest sub-sequence.

If the EOS is later than the erasure time (display time + LifeTime) of a display object event listed in the sub-sequence, the EOS location will take priority. The EOS we speak of here is the "corrected EOS" as described above. If a display object is being displayed at the EOS timing, it will be forcibly erased and simultaneously that sub-sequence will end.
In other words, in order to ensure that the display object for the last event will be displayed for the LifeTime, it is necessary to insert a duration that is greater than the LifeTime.

There may be cases in which the sequence data listed in an Auxiliary Sub-block of a Display

Object Event is longer than the LifeTime. Even in this case, the definition of the upper level (i.e., the LifeTime length) will be given priority for interpretation. At the erasure time of the Display Object Event (display time + LifeTime), the Auxiliary Sub-block sequence being executed will be forcible terminated and simultaneously the Display Object will be erased.

In principle it is desirable that the sequence data listed in the Auxiliary Sub-block be less than the LifeTime, and you should create data so that the erasure time of the Display Object Event (display time + LifeTime) is the same or earlier than EOS. The explanation above defines the behavior of the playback system when interpreting data that does not observe this guideline.

Similarly, if there is a difference in the length of the sequence data listed in each Sequence Data Chunk in the Graphic Track and Score Track PCM Track, the longest EOS will be used.

For example
if the Score Track sequence data is longer than the Graphics Track sequence data, only the background color will be displayed after the last Graphics display object has been erased (or erased due to EOS), and this will be held until the end of the longest EOS.

6.3.5. Event

Definition of the standard formant for Event

Event Type	: 1 byte	
Event Size	: 1-2 bytes	use Duration expression
Event Data	: Event Size bytes	variable length 0...16511 byte

The Event Size indicates the size of the Event Data.

There are the following three types of events

Short Control Event

Control Event

Display Object Event

Event details

1.) Short Control Event

(Expressed by 1 byte. It has no Event Size or Event Data.)

Event Type = 0x00..0x1f

Event Type	Description
0x00	NOP
0x01	Reset Origin Event
0x02~0x1F	Reserved

- NOP Event (No OPeration Event expressed by 1 byte. No Event Size or Event Data)

  Event Type = 0x00 is a 1-byte event, and is interpreted as NOP. It does nothing.

- Reset Origin Event

  Return the origin point that had been offset by Offset Origin Event back to the original, non-offset setting.

If the origin point has not been offset, this does nothing.

This is valid only for the Graphics Sequence Track in which the Reset Origin Event occurs.

2.) Control Event

Event Type = 0x20..0x3f

Event Size	:1~2byte	use Duration expression
Event Data	: Event Size bytes	variable length 0....16511 byte

Event Type	Description
0x20	BackDrop Color Definition Event
0x21	Offset Origin Event
0x22	User Event
0x23~0x3F	Reserved

- BackDrop Color Definition Event

	b7	b6	b5	b4	b3	b2	b1	b0
Event Type	0x20							
Event Size	0x01							
Event Data	BackDrop Color							

Defines the BackDrop Color. This will be ignored if specified for any other than Plane 0 (Chunk ID = Gsq0).

Value	Description
0x00~0xFF	BackDrop Color

- Offset Origin Event

	Description	b7	b6	b5	b4	b3	b2	b1	b0
Event Type		0x21							
Event Size		0x05~0x09							
Event Data	Coordinates	1	1	1	1	xx		yy	
		first byte of x-coordinate							
		second byte of x-coordinate[optional]							
		first byte of y-coordinate							
		second byte of y-coordinate [optional]							
	Xwidth	first byte of Xwidth							
		second byte of Xwidth [optional]							
	Ywidth	first byte of Ywidth							
		second byte of Ywidth [optional]							

Event Type = 0x21

Event Size : 1~2 bytes use Duration expression

Event Data

Coordinates : 3~5 bytes specify coordinates at which the offset rectangle will be placed. Specification of the coordinate system is required, and may not be omitted.

Xwidth : 1~2 bytes use Duration expression

Ywidth : 1~2 bytes use Duration expression

Move the location of the origin point for standard coordinates (upper left of the screen) and the origin point for symmetrical coordinate (lower right of the screen).

This does not affect the calculation when layout coordinates are specified.

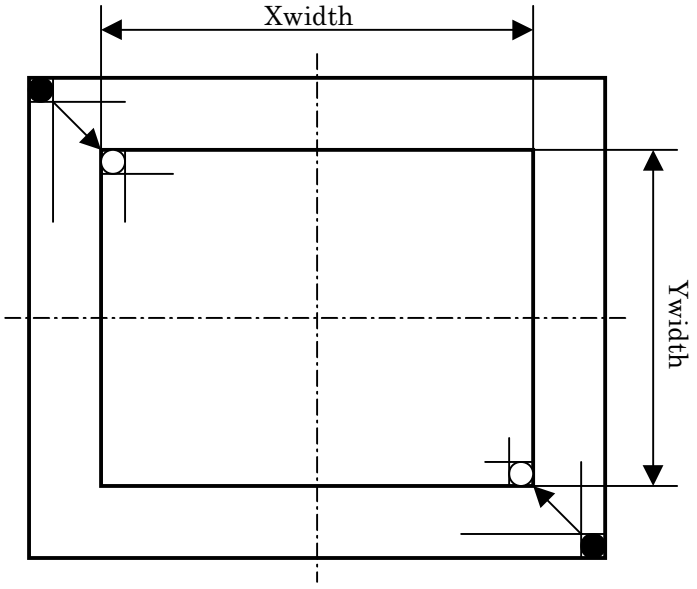
If the specified Xwidth and Ywidth are0x00, this is interpreted as if 0x01 had been specified.

The origin point for standard coordinates and symmetrical coordinates will move for Events that follow this control event. This will remain valid until the next time the control event

appears, or if not, until the end of the sequence.

If this is updated while a display object is displayed, priority will be given to the origin point at the time that the display event was interpreted. In other words, this depends on the surroundings of the event rather than on the interpretation of LifeTime.

The Offset Origin Event is effective only for the Graphics Sequence Track in which it is placed, and does not affect other sequence tracks.



● Original base point

○ Moved base point

(Example) Specify a rectangle slightly smaller than the display device, and use layout coordinates to center it.

The base point will move for the standard coordinate system and the symmetrical coordinate system. Flush-left or flush-right for layout coordinates will be calculated at the original size, and will not be affected by the rectangular region of this event.

- User Event

	b7	b6	b5	b4	b3	b2	b1	b0
Event Type	0x22							
Event Size	0x01							
Event Data	0	0	0	0	User Event ID			

User chooses Event Data from 0x00-0x0F arbitrarily.

User Event is set only on Plane0 and cannot be set on Plane1 or Plane2.

3.) Display Object Event

- General-purpose display event

Event Type	Description
0x40~0x7F	General purpose display event
0x80~0xFF	Reserved

		b7	b6	b5	b4	b3	b2	b1	b0
Event Type		0x40 ~ 0x7F							
Event Size		first byte of Size							
		second byte of Size [optional]							
LifeTime		first byte of LifeTime							
		second byte of LifeTime [optional]							
Coordinates	may omit	1	1	1	1	xx		yy	
		first byte of x-coordinate							
		second byte of x-coordinate [optional]							
		first byte of y-coordinate							
		second byte of y-coordinate [optional]							
Sub-block	Primary is required	listed separately (variable length)							
	Auxiliary	listed separately (variable length) [optional]							

The Primary Object Sub-block is required for the general-purpose display event.

Event Type = 0x40...0x7F used as an identifier.
Event Size :1~2 byte use Duration expression
Event Data
LifeTime :1~2byte use Duration expression, and specify the time from start of display to erasure.
Coordinates :2~5byte specify coordinates of display location.
Object Sub-block :variable length Option specify the display object, or the operation variation.

The Event Type is a distinguishing symbol used for extracting the display parameters defined for each Event Type by the Display Parameter Definition Chunk, and causing them to be reflected by display processing. For the purposes of SMAF data definition, there is no difference in the way in which various types of general-purpose display events are defined.
Parameters that assign meaning (such as the color to be produced) are specified in the Display Parameter Definition Chunk (§6.2.1).

The Event Size indicates the data size in byte units from the subsequent LifeTime to the end of the last Object Sub-block.

LifeTime indicates the time from when the display object specified by the sub-block is displayed until it is erased.
LifeTime is specified in TimeBase units.
Coordinates specify the display position of the display object.

The contents of the display object are defined in the Object Sub-block within the Display Object Event.

The SMAF specification does not assign meanings to each event type of the general-purpose display event.
The operational significance of each event type is specified by rules in the data production guidelines.
Example) lyrics, male part lyrics, female part lyrics, mixed part lyrics, song title, lyricist name, composer name, singer name, lines, interlude, comments, etc.

6.3.6. Object Sub-block

The general format for Object Sub-block data is as follows.

Sub-block Type :1byte
Sub-block Size :1~2byte
Sub-block Body :Sub-block Size bytes

There are two types of Object Sub-block: Primary and Auxiliary.
A Primary Sub-block specifies the content of the display object used by an event.
An Auxiliary Sub-block specifies attributes that apply operations (effects or changes) to a display object.

1.) Primary Sub-block

The Primary Sub-block describes the content of a display object.
For a general-purpose display event, one and only one Primary Sub-block must be specified.

There seven types of Primary Sub-block, as follows.

- Text
- Bitmap (binary image)
- Image (free image)
- Rectangle (rectangular area)
- Text Block (text with line returns)
- Image Tile (image tile)
- Bitmap Tile (binary image tile)

Sub-block Type	Description
0x01	Text
0x02	Bitmap
0x03	Image
0x04	Rectangle
0x05	Text Block
0x06	Image Tile
0x07	Bitmap Tile
0x08~7F	Reserved

- Text display object

This specifies a one-line character string as a display object.

Character string (variable length) specified by the Header of the Graphics Track Chunk 0x00 is interpreted as End Of Data (EOD). The character string is considered to be broken. If there is no EOD, the end will be determined from the Size data.
Since EOD cannot interpret correctly depending on a character code, EOD is not used in that case.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x01							
Size	first byte of Size							
	second byte of Size [optional]							
Body	character string data (variable length)							

Sub-block Body : variable length

For the specified character string data, a color image will be generated using the display parameters specified by the event type.

"Gaiji" (user-defined characters for the Japanese character code set) may be embedded in the character string data. When Text Encode Type is S-JIS, JIS character code sectors 14 and 15 are interpreted as the "gaiji" code area. When Unicode, interprets Private Use Area as the “gaiji” code.
If character codes in these areas appear, the "gaiji" bitmap data of the Font Data Chunk (§6.4.1) will be used as character font data, and processed for display.
If the code specified by the "gaiji" data does not exist in the Font Data Chunk, then a space for that font size will be displayed.

If there is no character string, the display event will be ignored.

- Bitmap display object

This specifies a binary image as a display object. It specifies data within the Bmp Chunk.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x02							
Size	0x01							
Body	image number							

Sub-block Body :1byte := image number

Specify the image number in the Primary Sub-bock Body. The image number is the value of the fourth byte of the Bmp Chunk (§6.5.2) in the Image Data Chunk.
The display parameters determined from the event type are applied to the specified

bitmap image data to generate a color image.
When the bitmap image data is developed, 1 is displayed as the text display color and 0 is displayed as the text background color.

This means that by specifying a bitmap (binary image), you can achieve the same result as if you had defined a character string of the desired shape.

If the specified image number does not exist, the display event will be ignored.

- Image display object

  Specify image display as a display object. This specifies data in the Image Chunk.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x03							
Size	0x01							
Body	image number							

Sub-block Body :1byte := image number

Specify the image number in the Primary Sub-block Body. The image number is the value of the fourth byte of the ChunkID of the Image Chunk (§6.5.2) in the Image Data Chunk.
When developing the specified image data into an SMAF display object, it must be matched to 256 colors.

If the specified image number does not exist, the display event will be ignored.

- Rectangle display object

  Specify a display object consisting of a rectangular area and a display color by which it will be filled.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x04							
Size	0x03~0x05							
Body	first byte of Xwidth							
	second byte of Xwidth [optional]							
	first byte of Ywidth							
	second byte of Ywidth[optional]							
	Color							

Sub-block Body :=

Number of X-direction dots Xwidth	:1~2 byte	use Duration expression.
Number of Y-direction dots Ywidth	:1~2 byte	use Duration expression.
Fill color	:1 byte	

If either Xwidth or Ywidth is 0x00, the display event will be ignored.

- TextBlock display object

Draw text with line returns as a display object.

The interpretation of the text and the basic display is the same as for a Text display object.

Character string (variable length) of the character code specified in the Header of the Graphics Track Chunk.

0x00 is interpreted as End Of Data (EOD). The character string is considered to be broken. EOD may be omitted.

A character string judges termination from Size Data.

Since EOD cannot be interpreted depending on a character code, EOD is not used.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x05							
Size	first byte of Size							
	second byte of Size [optional]							
Body	first byte of Block Width							
	second byte of Block Width [optional]							
	first byte of Line Space							
	second byte of Line Space[optional]							
	Back Color							
	character string data (variable length)							

Sub-block Body : variable length

Block Width :1~2 bytes use Coordval expression.
(width for left-right text; height for up-down text)

Line Space :1~2byte use Duration expression.

Back Color :1byte define the background color in the block.

Text : variable length

Block Width is handled as a 12 bit signed integer(Coordval expression). (-2048~2047)

If 0 is specified, this is considered as if the LCD width of the system had been specified.

The value of the positive range specifies as the number of pixels for the actual block side.

The value of the negative range specifies the number of pixels subtracted from the size of the system LCD.

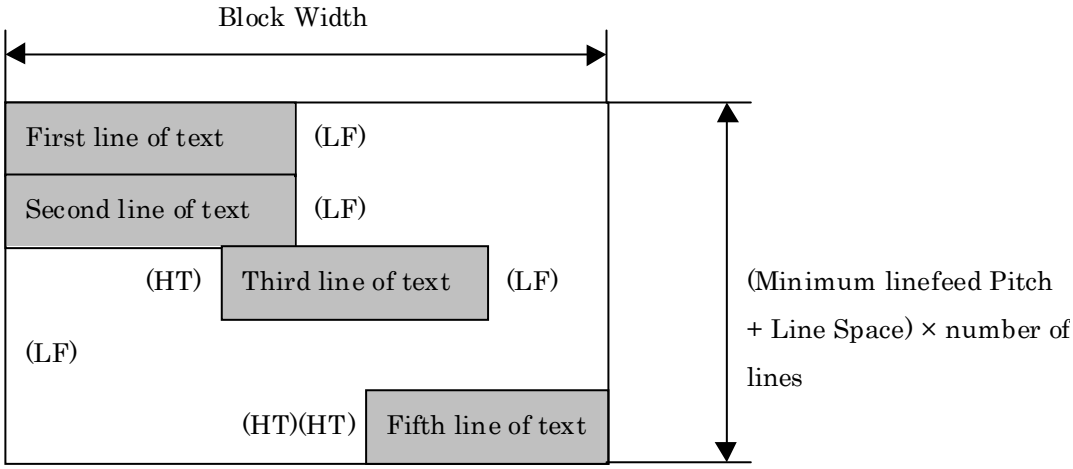
Line Space specifies the line return pitch.

When 0 is specified, it is regarded that the minimum text paragraph pitch depending on the system font size is specified.

The space produced by enlarging Line Space is smeared away by BackColor of TextBlock.

The following text data control codes are recognized.

Line feed	LF	(Example, in S-JIS, 0x0A)
Centering	HT	(Example, in S-JIS, 0x09)
Right justify	HT HT	(Example, in S-JIS, 0x09 0x09 2bytes)



If the text exceeds the Block Width without a line feed, a line feed will be inserted according to the Block Width. This forced line feed will be inserted if the next character is not LF. The display event will be ignored if the Block Width is smaller than the font size. When specifying vertical text, the Block Width is the vertical size, and the Line Pitch is the horizontal pitch.

- Image Tile display object

Tile the desired image from the Image Chunk and render it as a display object.
The image number is specified in the Primary Sub-block Body. The image number is the value of the fourth byte of the ChunkID of the Image Chunk (§6.5.1) in the Image Data Chunk.

When developing the specified image data into an SMAF display object, it must be matched to 256 colors.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x06							
Size	0x05~0x0A							
Body	image number							
	first byte of Xwidth							
	second byte of Xwidth [optional]							
	first byte of Ywidth							
	second byte of Ywidth [optional]							
	1	1	1	1	xx		yy	
	first byte of X-coordinate							

	second byte of X-coordinate [optional]
	first byte of Y-coordinate
	second byte of Y-coordinate [optional]

Sub-block Body

Image number :1 byte specify the image data in the Image Chunk

Image size : specify the image size for Xwidth and Ywidth

Coordinates of paste location : use 2~5 byte Coordinates expression.

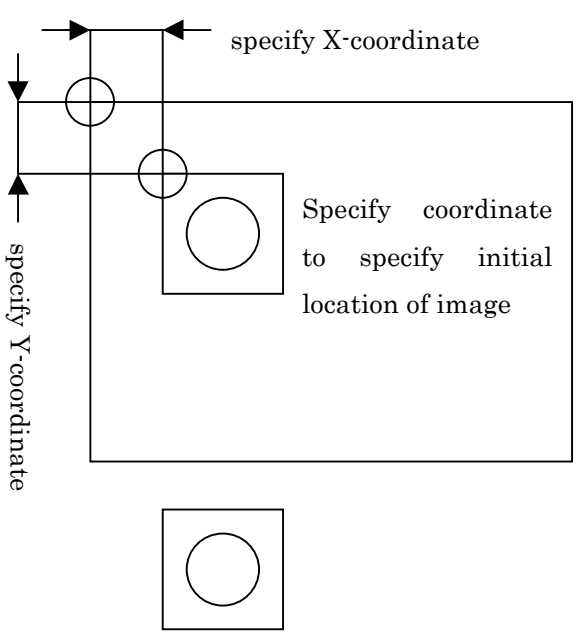
In the case of

Xwidth = 0

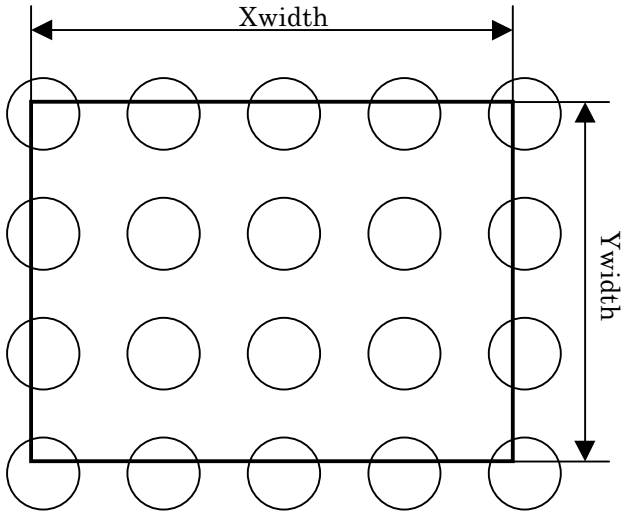
Ywidth = 0

the respective size will be the LCD size.

The specified image will be laid out in tile fashion from the base location, to the same size as the physical display device.



Specified image



Layout result image

- Bitmap Tile display object

Tile the desired image from the Bmp Chunk as a display object.

The Primary Sub-block Body specifies the image number. The image number is the value of the fourth byte of the ChunkID of the Bmp Chunk (§6.5.1) in the Image Data Chunk.

When developing the specified image data into an SMAF display object, use the font color

and font background color display parameters.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x07							
Size	0x05~0x0A							
Body	image number							
	first byte of Xwidth							
	second byte of Xwidth [optional]							
	first byte of Ywidth							
	second byte of Ywidth [optional]							
	1	1	1	1	xx		yy	
	first byte of X-coordinate							
	second byte of X-coordinate [optional]							
	first byte of Y-coordinate							
	second byte of Y-coordinate [optional]							

Sub-block Body

Image number : 1 byte specify the image in the Bmp Chunk

Image size : specify the Xwidth Ywidth for the image size

Paste location coordinates : use 2~5 byte Coordinates expression

When

Xwidth = 0

Ywidth = 0

the respective size will be the LCD size.

Tile the specified image from the base location, up to the same size as the physical display device.

The method of arrangement is the same as in the Image Tile definition.

2.) Auxiliary Sub-block

Auxiliary Sub-block is a data expression used to add various embellishments to the display object specified in the Primary Sub-block.

In the case of an Display Object Event that does not include an Auxiliary Sub-block, the contents of the Primary Sub-block are simply displayed for the specified time (=LifeTime), and then erased.

The Auxiliary Sub-block is a sub-sequence that can be used to modify the display content of a display object, move its display location, or change its display area.

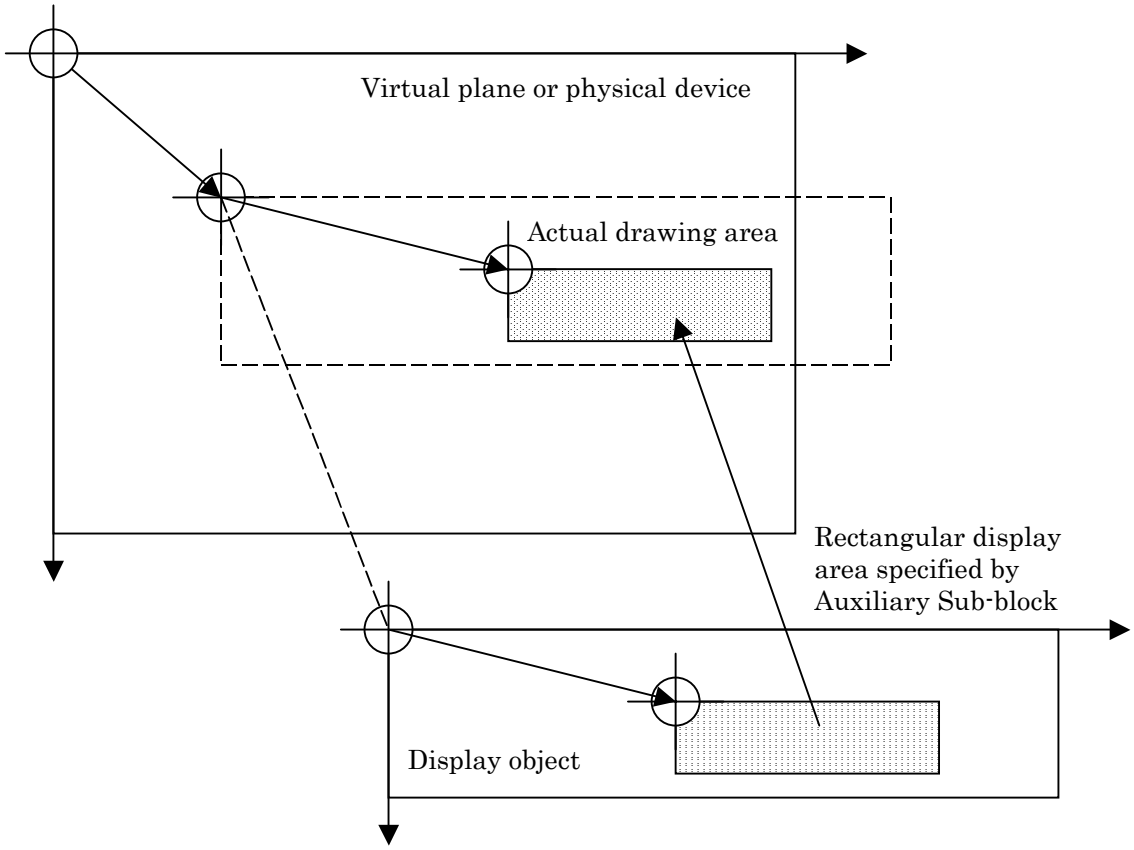
Auxiliary Sub-blocks can include

- Changes in display parameters
- Sequences that change text color
- Sequences for banner display
- Sequences that move the display location
- Blinking sequences
- Fill-InOut sequences

Combinations of operations can be expressed by listing multiple sub-blocks.

However, it is not possible for multiple instances of the same sub-block type sequence to be listed in parallel.

Sub-block Type	Description
0x80	Parameter override
0x81	Wipe Timing
0x82	WipeSeq
0x83	BannerInfo
0x84	TravelSeq
0x85	BlinkSeq
0x86	ColorBlinkSeq
0x87	Fill-InOutSeq
0x88~FF	Reserved



Conceptual diagram of specifying the display object and display area

- Parameter override

A data expression for overriding the display parameter specified for each event type. Use this when you wish to change display parameters (e.g., display color or font) even for event types that are defined in the Display Parameter Definition Chunk.

Sub-block Type := 0x80:Parameter override
Sub-block Body :=
Repeat {
prmID :1 byte
Value :1 byte
}

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x80							
Size	variable length							
Body	[subsequently, repetitions in 2-byte units]							
	prmID							
	Value							

PrmID	Description
0x01	Font Type
0x02	Font Size
0x03	Direction (text direction)
0x04	Attribute (text attribute; future expansion)
0x05~0x0F	Reserved
0x10	Font Color 0 (text color)
0x11	Font Color 1 (text color after change)
0x12	Edge Color 0 (text edge color; future expansion)
0x13	Edge Color 1 (edge color after change; future expansion)
0x14	Back Color 0 (text background color)
0x15	Back Color 1 (text background color after change)
0x16~0x1F	Reserved
0x20	Coordinates (default coordinate type)
0x21~0x2F	Reserved
0x30	BackDropColor (backdrop color; specifies Plane 0 color)
0x31	Transparent Color
0x32	Transparent Enable
0x33~0xFF	Reserved

- WipeTiming

The display color of a display object can be changed during its period of display.
The time for the color change is specified as sequence data.
When the time specified here has elapsed following the display start time, the color of the entire character string will be changed at once.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x81							
Size	0x01~0x02							
Body	first byte of WipeTime							
	second byte of WipeTime [optional]							

Sub-block Type := 0x81:WipeTiming

Sub-block Body :=

WipeTime :1~2byte use Duration expression.

WipeTime is specified using TimeBase as the time base.
Specify the relative time from the start time of the display event.

When WipeTiming is specified to TextBlock, the text portion excluding a background color portion for all in TextBlock is changed the color.

- WipeSeq

The text color can be changed per one character during the display of a display object.
Specifies more than one of the time to change color.
Whenever the specified time passes since the display start time, changes the color of one character in order from the initial character of a display object.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x82							
Size	Variable length							
Body	The 1st byte of Entry							
	The 2nd byte of Entry [Option]							
	The 1st byte of Timing Data							
	The 2nd byte of Timing Data [Option]							
	~							
	The following, repeats Timing Data for the number of Entry.							

Sub-block Type := 0x82:WipeSeq

Sub-block Body :=

Entry :1 ~ 2byte Adopts Duration expression.
Specifies the number of Timing Data.

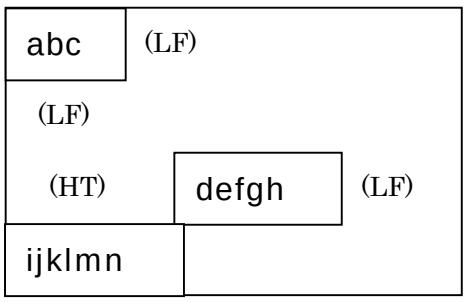
Timing Data:1 ~ 2byte Adopts Duration expression. Repeats for the number of Entry.

Timing Data specifies TimeBase as the reference time.
Each Entry of Timing Data is the relative time to color change of the following letter after the color change of a letter.
Timing Data of the head of Entry is the time from the display start time of an event to the color change of the first letter.

The corresponding indication Event is Text and TextBlock.
In the case of Timing Data = 0x00, changes color simultaneously with a front letter.
Correspondence of the number of Entry and the number of Data

The number of Entry=< Data Interprets Data of the number of Entry.
The number of Entry> Data Interprets all Data and reduces the number of Entry.
Furthermore, the followings are also interpreted.
Correspondence of the number of Entry and the number of letters of Display Event
The number of Entry =< letters Changes the color of the corresponding number of letters. The color of remained letters are not changed.
The number of Entry > letters The remained Entry is ignored.

When WipeSeq is specified to be TextBlock, counts the number of letters of real text portion except the control code from a head, and changes the color of the correspondence portion.



No.	Data	Time to change color	Letter to change color
Entry 0	1 sec	1 sec	a
Entry 1	1 sec	2 sec	b
Entry 2	1 sec	3 sec	c
Entry 3	3 sec	6 sec	d
Entry 4	0 sec	6 sec	e
Entry 5	0 sec	6 sec	f
Entry 6	0 sec	6 sec	g
Entry 7	1 sec	7 sec	h

Control code is not the object of color change.
The color of "defg" is changed simultaneously after 6 seconds from the display start.
The color of "ijklmn" is not changed in this case.

The example of TextBlock color change

- BannerInfo

A display object can be displayed as a banner during its display period.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x83							
Size	0x07~0x12							
Body	0	0	0	0	BannerType			
	BackColor							
	first byte of WindowSpan							
	second byte of WindowSpan [optional]							
	first byte of EffectiveSpan							
	second byte of EffectiveSpan [optional]							
	first byte of InitialPosition							
	second byte of InitialPosition [optional]							
	first byte of BannerTravel							
	second byte of BannerTravel [optional]							
	first byte of BannerTime							
	second byte of BannerTime [optional]							

Sub-block Type := 0x83:BannerInfo.

Sub-block Body :=

BannerType :1 byte

Bit3 = 0: horizontal banner, 1: vertical banner

Bit2 = 0: positive direction, 1: negative direction

Positive direction refers to a banner display object that moves from right to left or down to up, as seen when facing the screen.

Bit1 = wrap in negative direction (at left edge or top edge) 0: no, 1: yes

Bit0 = wrap in positive direction (at right edge or bottom edge) 0: no, 1: yes

Bits 4-7 are always 0, and BannerType = 0x10...ff are reserved for future use

BackColor :1 byte

Color of the portion that has no display image. (Valid only for an Image Chunk data display object.)

WindowSpan :1 ~ 2bytes use Duration expression.

Width/height of the display area.

If 0 is specified, the banner display sequence (Auxiliary Sub-block) will be ignored.

The WindowSpan value changes the ObjectWidth used to determine the coordinates when using layout coordinates.

Effective Span :1 ~ 2 bytes use Duration expression.

Add this when you wish to change the actual size of the image and the frequency at which the banner will repeat.

Effective Span must be set to a value that is larger than the actual image size.

If you specify a value less than the original image size (including 0), it will be interpreted as if you had specified a value matching the original image size.

In the case of a horizontal banner, this specifies the effective width of the original image in the horizontal direction. In the case of a vertical banner, this specifies the effective height in the vertical direction.

InitialPosition :1 ~ 2 bytes use Coordval expression.

Initial display position.

BannerTravel :1 ~ 2 bytes use Duration expression.

The distance of movement until banner display ends. If 0, considered as unspecified.

BannerTime :1~2 bytes use Duration expression.

The time required to move the display to the position specified by BannerTravel. If this time is exceeded, the image displayed in the display area will stop and remain displayed. BannerTime is specified relative to TimeBase.

If BannerTravel = 0, BannerTime is understood as specifying the time required to move 100 pixels, and banner display will continue repeatedly at this speed without stopping until the display erasure time.

Basic concepts of banner display

A banner is defined by sending the display object expressed in the Primary Sub-block to the "display area," or by specifying a time change for the location of the image that is being sent. For distance and size, the units are in single pixels of the display device.

For convenience of explanation, we define a "banner location" method of specifying the location. The banner location is coordinates that specify the base location when sending the image to the display area. We define the banner location to be the upper left coordinates when the rectangle corresponding to the destination display area is placed on the source image. The coordinates are expressed as local coordinates based at the upper left of the image.

The banner location can also be specified outside of the image.

If wrap-around occurs, it is assumed that the original image is tiled seamlessly, and the image will be sent to the display area. In other words, the banner location will wrap around in units of the image size.

If wrap-around does not occur, it will be painted over as an area that has no image.

If the display object is a binary-specified image such as Text, Bitmap, Text Block, or Bitmap Tile, then it will be painted over using the text background color of that display object. The BackColor will be ignored.

If the display object is an Image or Image Tile, it will be painted over by the BackColor within the banner display Sub-block.

Banner Type can be used to restrict the way in which the original image is wrapped. Use Banner Type to specify the following functions.

□ Banner direction

Horizontal banner

Project the image in a display area of the same height as the image, and display it while moving horizontally.

Right-to-left movement of the display object on the screen is defined as the positive direction, and left-to-right movement is defined as the negative direction.

Vertical banner

Project the image in an display area of the same width as the image, and display it while moving vertically.

Upward movement of the display object on the screen is defined as the positive direction, and downward movement is defined as the negative direction.

Definition of image wrap-around method

Negative direction wrap-around (enable/disable)

Specify whether the image will wrap-around when the banner location becomes a negative value.

Positive direction wrap-around (enable/disable)

Specify whether the image will wrap-around when the banner location becomes greater than the image size. If wrap-around does not occur, this means that there is no original image.

If there is no wrap-around directive and the banner displays that area, it will be painted over as an area containing no image, as described below.

If the display object is an binary image such as Text, Bitmap, Text Block, or Bitmap Tile, it will be painted over using the text background color of that display object. The BackColor will be ignored.

If the display object is an Image or Image Tile, it will be painted over using the BackColor of the banner display sub-block.

□ Definition of effective image size

When an image is repeated displayed as a banner, there will be cases in which you would like to allow an interval between repetitions. In such cases, you can specify the "effective size" of the image, and treat it as being larger than it actually is.

"Effective size" must be larger than the original image. If the portion extending beyond the original is being sent to the display area, it will be painted over as follows, since no original image exists.

If the display object is a binary image such as Text, Bitmap, Text Block, or Bitmap Tile, it will be painted over using the text background color of that display object.

The BackColor will be ignored.

If the display object is an Image or Image Tile, it will be painted over using the BackColor in the banner display Sub-block.

In the case of text, space characters can be inserted to obtain a visually identical result. However, it will occupy more memory area as a display object.

If the EffectiveSpan is less than the size of the original image, the banner display sequence will be ignored.

□ Banner distance and required time

The banner distance is the absolute value difference between the banner locations between the banner start and end times.

Required time is the time required for moving the specified banner distance.

After the banner has moved the specified banner distance, it will stop.

If the banner distance is specified as 0, the banner will not stop. (It will continue moving until the display erasure time.)

If the banner distance is specified as 0, the required time will be considered to be the time for 100 pixels.

Special values

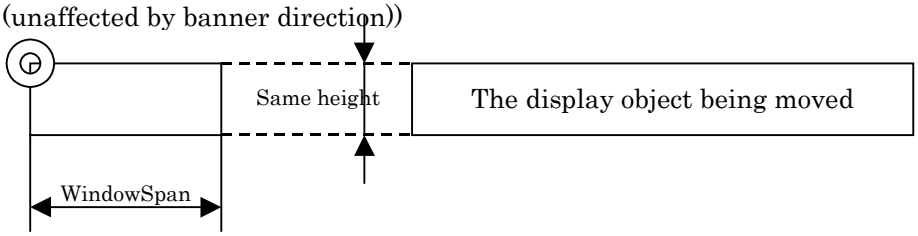
If WindowSpan = 0x00, the banner display sequence will be ignored.

If EffectiveSpan is smaller than the original image size, the banner display sequence will be ignored.

If BannerTime = 0x00, the banner display sequence will be ignored.

Concept of horizontal banner

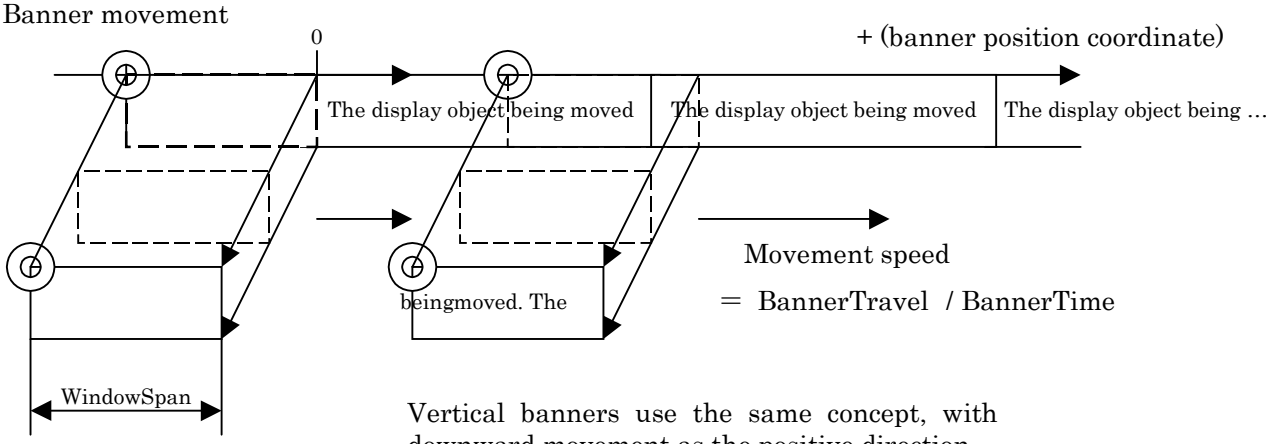
On-screen display area (the upper right circle is the base location for banner placement (unaffected by banner direction))



Example of positive-direction banner with positive-direction wrap-around
EffectiveSpan ＝0
InitialPosition ＝－WindowSpan

In the case of a positive-direction banner, the banner location moves the local coordinate system from left to right in the positive direction. As a result, the object will move from right to left on the screen.

Banner movement



Vertical banners use the same concept, with downward movement as the positive direction.

Move to on-screen display area

Negative direction (banner location movement corresponding to the banner direction) Positive direction

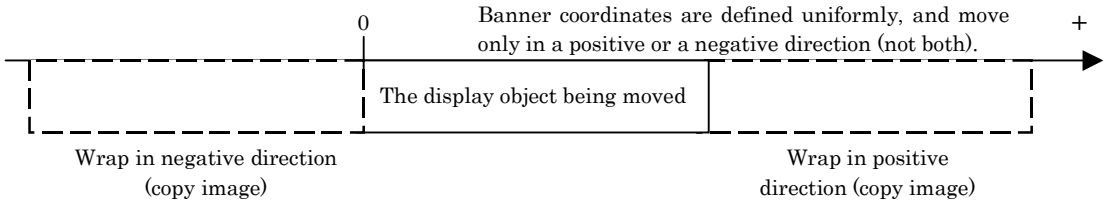
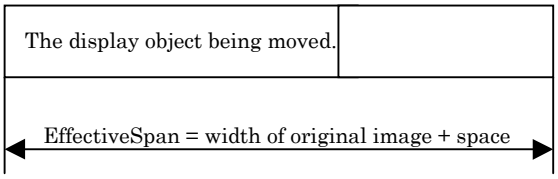


Image wrap-around wraps the size that includes the EffectiveSpan.
EffectiveSpan is always added in the positive direction of the banner coordinates. (Right side or lower side.)



Conceptual diagram for horizontal banner

- TravelSeq

The display object can be moved while it is being displayed.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x84							
Size	variable length							
Body	Step							
	first byte of number of Entries							
	second byte of number of Entries [optional]							
	[following is repeated according to the number of Entries]							
	1	1	1	1	xx		yy	
	first byte of target x-coordinate							
	second byte of target x-coordinate[optional]							
	first byte of target y-coordinate							
	second byte of target y-coordinate [optional]							
	first byte of arrival time							
	second byte of arrival time [optional]							
	[repeated]							

Sub-block Type := 0x84:TravelSeq

Sub-block Body :=

Step :1 byte

Display update interval. 0:= as smoothly as possible, 1...:= update time interval

Number of entries :1 - 2 byte use Duration expression.

Structure of each Entry{

Target coordinates :2 - 5 bytes use Coordinates expression.

Arrival time :1 - 2byte use Duration expression.

} repeat by the "Number of entries"

Arrival time is specified relative to TimeBase.

Basic concept

Update the display of the moving sequence specified by Entry, applying linear interpolation to determine the coordinate at which the display object should be according to the Step data.

The arrival time is specified in units of the TimeBase. The moving sequence is listed as relative time based on the starting time of the display event. The arrival time specifies the time interval for coordinate movement. Use Duration expression for the arrival time.

Correspondence of the number of Entry and the number of Data

The number of Entry=< Data Interprets Data of the number of Entry

The number of Entry> Data Interprets all Data and reduces the number of Entry.

Special values

If the arrival time is 0x00, the original position for that entry time will be erased, and the object will be understood to have moved to a new position. (Instantaneous movement.) However, two or more consecutive 0x00 arrival times will be ignored for the second and following entries. If the target coordinates are the same as the preceding or following entries, the object will be stopped at that location for the corresponding time.

- BlinkSeq

While a display object is being displayed, it can be made to blink in the specified on/off cycle.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x85							
Size	0x03-0x06							
Body	first byte of OnTime							
	second byte of OnTime [optional]							
	first byte of OffTime							
	second byte of OffTime [optional]							
	first byte of Refrain							
	second byte of Refrain [optional]							

Sub-block Type := 0x85:BlinkSeq

Sub-block Body:=

OnTime :1 - 2 bytes Erase the display after this time has elapsed. Relative time from the display event start time. Use Duration expression.

OffTime :1 - 2 bytes Resume display after this time has elapsed. Use Duration expression.

Refrain :1 - 2 bytes Specify the number of times that the state will change. If you specify 0, the change will continue repeating for the LifeTime of the display object. Use Duration expression.

Both OnTime and OffTime are specified relative to TimeBase.

Basic concept

Begin the sequence with the display object displayed from the start time of the display event. When the OnTime has elapsed, change the state to non-display. Then when OffTime has elapsed, change the state back to display.

Refrain specifies the number of times that the state will change. If you specify 0, the cycle will continue for the duration of the LifeTime.

Special values

The blinking display sequence will be ignored if either OnTime or OffTime is 0x00.

- ColorBlinkSeq

  While a display object is being displayed, it can be made to alternate colors in the specified on/off cycle.

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x86							
Size	0x03~0x06							
Body	first byte of OnTime							
	second byte of OnTime [optional]							
	first byte of OffTime							
	second byte of OffTime [optional]							
	first byte of Refrain							
	second byte of Refrain [optional]							

Sub-block Type := 0x86:ColorBlinkSeq

Sub-block Body :=

OnTime :1 ~ 2 bytes Change the color after this time has elapsed. Relative time from the display event start time. Use Duration expression.

OffTime :1 ~ 2 bytes Restore the color after this time has elapsed. Use Duration expression.

Refrain :1 ~ 2 bytes Specify the number of times that the state will change. If you specify 0, the change will continue repeating for the LifeTime of the display object. Use Duration expression.

Both OnTime and OffTime are specified relative to the TimeBase.

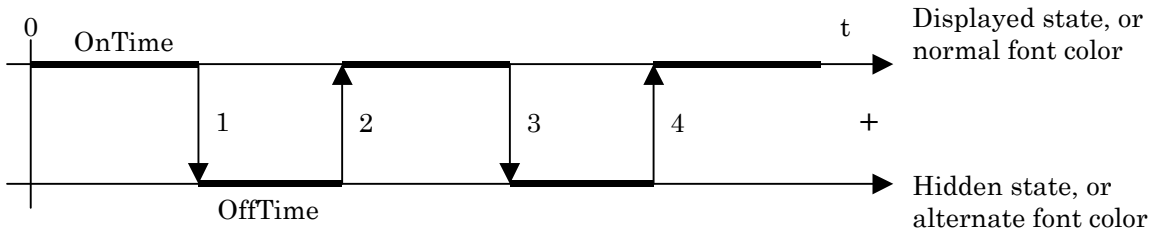
Basic concept

Begin the sequence with the display object displayed in the font color and font background color, from the start time of the display event. When the OnTime has elapsed, change the state to the alternate font color and alternate font background color. Then when OffTime has elapsed, change the state back to the original color. Refrain specifies the number of times that the state will change. If you specify 0, the cycle will continue for the duration of the LifeTime.

Special values

The color blink sequence will be ignored if either OnTime or OffTime are 0x00.

Concepts of OnTime, OffTime,
and Refrain in Blink sequences
and Color Blink sequences



If Refrain = 0, continue alternating during LifeTime

Conceptual diagram of Blink and Color Blink sequence

- Fill-InOutSeq

Sub-block	b7	b6	b5	b4	b3	b2	b1	b0
Type	0x87							
Size	0x04~0x07							
Body	Fill-InType				Fill-OutType			
	first byte of Fill-InTime							
	second byte of Fill-InTime [optional]							
	first byte of KeepTime							
	second byte of KeepTime [optional]							
	first byte of Fill-OutTime							
	second byte of Fill-OutTime [optional]							

Sub-block Type := 0x87:Fill-InOutSeq

Sub-block Body :=

ShutterType :1 byte Specify the display start effect and the display end effect for the object.

Fill-InTime :1 ~ 2 bytes Specify the time until the display object is fully displayed.

KeepTime :1 ~ 2 bytes Specify the time that the display object will remain fully displayed.

Fill-OutTime :1 ~ 2 bytes Specify the time until the display object is fully hidden.

Fill-InTime, KeepTime, and Fill-OutTime each use Duration expression.

Fill-InTime, KeepTime, and Fill-OutTime are each specified relative to the TimeBase.

Fill-InType Fill-OutType	Description
0	No effect
1	Left-Horizontal In(Out)
2	Right-Horizontal In(Out)
3	Center-Horizontal In(Out)
4	Top-Vertical In(Out)
5	Bottom-Vertical In(Out)
6	Center-Vertical In(Out)
7	Left-Top-Corner In(Out)
8	Left-Bottom-Corner In(Out)
9	Right-Top-Corner In(Out)
10	Right-Bottom-Corner In(Out)
11	Center-Rectangle In(Out)
12…15	Reserved

The name of the operation is defined relative to the display start and display end positions.

Basic concepts

This describes a sequence that changes the display area of the display object between the display start time and display end time.

The upper 4 bits of ShutterType specify the type of display area change that will occur at the start of display, and the lower 4 bits specify the type of display area change that will occur at the end of display.

The display start time begins from a hidden state. When the Fill-In Time has elapsed, all of the display object will be visible.

The display end time is the Fill-InTime + KeepTime. From a completely visible state, the area will begin changing and will reach a hidden state when the Fill-OutTime has elapsed.

□ ShutterType

(In)

Horizontal change types

Left-Horizontal-In	Initiate display from the left edge of the display object toward its right edge.
Right-Horizontal-In	Initiate display from the right edge of the display object toward its left edge.
Center-Horizontal-In	Initiate display from the center of the display object toward its left and right edges.

Vertical change types

Top-Vertical-In	Initiate display from the top edge of the display object toward its lower edge.
Bottom-Vertical-In	Initiate display from the lower edge of the display object toward its top edge.
Center-Vertical-In	Initiate display from the center of the display object toward

its top and bottom edges.

Corner rectangular change types

Left-Top-Corner-In	Initiate display from the upper left corner of the display object toward its lower right corner.
Left-Bottom-Corner-In	Initiate display from the lower left corner of the display object toward its upper right corner.
Right-Top-Corner-In	Initiate display from the upper right corner of the display object toward its lower left corner.
Right-Bottom-Corner-In	Initiate display from the lower right corner of the display object toward its upper left corner.
Center-Rectangle-In	Initiate display from the center of the display object in a rectangular shape.

(Out)

Horizontal change types

Left-Horizontal-Out	Erase from the right edge of the display object toward its left edge.
Right-Horizontal-Out	Erase from the left edge of the display object toward its right edge.
Center-Horizontal-Out	Erase from the left and right edges of the display object toward its center.

Vertical change types

Top-Vertical-Out	Erase from the lower edge of the display object toward its upper edge.
Bottom-Vertical-Out	Erase from the upper edge of the display object toward its lower edge.
Center-Vertical-Out	Erase from the upper and lower edges of the display object toward its center.

Corner rectangular change types

Left-Top-Corner-Out	Erase from the lower right corner of the display object toward its upper left corner.
Left-Bottom-Corner-Out	Erase from the upper right corner of the display object toward its lower left corner.
Right-Top-Corner-Out	Erase from the lower left corner of the display object toward its upper right corner.
Right-Bottom-Corner-Out	Erase from the upper left corner of the display object toward its lower right corner.
Center-Rectangle-Out	Erase from the periphery of the display object toward its center in a rectangular shape.

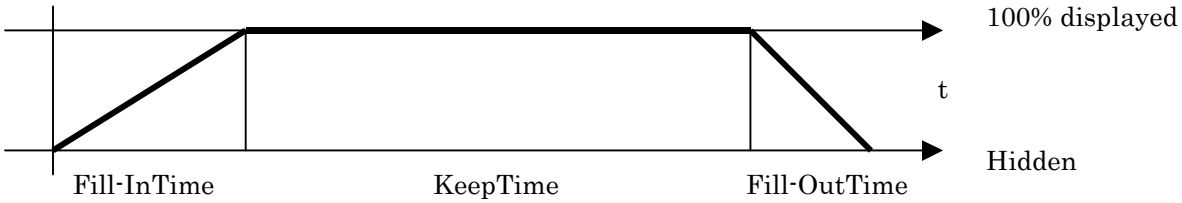
Special values

If the ShutterType is 0x00 (neither specified), the Fill-InOut sequence will be ignored.

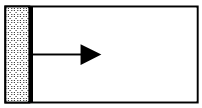
If the ShutterType is 0x0nor 0xn0, the 0 item will be interpreted as unspecified, and only the display initiation or the erasure will be performed, adding it to the KeepTime but ignoring the Fill-InTime and Fill-OutTime values.

If the Fill-InTime or Fill-OutTime are 0x00, this will be interpreted as an unspecified Time 0, regardless of the ShutterType.

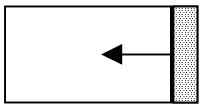
Concept of elapsed time



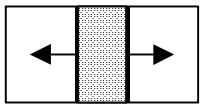
ShutterType concept



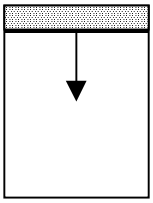
Left-Horizontal-In



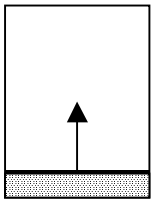
Right-Horizontal-In



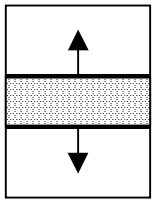
Center-Horizontal-In



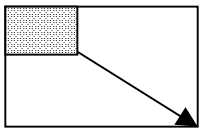
Top-Vertical-In



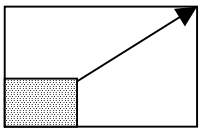
Bottom-Vertical-In



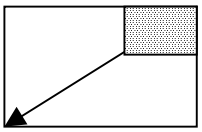
Center-Vertical-In



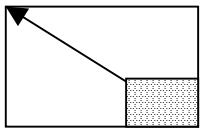
Left-Top-Corner-In



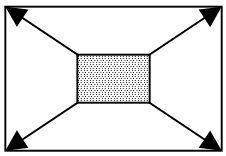
Left-Bottom-Corner-In



Right-Top-Corner-In

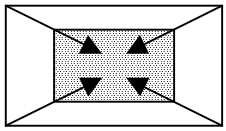


Right-Bottom-Corner-In

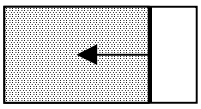


Center-Rectangle-In

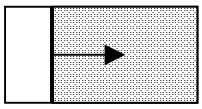
If
ShutterType = 0
Fill-InTime = 0
Fill-OutTime = 0
then Fill-InOut is taken as unspecified.



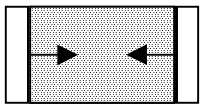
Center-Rectangle-Out



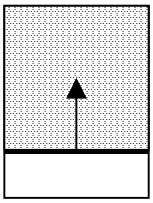
Left-Horizontal-Out



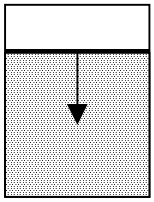
Right-Horizontal-Out



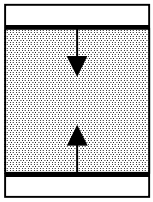
Center-Horizontal-Out



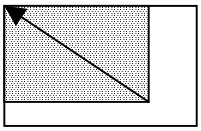
Top-Vertical-Out



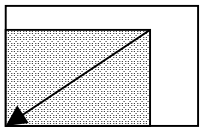
Bottom-Vertical-Out



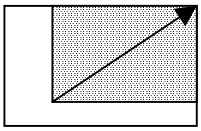
Center-Vertical-Out



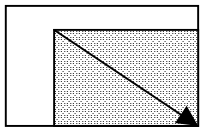
Left-Top-Corner-Out



Left-Bottom-Corner-Out



Right-Top-Corner-Out



Right-Bottom-Corner-Out

3.) Applicability and interpretation of Display Parameters

- Default settings for the SMAF file

  In the Display Parameter Definition Chunk (§6.2.1), the specification of Event Type = 0x00(required) shall be the default setting for SMAF files.

  The default setting is valid for the same Graphics Track Chunk.

- Individual settings for each event type

  By making definitions relative to the default settings in the same Display Parameter Definition Chunk, it is possible to overwrite settings for each event type.

  If an event type in the sequence data for which such a definition exists is specified, those individual settings will be selected.

  Individual settings for each event type are valid within the same Graphic Track Chunk.

- Event settings using the Parameter Override Sub-block

  The Parameter Override Sub-block can be used to make settings that override the event settings in the sequence data. (They will apply before the event content is interpreted.)

  Parameter Override applies only to everything (Event, Primary Sub-block, Auxiliary Sub-block) from the beginning of the event to the end.

- Settings for individual data(Coordinates)

  Settings for normal display parameters can be modified only by the settings described above.

  For coordinates, individual data can also be overridden for the above three settings.

  Coordinates that specify the display location of an event are interpreted after Parameter Override, and if there is a change, it will be valid from that point until the end of the event.

  If there is a change in the individual settings of sequence data in the Auxiliary Sub-block, it will be valid from that point until the next setting within the same sub-block or until the end of that sub-block.

  (The same event in the next sub-block will start from the event settings.)

- Operating rules for coordinate settings

  In order to avoid the indiscriminate use of settings that can be individually overwritten, the SMAF content production guidelines or the SMAF playback system implementation guidelines will provide rules for use.

4.) Sub-block support table

The following table shows which data expressions are supported by the file format.

Auxiliary / Primary	Text	Bmp	Image	Rectangle	TextBlock	ImageTile	BitmapTile
Parameter Override	0	0	0(some)	0(some)	0	0(some)	0(some)
WipeTiming	0	0	-	-	0	-	-
WipeSeq	0	-	-	-	0	-	-
BannerInfo	0	0	0	0	0	0	0
TravelSeq	0	0	0	0	0	0	0
BlinkSeq	0	0	0	0	0	0	0
ColorBlinkSeq	0	0	-	-	0	-	-
Fill-InOutSeq	0	0	0	0	0	0	0

(For some, refer to the following Display Parameter support table.)

Details of operational support (including the playback system and the content) are given separately in the SMAF playback system implementation guidelines and in the content production guidelines.

5.) Display Parameter support table

File format support for various data expressions is shown by the following table.

Display Parameter	Text	Bmp	Image	Rectangle	TextBlock	ImageTile	BitmapTile
Font Type	0	-	-	-	○	-	-
Font Size	0	-	-	-	○	-	-
Direction	0	-	-	-	○	-	-
Attribute	0	-	-	-	○	-	-
Font Color 0	0	0	-	-	○	-	0
Font Color 1	0	0	-	-	○	-	0
Edge Color 0	0	-	-	-	○	-	-
Edge Color 1	0	-	-	-	○	-	-
Back Color 0	0	0	-	-	○	-	0
Back Color 1	0	0	-	-	○	-	0
Coordinates	0	0	0	0	0	0	0
BackDrop Color	-	-	-	-	-	-	-
Transparent Color	0	0	0	0	0	0	0
Transparent Enable	0	0	0	0	0	0	0

Details of operational support (including the playback system and the content) are given separately in the SMAF playback system implementation guidelines and in the content production guidelines.

6.4. Font Data Chunk

Chunk ID	"Gftd"	:Font Data Chunk

The content of the Font Data Chunk is an array of Font Chunks containing "gaiji" (externally defined character) bitmaps.

6.4.1. Font Chunk

Chunk ID	"Ge**"	:Font Chunk

This is the gaiji data resource chunk.

The Font Chunk combines FontType and FontSize into a single chunk, and is searched for by its Code and used.

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	0x47 (“G”)							
Data#1	0x65 (“e”)							
Data#2	FontType							
Data#3	FontSize							

The Font Chunk consists of

Header

FontData array of gaiji data (variable length data array)

The Header consists of

Entry Number :1~2 bytes Number of gaiji in the chunk. Use Duration expression.

The data structure for each gaiji character is

FontData {

Code :2 byte

Size :1~2byte Use Duration expression.

Bitmap :variable length

}

FontData is repeated for the number of gaiji.

FontType and FontSize search for the specified chunk using the value specified by Display Parameter of the display object. Displayed using the corresponding code within the chunk.

The format does not define the horizontal dimension dot size of the gaiji bitmap within the chunk.

The playback system will successively display the bitmap it obtains, without tightening the spacing of the dots.

For an actual portable terminal device, the initial gaiji operating rules will be specified in the implementation guidelines.

The data format definition for the bitmap used is given in Bmp Chunk.

6.4.2. Unicode Font Chunk

Chunk ID	"Gu**"	:Unicode Font Chunk

This is the resource Chunk of originalcharacter font data.

Font Chunk is summarized to one Chunk by FontType and FontSize, and search and uses it by Code.

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data#0	0x47 (“G”)							
Data#1	0x75 (“u”)							
Data#2	FontType							
Data#3	FontSize							

Font Chunk consists of the following.

Header

FontData The array of original-character-font data (Variable-length-data array)

Header consists of the following.

EntryNumber :1 ~ 2byte The number of original character fonts in Chunk adopts Duration expression.

The data structure for each original character is as follows.

FontData {

Size :1 ~ 2byte Adopts Duration expression.

Code : Variable length

Size :1 ~ 2byte Adopts Duration expression.

Bitmap : Variable length

}

FontData is repeated the times of the number of original character.

FontType and FontSize search the specified Chunk with the specified value of Display Parameter of a display object. Displays using the relevant code in Chunk.

Set BOM (byte-order mark) as each Code head.

When no BOM, interpret as big endian.

On the format, the dot size of width direction of the original character Bitmap in Chunk is not specified.

Play system displays the obtained Bitmap in order continuously without the space of dots.

Application rule of the original-character on the actual mobile is specified by the installation guideline.

The definition of the data format of Bitmap to be used is described to Bmp Chunk.

6.5. Image Data Chunk

Chunk ID "Gimd" :Image Data Chunk

The content of the Image Data Chunk is a array of image data chunks.

There are three types of image data chunk, as follows.

Image Chunk

Bmp Chunk

Link Chunk

The lowest byte of the ChunkID of an image data chunk has a unique number within the image data chunk. This is the "image ID" specified by the Display Object Event.

6.5.1. Image Chunk

Chunk ID "Gig*" :Image Chunk *=0x00 ~ 0xFF

This contains data of general purpose image formats such as JPEG or PNG.

Image data is used by specifying Image as the Primary sub-block type.

6.5.2. Bmp Chunk

Chunk ID "Gbm*" :Bmp Chunk *=0x00 ~ 0xFF

This contains bitmap (binary image) data.

Bitmap data is used by specifying Bitmap as the Primary sub-block type.

Bitmap data format

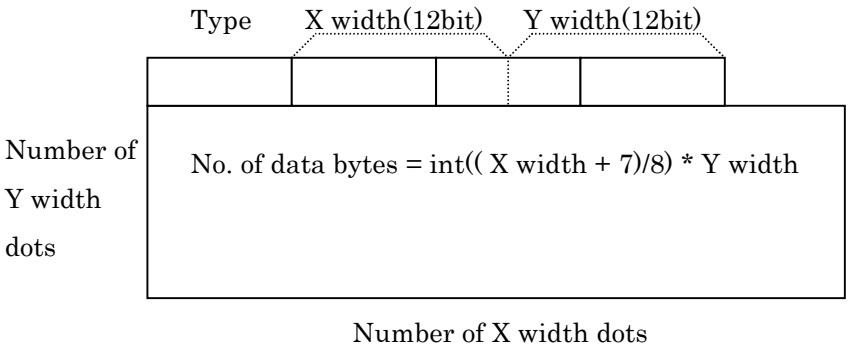
Type (1 byte)

Size (3 bytes = X direction dot number, 12 bit expression; Y direction line number, 12 bit expression)

Data (definition by 1 bit per dot; variable length by size)

Type	Description
0x00	Non-compressed
0x01	8 bit Run Length compression
0x01~FF	Reserved

Size	b7	b6	b5	b4	b3	b2	b1	b0
Data #1	Xwidth (upper 8 bits)							
Data #2	Xwidth (lower 4 bits)				Ywidth (upper 4 bits)			
Data #3	Ywidth (lower 8 bits)							



The data MSB specifies the upper left corner of the screen.

Fill 0 in any remaining bits at the end of the file.

On a PC that uses SMAF, binary image files shall have a filename extension of *.bms (BitMap of SMAF).

Run Length compression algorithm (8 bit Run Length)

Viewing the data in byte units,

If there are 'n' repetitions of the same data (when n is 2~129),

byte ^(n - 2)

byte val

・If there are 'n' repetitions of different data (n is 1 ~ 128),

byte n - 1

byte val1, val2, val3..., valn

Ex.) 00 00 00 00 00 05 05 01 02 ==> fc 00 ff 05 01 01 02

6.5.3. Link Chunk

Chunk ID	"Gln*"	:Link Chunk	*=0x00 ~ 0xFF

This is used to specify images outside the SMAF file.

Detailed definition is to be determined.

7. Basic concepts of the Master Track

7.1. Position of the Master Track

In SMAF data, the master track is a data expression expressing a sequence that controls musical data sequences and the SMAF playback system itself.
The master track is one of the tracks that make up SMAF, but unlike the Score Track PCM Track or Graphics Track, it does not have a corresponding output device.

7.2. Basic structure of the Master Track

The master track consists of

- Header
- Sequence Data Chunk

In the same way as the sequence data of other tracks, the Sequence Data Chunk consists of Durations, Events, and EOS.

The content expressed by events can be broadly divided into the following two types.

Event types

- Musical data events
- Performance control events

7.2.1. Musical data events

Musical data events are events corresponding to chord names, key signatures, time signatures, tempo, bar lines, and rehearsal marks.
By attaching data events in synchronization with a musical sequence being played, it is possible to express the musical structure, or obtain musical information for that time.

7.2.2. Performance control events

Performance control events are events that control the SMAF playback system itself.
For example, this provides for events such as Pause or Jump (branching).

(Implementation example)
In order to support musical control events, the playback system must implement GUI processing such as Seek functionality and key input processing.

For example if the playback system processes a Pause event on the master track, playback would pause at that time. Playback would resume when any key was input from the GUI. It would also be possible to Seek for a time that was input.

Details are not defined by this specification manual. This is for future expansion.

A SMAF (display) playback system that implements the contents of this specification manual will not interpret sequence data of the Master Track Chunk.

7.2.3. Guidelines for musical data events

Keep the size of the master track for a single song to within 10~20% of the Score Track.
As far as possible, allow simple conversion from the data expressions of the XF format.
XF->Master Track conversion must be easy. The opposite conversion need not be possible.

Content that should be expressed, and operating rules

TimeSignature

TimeSignature must be specified at the beginning of the song.

TimeSignature may be changed only the timing of a bar line.

If it is changed at a timing location other than a bar line, it will be considered to have been specified at the bar line that follows.

Key Signature

Restrictions on the timing of KeySignature are the same as for TimeSignature.

Tempo

If we assume that the minimum tempo is 40, this can be expressed by 21 bits. Expressible as 7 bit x 3.

The SMF data expression is 24 bit, and to avoid incorrect conversion, we also define a 7 bit x 4 = 28 bit expression.

Restrictions on the timing of tempo changes are the same as for TimeSignature.

Measure Mark

In order to reduce the amount of data, we record only the location of the measure marks. The beat timing can be determined from the TimeSignature and the location of the measure marks.

Chord Name

The format is based on the XF chord names, but normally omits the fractional chords to reduce the amount of data.

The chord names that can be specified in operation are defined separately.

8. Master Track data format

8.1. Master Track Chunk

The Master Track contains information that allows a sequencer to determine the structure and data of the song, and controls the flow of time. The following functionality is planned for expansion.

Chunk ID	"MSTR"	:Master Track Chunk

The internal structure of the Master Track is as follows.

- Header
- Sequence Data Chunk

Header contents

- FormatType :1 byte
- Sequence Type :1 byte
- TimeBase_D :1 byte
- OptionSize :1 byte
- OptionData :size specified by OptionSize(0~255byte)

1. Format Type

This defines the master track data format that is used.

Format Type	Description
0x00	Handy Phone Standard
0x01~0xFF	Reserved

2. Sequence Type

Sequence Type	Description
0x00	Stream Sequence
0x01	Sub-Sequence
0x02~0xFF	Reserved

Stream Sequence

The sequence data of the score track is expressed as one stream of sequence data.

Sub-Sequence

The sequence data of the score track is expressed as a succession of multiple instances of phrase data.

3. TimeBase_D

TimeBase_D	Description
0x00	1 msec
0x01	2 msec
0x02	4 msec
0x03	5 msec
0x04~0x0F	Reserved
0x10	10 msec
0x11	20 msec
0x12	40 msec
0x13	50 msec
0x14~0xFF	Reserved

Duration based time in the Score Track follows TimeBase_D.

4. OptionSize

This specifies the size of the OptionData (for extension; see below). Normally set OptionSize:= 0x00.

The size of expansion will be a multiple of four.

5. OptionData

Describes data for future expansion. When interpreting, you must not assume that this is zero.

8.2. Master Track Sequence Data Chunk

Chunk ID	"Mssq"	:Master Track Sequence Data Chunk

Sequence Data expression

Sequence data is a variable-length byte stream.

The structural elements of sequence data are Durations, Events, and EOS.

The first element in sequence data must be a duration, and durations and events must occur alternately.

EOS signifies End Of Sequence. It is expressed by four or more consecutive bytes of 0x00.

When interpreting sequence data, you must check for EOS before interpreting events or durations.

The data format guarantees that no events or durations will have four or more consecutive bytes of 0x00.

Duration expression

It is the same as Duration expression of Handyphone Standard.

8.3. Event

Event data expression

Events are variable length, with one byte or greater.

MSB=0 for the last byte of variable-length elements, and MSB=1 for other bytes.

The end of an event is always determined by the appearance of MSB=0 data.

Even currently-known events may be expanded in the future, increasing their data length. (In no event will data lengths decrease.)

Unknown events may be skipped by reading ahead to the next MSB=0 data (including the MSB=0 data).

No upper limit on data length is defined.

No Operation Event (NOP Event)

Does nothing. Used to divide Duration, as in the Score Track or Graphics Track.

NOP Event = 0x00

8.3.1. Music data event

First-defined music data events are the following types.

1. Chord Name

The chord name is expressed as a root note (C...B), an accidental symbol (#, b), and a two-byte chord type.

This follows the XF expression.

Data Count	b7	B6	b5	b4	b3	b2	b1	b0
Data #1	1	0	accidental symbol			root note name		
Data #2	S	chord type (follows XF)						
[Data #3]	1	0	accidental symbol			root note name		
[Data #4]	0	chord type (follows XF)						

If you wish to express a fractional chord, list two chord names in succession, using four bytes for the expression.

S	Description
0	2 byte expression
1	4 expression of fractional chord

Data structure

First byte and third byte (3 bit accidental symbol and root note name)

Accidental (3 bit)	Description

0	bbb
1	bb
2	b
3	Natural
4	#
5	##
6	###
7	use is prohibited

Root note (3 bit)	Description
0	not used (XF Reserved)
1	C
2	D
3	E
4	F
5	G
6	A
7	B

Second byte and fourth byte (7-bit Chord Type)

Value	Type	Value	Type	Value	Type	Value	Type
0	Maj	9	min6	18	dim7	27	7(#9)
1	Maj6	10	min7	19	7th	28	Maj7aug
2	Maj7	11	Min7b5	20	7sus4	29	7aug
3	Maj7(#11)	12	Min(9)	21	7b5	30	1+8
4	Maj(9)	13	Min7(9)	22	7(9)	31	1+5
5	Maj7(9)	14	Min7(11)	23	7(#11)	32	sus4
6	Maj6(9)	15	MinMaj7	24	7(13)	33	1+2+5
7	aug	16	MinMaj7(9)	25	7(b9)	34	cc
8	min	17	dim	26	7(b13)	35~127	unspecified

2. KeySignature

KeySignature is expressed using 2 bytes.

Data structure

Data Count	b7	B6	b5	b4	b3	b2	b1	b0
Data #1	1	0	1	1	1	0	0	0
Data #2	0	0	0	m	number of accidentals			

Number of accidentals := number of # or b. Interpret as four-bit signed binary. >0 is number of #, and <0 is number of b.

m	Description
0	Major
1	Minor

3. Time Signature

Time Signature is expressed as 2 bytes.

Data structure

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #1	1	0	1	1	1	fff denominator		
Data #2	0	nnnnnnn numerator						

First byte := 0xB9~0xBD

Data fff	Denominator
0	use prohibited
1	2
2	4
3	8
4	16
5	32
6	undefined
7	undefined

If an undefined value is specified, the TimeSignature event will be ignored.

0 is prohibited, since it would be the same code as the KeySignature .

Second byte:= 0nnnnnnn

nnnnnnn: denominator of the time signature. Valid range is 1...64.

4. Tempo

Expresses the length of one beat in microsecond units. Data length is 4 or 5 bytes.

Data structure

Data Count	b7	B6	b5	b4	b3	b2	b1	b0
Data #1	1	1	1	1	0	0	0	0
Data #2	1	upper 7 bits						
Data #3	1	middle 7 bits						
Data #4	S	lower 7 bits						
[Data #5]	0	optional lowest 7 bits						

S	Description
0	4 byte expression
1	5 byte expression

Four-byte expression(use if beat length can be expressed in 21 bits or less)

Five-byte expression (use if beat length cannot be expressed in 21 bits or less)

Using a five-byte expression to express a tempo value expressible in four bytes is not forbidden, but cannot be recommended.

5. Measure Mark (bar line)

Indicates the division between measures. Expressed as one byte.

Data structure

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #1	0	1	1	1	0	0	0	1

First byte := 0x71

6. Rehearsal Mark

Indicates a rehearsal mark compatible with the XF format.

Data structure

Data Count	b7	b6	b5	b4	b3	b2	b1	b0
Data #1	0	1	0	0	xxxx			

First byte := 0100xxxx

xxxx: 0:Intro,1:Ending,2:Fill-in,3:A,....,15:M

Data xxxx	Rehearsal mark
0	Intro
1	Ending
2	Fill-in
3	A
4	B
….	….
….	….
15	M

8.3.2. Performance control events

Details of performance control events are undefined.

The Pause event should be defined first.

9. Appendix

9.1. Chunk ID & TAG list

SMAF basic functions: Chunk ID & TAG names

Track	Sub	S-Sub	TAG	Description
MMMD				Mobile Application Data Chunk
CNTI				Contents Information Chunk
			VN	Vendor name
			CN	Carrier name
			CA	Category name
			ST	Song title
			AN	Artist name
			WW	Lyricist
			SW	Composer
			AW	Arranger
			CR	Copyright©
			GR	Management group
			MI	Manegement information
			CD	Creation date
			UD	Revision date
			A0	Reserved
			A1	Reserved
			A2	Reserved
			SS	Snap shot
OPDA				Optional Data Chunk
	Dch*			Data Chunk
			VN	Vendor name
			CN	Carrier name
			CA	Category name
			ST	Song title
			AN	Artist name
			WW	Lyricist
			SW	Composer
			AW	Arranger
			CR	Copyright©
			GR	Management group
			MI	Manegement information
			CD	Creation date
			UD	Revision date
			ES	Status after edit
			VC	Virtial card
			IC	Image creater
			IR	Image copyright
			IP	Image editor
			TR	Text copyright
			TP	Text editor
			GP	Contents editor
			VE	Reserved
			CE	Reserved
			UI	Reserved
			OW	Reserved
			A0	Reserved
			A1	Reserved

			A2	Reserved
			SS	Snap shot
			LC	Locale infomation
			RF	Resource infomation
MTR*				Score Track Chunk
	MspI			Score Seek & Phrase Info
			st	Start
			sp	Stop
			Pa	Phrase "a"
				:
			Pz	Phrase "z"
			PA	A-melo
			PB	B-melo
			PE	Ending
			PI	Intro
			PK	KANSOU
			PS	SABI
			PR	Refrain
			SL	Sub-sequence List
	Mtsu			Score Track Setup Data Chunk
	Mtsq			Score Track Sequence Data Chunk
	Mtsp			Score Track Stream PCM Data Chunk
		Mwa*		Score Track Stream Wave Data Chunk
ATR*				PCM Audio Track Chunk
	AspI			PCM Audio Track Seek & Phrase Info Chunk
	Atsu			PCM Audio Track Setup Data Chunk
	Atsq			PCM Audio Track Sequence Data Chunk
	Awa*			PCM Audio Track Wave Data Chunk

(*) Uppercase and lowercase are distinguished.

SMAF extended functions: Chunk ID & TAG name list (continued)

Track	Sub	S-Sub	TAG	Description
GTR*				Graphics Track Chunk
			RS	Rendering Size
	Gtsu			Setup Data Chunk
		Gdpd		Display Parameter Definition Chunk
		Gcpd		Color Palette Definiton Chunk
	Gsq*			Graphics Track Sequence Data Chunk
	Gftd			Font Data Chunk
		Ge**		Font Chunk (Distinction of FontType/FontSize)
		Gu**		Unicode Font Chunk (Distinction FontType/Size)
	Gimd			Image Data Chunk
		Gig*		Image Chunk
		Gbm*		Bmp Chunk
		Gln*		Link Chunk
MSTR				Master Track Chunk
	Mssq			Master Track Sequence Data Chunk

(*) Uppercase and lowercase are distinguished.

9.2. CRC sample code

CRC sample code is given here in C.
This sample obtains the CRC value for a byte string 'c' of 'n' bytes.
This sample speeds execution by creating a table first.

```
#define CHAR_BIT      8         /* number of bits in a char */
#define UCHAR_MAX     0xff      /* maximum unsigned char value */
typedef unsigned charUint8;
typedef unsigned short  Uint16;
static Uint16 crctable[UCHAR_MAX + 1];
#define CRCPOLY1  0x1021U

     void  makeCRCTable(void)
{
        Uint16 i, j, r;
        for (i=0; I <= UCHAR_MAX; i++) {
           r = i << (16-CHAR_BIT);
           for (j=0; j < CHAR_BIT; j++) {
           if (r & 0x8000U) {
                         r = (r<<1) ^ CRCPOLY1;
                } else {
                         r <<= 1;
                }
           }
           crctable[i] = r&0xFFFFU;
        }
}

Uint16 makeCRC(int n, Uint8* c)
{
        Uint16  r;
        r = 0xFFFFU;
        while (--n >= 0) {
           r = (r << CHAR_BIT) ^ crctable[(Uint8)(r >> (16 - CHAR_BIT)) ^ *c++];
        }
   return ~r & 0xFFFFU;
}
```

9.3. BNF notation

BNF notation of SMAF is described below.

```
;-------------------------------------------------------------------------
;     SMAF  (BNF notation)
;
;  * : 0 times or more, + : 1 time or more, ? : 0 or 1 time
;-------------------------------------------------------------------------

BYTE = [\x00-\xff]              ; byte data definition
CHAR = [\x00-\x2b\x2d-\xff]     ; char data definition
ChunkSize = (BYTE BYTE BYTE BYTE)

;----- File Chunk (section 4.1) ----------------------------------------------------

Smaf_Chunk = "MMMD" ChunkSize  Contents_Info_Chunk Optional_Data_Chunk?
                (Score_Track_Chunk? | PCM_Audio_Track_Chunk? |
                Graphics_Track_Chunk?)+

;----- Contents Info Chunk (section 4.2) -------------------------------------------

Contents_Info_Chunk = "CNTI" ChunkSize
                Contents_Class Contents_Type
                Contents_Code_Type Copy_Status Copy_Counts
                ((ContentsTag ":" CHAR) ("," ContentsTag ":" CHAR )*)?
  Contents_Class          = BYTE
  Contents_Type           = BYTE
  Contents_Code_Type      = BYTE
  Copy_Status             = BYTE
  Copy_Counts             = BYTE
  CotentsTag = ("VN"|"CN"|"CA"|"ST"|"AN"|"WW"|"SW"|"AW"|"GR"|"MI"|"CR"|"CD" |"UD")

;----- Optional Data Chunk (section 4.3) -------------------------------------------

Optional_Data_Chunk = "OPDA" ChunkSize Data_Chunk*

  Data_Chunk = "Dch" [\x00-\xff] ChunkSize (Contents_Tag Data_Size Data)*
  CotentsTag = BYTE BYTE
  Data_Size = BYTE BYTE
  Data = BYTE*

;----- Score Track Chunk (section 4.4) ---------------------------------------------

Score_Track_Chunk = "MTR" [\x00-\xff] ChunkSize
          Format_Type Sequence_Type Timebase_D
          Timebase_G Channel_Status
         (Seek&Phrase_Info_Chunk? Setup_Data_Chunk? Sequence_Data_Chunk |
          Seek&Phrase_Info_Chunk? Sequence_Data_Chunk Setup_Data_Chunk? |
          Setup_Data_Chunk? Seek&Phrase_Info_Chunk? Sequence_Data_Chunk |
          Setup_Data_Chunk? Sequence_Data_Chunk Seek&Phrase_Info_Chunk? |
          Sequence_Data_Chunk Setup_Data_Chunk? Seek&Phrase_Info_Chunk? |
          Sequence_Data_Chunk Seek&Phrase_Info_Chunk? Setup_Data_Chunk? )
   Format_Type  = BYTE

    ;   for Handy Phone Standard Format (Format_Type=0x00)

  Sequence_Type     = BYTE
  Timebase_D        = BYTE
```

```
  Timebase_G          = BYTE
  Channel_Status = BYTE BYTE
  Seek&Phrase_Info_Chunk = "MsqI" ChunkSize
      ( (StartPoint4 "," StopPoint4 (PhraseList)?)
       |((StartPoint1 "," StopPoint1 PhraseList SubsequenceList)? );
  StartPoint1 = "st:" BYTE;
  StopPoint1  = "sp:" BYTE;
  StartPoint4 = "st:" BYTE BYTE BYTE BYTE;
  StopPoint4  = "sp:" BYTE BYTE BYTE BYTE;
  PhraseList = ("
              ("Pa"|"Pb"|"Pc"|"Pd"|"Pe"|"Pf"|"Pg"|
               "Ph"|"Pi"|"Pj"|"Pk"|"Pl"|"Pm"|"Pn"|
               "Po"|"Pp"|"Pq"|"Pr"|"Ps"|"Pt"|"Pu"|
               "Pv"|"Pw"|"Px"|"Py"|"Pz"|
               "PA"|"PB"|"PE"|"PI"|"PK"|"PS"|"PR")
              ":" BYTE BYTE BYTE BYTE BYTE BYTE BYTE BYTE))*;
  SubsequenceList = ("," "SL:" [a-zABEIKSR]+ );
  Setup_Data_Chunk = "Mtsu" ChunkSize BYTE+
   Sequence_Data_Chunk = "Mtsq" ChunkSize
               ((Duration Event)|(\x00 \x00 \x00 \x00))+
  Duration = BYTE BYTE?
  Event          = BYTE BYTE*

;    for  Mobile Standard Format (Format_Type=0x01,0x02)

   Sequence_Type     = BYTE
  Timebase_D          = BYTE
  Timebase_G          = BYTE
   Channel_Status = BYTE BYTE BYTE BYTE BYTE BYTE BYTE BYTE
               BYTE BYTE BYTE BYTE BYTE BYTE BYTE BYTE
  Seek&Phrase_Info_Chunk = "MsqI" ChunkSize
      ( (StartPoint4 "," StopPoint4 (PhraseList)?)
       |((StartPoint1 "," StopPoint1 PhraseList SubsequenceList)? );
  StartPoint1 = "st:" BYTE;
  StopPoint1  = "sp:" BYTE;
  StartPoint4 = "st:" BYTE BYTE BYTE BYTE;
  StopPoint4  = "sp:" BYTE BYTE BYTE BYTE;
  PhraseList = ("
              ("Pa"|"Pb"|"Pc"|"Pd"|"Pe"|"Pf"|"Pg"|
               "Ph"|"Pi"|"Pj"|"Pk"|"Pl"|"Pm"|"Pn"|
               "Po"|"Pp"|"Pq"|"Pr"|"Ps"|"Pt"|"Pu"|
               "Pv"|"Pw"|"Px"|"Py"|"Pz"|
               "PA"|"PB"|"PE"|"PI"|"PK"|"PS"|"PR")
              ":" BYTE BYTE BYTE BYTE BYTE BYTE BYTE BYTE))*;
  SubsequenceList = ("," "SL:" [a-zABEIKSR]+ );
  Setup_Data_Chunk = "Mtsu" ChunkSize BYTE+
   Sequence_Data_Chunk = "Mtsq" ChunkSize
               ((Duration Event)|(\x00 \x00 \x00 \x00))+
  Duration = BYTE BYTE?
  Event          = BYTE BYTE*
   Stream_PCM_Data_Chunk = "Mtsp" ChunkSize Stream_Wave_Data_Chunk+
   Stream_Wave_Data_Chunk = "Mwa" [\x00-\xff] ChunkSize BYTE+

;----- PCM Audio Track Chunk(section 4.5) ----------------------------------------

  PCM_Audio_Track_Chunk = "ATR" [\x00-\xff] ChunkSize
            Format_Type Sequence_Type Wave_Type
            Timebase_D Timebase_G
     (Seek&Phrase_Info_Chunk? Setup_Data_Chunk? Sequence_Data_Chunk Wave_Data_Chunk+|
```

```
    Seek&Phrase_Info_Chunk? Setup_Data_Chunk? Wave_Data_Chunk+ Sequence_Data_Chunk|
    Seek&Phrase_Info_Chunk? Sequence_Data_Chunk Setup_Data_Chunk? Wave_Data_Chunk+|
    Seek&Phrase_Info_Chunk? Sequence_Data_Chunk Wave_Data_Chunk+ Setup_Data_Chunk?|
    Seek&Phrase_Info_Chunk? Wave_Data_Chunk+ Sequence_Data_Chunk Setup_Data_Chunk?|
    Seek&Phrase_Info_Chunk? Wave_Data_Chunk+ Setup_Data_Chunk? Sequence_Data_Chunk|
    Setup_Data_Chunk? Seek&Phrase_Info_Chunk? Sequence_Data_Chunk Wave_Data_Chunk+|
    Setup_Data_Chunk? Seek&Phrase_Info_Chunk? Wave_Data_Chunk+ Sequence_Data_Chunk|
    Setup_Data_Chunk? Sequence_Data_Chunk Seek&Phrase_Info_Chunk? Wave_Data_Chunk+|
    Setup_Data_Chunk? Sequence_Data_Chunk Wave_Data_Chunk+ Seek&Phrase_Info_Chunk?|
    Setup_Data_Chunk? Wave_Data_Chunk+ Seek&Phrase_Info_Chunk? Sequence_Data_Chunk|
    Setup_Data_Chunk? Wave_Data_Chunk+ Sequence_Data_Chunk Seek&Phrase_Info_Chunk?|
    Sequence_Data_Chunk Setup_Data_Chunk? Seek&Phrase_Info_Chunk? Wave_Data_Chunk+|
    Sequence_Data_Chunk Setup_Data_Chunk? Wave_Data_Chunk+ Seek&Phrase_Info_Chunk?|
    Sequence_Data_Chunk Seek&Phrase_Info_Chunk? Setup_Data_Chunk? Wave_Data_Chunk+|
    Sequence_Data_Chunk Seek&Phrase_Info_Chunk? Wave_Data_Chunk+ Setup_Data_Chunk?|
    Sequence_Data_Chunk Wave_Data_Chunk+ Setup_Data_Chunk? Seek&Phrase_Info_Chunk?|
    Sequence_Data_Chunk Wave_Data_Chunk+ Seek&Phrase_Info_Chunk? Setup_Data_Chunk?)
  Format_Type    = BYTE
  Sequence_Type      = BYTE
  Wave_Type      = BYTE
  Timebase_D         = BYTE
   Timebase_G        = BYTE

;   for Handy Phone Standard Format (section 4.5.1)

  Seek&Phrase_Info_Chunk = "AsqI" ChunkSize
      ( (StartPoint4 "," StopPoint4 (PhraseList)?)
       |((StartPoint1 "," StopPoint1 PhraseList SubsequenceList)? );
  StartPoint1 = "st:" BYTE;
  StopPoint1  = "sp:" BYTE;
  StartPoint4 = "st:" BYTE BYTE BYTE BYTE;
  StopPoint4  = "sp:" BYTE BYTE BYTE BYTE;
  PhraseList       = ("," 
             ("Pa"|"Pb"|"Pc"|"Pd"|"Pe"|"Pf"|"Pg"|
              "Ph"|"Pi"|"Pj"|"Pk"|"Pl"|"Pm"|"Pn"|
              "Po"|"Pp"|"Pq"|"Pr"|"Ps"|"Pt"|"Pu"|
              "Pv"|"Pw"|"Px"|"Py"|"Pz"|
              "PA"|"PB"|"PE"|"PI"|"PK"|"PS"|"PR")
             ":" BYTE BYTE BYTE BYTE BYTE BYTE BYTE BYTE))*;
  SubsequenceList    = ("," "SL:" [a-zABEIKSR]+ );
  Setup_Data_Chunk = "Atsu" ChunkSize BYTE+
   Sequence_Data_Chunk = "Atsq" ChunkSize
               ((Duration Event) | (\x00 \x00 \x00 \x00))+
  Duration = BYTE BYTE?
  Event          = BYTE BYTE*
  Wave_Data_Chunk = "Awa" [\x00-\xff] ChunkSize BYTE+

= BYTE

;----- Graphic Track Chunk (section 6.1) ----------------------------------------------

Graphics_Track_Chunk = "GTR" [\x00-\xff] ChunkSize
        Format_Type Player_Type Text_Encode_Type Color_Type
        Timebase Option_Size Option_Data
            (Setup_Data_Chunk Sequence_Data_Chunk+ Font_Data_Chunk? Image_Data_Chunk? |
              Setup_Data_Chunk Sequence_Data_Chunk+ Image_Data_Chunk? Font_Data_Chunk? |
              Setup_Data_Chunk Image_Data_Chunk? Sequence_Data_Chunk+ Font_Data_Chunk? |
              Setup_Data_Chunk Image_Data_Chunk? Font_Data_Chunk? Sequence_Data_Chunk+ |
              Setup_Data_Chunk Font_Data_Chunk? Sequence_Data_Chunk+ Image_Data_Chunk? |
```

```
        Setup_Data_Chunk Font_Data_Chunk? Image_Data_Chunk? Sequence_Data_Chunk+ |
        Sequence_Data_Chunk+ Setup_Data_Chunk Font_Data_Chunk? Image_Data_Chunk? |
        Sequence_Data_Chunk+ Setup_Data_Chunk Image_Data_Chunk? Font_Data_Chunk? |
        Sequence_Data_Chunk+ Image_Data_Chunk? Setup_Data_Chunk Font_Data_Chunk? |
        Sequence_Data_Chunk+ Image_Data_Chunk? Font_Data_Chunk? Setup_Data_Chunk |
        Sequence_Data_Chunk+ Font_Data_Chunk? Setup_Data_Chunk Image_Data_Chunk? |
        Sequence_Data_Chunk+ Font_Data_Chunk? Image_Data_Chunk? Setup_Data_Chunk |
        Font_Data_Chunk? Setup_Data_Chunk Sequence_Data_Chunk+ Image_Data_Chunk? |
        Font_Data_Chunk? Setup_Data_Chunk Image_Data_Chunk? Sequence_Data_Chunk+ |
        Font_Data_Chunk? Sequence_Data_Chunk+ Setup_Data_Chunk Image_Data_Chunk? |
        Font_Data_Chunk? Sequence_Data_Chunk+ Image_Data_Chunk? Setup_Data_Chunk |
        Font_Data_Chunk? Image_Data_Chunk? Setup_Data_Chunk Sequence_Data_Chunk+ |
        Font_Data_Chunk? Image_Data_Chunk? Sequence_Data_Chunk+ Setup_Data_Chunk |
        Image_Data_Chunk? Setup_Data_Chunk Sequence_Data_Chunk+ Font_Data_Chunk? |
        Image_Data_Chunk? Setup_Data_Chunk Font_Data_Chunk? Sequence_Data_Chunk+ |
        Image_Data_Chunk? Sequence_Data_Chunk+ Setup_Data_Chunk Font_Data_Chunk? |
        Image_Data_Chunk? Sequence_Data_Chunk+ Font_Data_Chunk? Setup_Data_Chunk |
        Image_Data_Chunk? Font_Data_Chunk? Setup_Data_Chunk Sequence_Data_Chunk+ |
        Image_Data_Chunk? Font_Data_Chunk? Sequence_Data_Chunk+ Setup_Data_Chunk |)

Format_Type          = BYTE
Player_Type          = BYTE
Text_Encode_Type = BYTE
Color_Type           = BYTE
 Timebase            = BYTE
 Option_Size         = BYTE
 Option_Data         = BYTE*      (Option_Size byte)

 Setup_Data_Chunk   = "Gtsu" ChunkSize
                (Display_Prameter_Definition_Chunk  Color_Palette_Definition_Chunk? |
                 Color_Palette_Definition_Chunk?  Display_Prameter_Definition_Chunk)
 Diplay_Parameter_Definition_Chunk = “Gdpd” ChunkSize (prmID Value)+
  prmID        = BYTE
  Value      = BYTE

 Color_Palette_Definition_Chunk =”Gcpd” ChunkSize BYTE+

 Sequence_Data_Chunk = "Gsq" [\x00-\xff] ChunkSize
          (Duration Event)*  Duration?  (\x00 \x00 \x00 \x00)+
 Duration = BYTE BYTE?
 Event       = BYTE BYTE*
 Event       = Short_Control_Event | Control_Event | Display_Object_Event

      Short_Control_Event    = EventType

      Control_Event          = EventType EventSize EventData

      DisplayObjectEvent     = EventType EventSize LifeTime Coordinates ObjectSubblock+

                    EventType    = BYTE

                    EventSize    = BYTE BYTE?

                    EventData    = BYTE+   (EventSize byte)

         Coordinates  = Format?  Position  Position

         ObjectSubblock   = SubblockType SubblockSize SubblockBody

            SubblockType    = BYTE

            SubblockSize    = BYTE BYTE?

            SubblockBody    = BYTE+      (SubblockSize byte)
```

```
Font_Data_Chunk = “Gftd” ChunkSize (Font_Chunk* | Unicode_Font_Chunk*)
 Font_Chunk =”Ge” [\x00-\xff = FontType][\x00-\xff = FontSize] ChunkSize
                    EntryNumber FontData*
            EntryNumber    = BYTE BYTE?
            FontData       = BYTE+
            FontData       = Code Size Bitmap
        Code           = BYTE BYTE
                Size           = BYTE BYTE?
                Bitmap         = BYTE+
                Bitmap         = TYPE XYwidth BitmapData
                        TYPE           = BYTE
                        XYwidth        = BYTE BYTE BYTE  (12bit x 2 = 3 byte expression)
                        BitmapData     = (refer to § 6.5.2)
 Font_Chunk =”Gu” [\x00-\xff = FontType][\x00-\xff = FontSize] ChunkSize
                    EntryNumber FontData*
            EntryNumber    = BYTE BYTE?
            FontData       = BYTE+
            FontData       = Size Code Size Bitmap
                Size           = BYTE BYTE?
                Code           = BYTE+
                Size           = BYTE BYTE?
                Bitmap         = BYTE+
                Bitmap         = TYPE XYwidth BitmapData
                        TYPE           = BYTE
                        XYwidth        = BYTE BYTE BYTE  (12bit x 2 = 3 byte expression)
                        BitmapData     = (refer to § 6.5.2)

Image_Data_Chunk =”Gimd” ChunkSize Image_Chunk* Bmp_Chunk* Link_Chunk*
   Image_Chunk = ”Gig” [\x00-\xff] ChunkSize ImageData*
      ImageData = BYTE*
   Bmp_Chunk = "Gbm"[\x00-\xff] ChunkSize Bitmap*
      Bitmap = TYPE SIZE BitmapData
      TYPE = BYTE
      SIZE = BYTE BYTE BYTE
Link_Chunk = “Gln* [\x00-\xff] ChunkSize LinkData*
      LinkData = BYTE*

;-------------------------------------------------------------------------------------------------------------------------
```

9.4. Standard voice map

The following standard voice map is defined in order to maintain playback compatibility in the Score Track Chunk.

9.4.1. Handy phone standard

PC#	Name Bank 0x00	0x80	PC#	Name Bank 0x00	0x80	PC#	Name Bank 0x00	0x80	PC#	Name Bank 0x00	0x80
0	GrandPno		32	AcoBass	Sticks	64	SprnoSax	Conga L	96	Rain	
1	BritePno		33	FngrBass	Bass Drum L	65	AltoSax	Timbale H	97	SoundTrk	
2	E.GrandP		34	PickBass	Open Rim Shot	66	TenorSax	Timbale L	98	Crystal	
3	HnkyTonk		35	Fretless	Bass Drum M	67	Bari.Sax	Agogo H	99	Atmosphr	
4	E.Piano1		36	SlapBas1	Bass Drum H	68	Oboe	Agogo L	100	Bright	
5	E.Piano2		37	SlapBas2	Closed Rim Shot	69	Eng.Horn	Cabasa	101	Goblins	
6	Harpsi		38	SynBass1	Snare M	70	Bassoon	Maracas	102	Echoes	
7	Clavi		39	SynBass2	Hand Clap	71	Clarinet	Samba Whistle H	103	Sci-Fi	
8	Celesta		40	Violin	Snare H	72	Piccolo	Samba Whistle L	104	Sitar	
9	Glocken		41	Viola	Floor Tom L	73	Flute	Guiro Short	105	Banjo	
10	MusicBox		42	Cello	Hi-Hat Closed	74	Recorder	Guiro Long	106	Shamisen	
11	Vibes		43	Contrabs	Floor Tom H	75	PanFlute	Claves	107	Koto	
12	Marimba		44	TremStr	Hi-Hat Pedal	76	Bottle	Wood Block H	108	Kalimba	
13	Xylophon		45	PizzStr	Low Tom	77	Shakhchi	Wood Block L	109	Bagpipe	
14	TubulBel		46	Harp	Hi-Hat Open	78	Whistle	Cuica Mute	110	Fiddle	
15	Dulcimar		47	Timpani	Mid Tom L	79	Ocarina	Cuica Open	111	Shanai	
16	DrawOrgn		48	Strings1	Mid Tom H	80	SquareLd	Triangle Mute	112	TnklBell	
17	PercOrgn		49	Strings2	Crash Cymbal 1	81	SawLead	Triangle Open	113	Agogo	
18	RockOrgn		50	Syn.Str1	High Tom	82	CaliopLd	Shaker	114	SteelDrm	
19	ChrchOrg		51	Syn.Str2	Ride Cymbal 1	83	ChiffLd	Jingle Bell	115	WoodBlk	
20	ReedOrgn		52	ChoirAah	Chinese Cymbal	84	CharanLd	Belltree	116	TaikoDrm	
21	Acordion		53	VoiceOoh	Ride Cymbal Cup	85	VoiceLd		117	MelodTom	
22	Harmnica		54	SynVoice	Tamboulin	86	FifthLd		118	Syn.Drum	
23	TangoAcd		55	Orch.Hit	Splash Cymbal	87	Bass&Ld		119	RevCymbl	
24	NylonGtr	SeqClick H	56	Trumpet	Cowbell	88	NewAgePd		120	FretNoiz	
25	SteelGtr	Brush Tap	57	Trombone	Crash Cymbal 2	89	WarmPad		121	BrthNoiz	
26	JazzGtr	Brush Swirl L	58	Tuba	Vibraslap	90	PolySyPd		122	SeaShore	
27	CleanGtr	Brush Slap	59	Mute.Trp	Ride Cymbal 2	91	ChoirPad		123	Tweet	
28	Mute.G.tr	Brush Swirl H	60	Fr.Horn	Bongo H	92	BowedPad		124	Telphone	
29	Ovrdrive	Snare Roll	61	BrasSect	Bongo L	93	MetalPad		125	Helicptr	
30	Dist.Gtr	Castanet	62	SynBras1	Conga H Mute	94	HaloPad		126	Applause	
31	GtrHarmo	Snare L	63	SynBras2	Conga H Open	95	SweepPad		127	Gunshot	

9.4.2. Mobile standard

normal voice

PC#	Name
0	GrandPno
1	BritePno
2	E.GrandP
3	HnkyTonk
4	E.Piano1
5	E.Piano2
6	Harpsi
7	Clavi
8	Celesta
9	Glocken
10	MusicBox
11	Vibes
12	Marimba
13	Xylophon
14	TubulBel
15	Dulcimar
16	DrawOrgn
17	PercOrgn
18	RockOrgn
19	ChrchOrg
20	ReedOrgn
21	Acordion
22	Harmnica
23	TangoAcd
24	NylonGtr
25	SteelGtr
26	JazzGtr
27	CleanGtr
28	Mute.G.tr
29	Ovrdrive
30	Dist.Gtr
31	GtrHarmo

PC#	Name
32	AcoBass
33	FngrBass
34	PickBass
35	Fretless
36	SlapBas1
37	SlapBas2
38	SynBass1
39	SynBass2
40	Violin
41	Viola
42	Cello
43	Contrabs
44	TremStr
45	PizzStr
46	Harp
47	Timpani
48	Strings1
49	Strings2
50	Syn.Str1
51	Syn.Str2
52	ChoirAah
53	VoiceOoh
54	SynVoice
55	Orch.Hit
56	Trumpet
57	Trombone
58	Tuba
59	Mute.Trp
60	Fr.Horn
61	BrasSect
62	SynBras1
63	SynBras2

PC#	Name
64	SprnoSax
65	AltoSax
66	TenorSax
67	Bari.Sax
68	Oboe
69	Eng.Horn
70	Bassoon
71	Clarinet
72	Piccolo
73	Flute
74	Recorder
75	PanFlute
76	Bottle
77	Shakhchi
78	Whistle
79	Ocarina
80	SquareLd
81	SawLead
82	CaliopLd
83	ChiffLd
84	CharanLd
85	VoiceLd
86	FifthLd
87	Bass&Ld
88	NewAgePd
89	WarmPad
90	PolySyPd
91	ChoirPad
92	BowedPad
93	MetalPad
94	HaloPad
95	SweepPad

PC#	Name
96	Rain
97	SoundTrk
98	Crystal
99	Atmosphr
100	Bright
101	Goblins
102	Echoes
103	Sci-Fi
104	Sitar
105	Banjo
106	Shamisen
107	Koto
108	Kalimba
109	Bagpipe
110	Fiddle
111	Shanai
112	TnklBell
113	Agogo
114	SteelDrm
115	WoodBlk
116	TaikoDrm
117	MelodTom
118	Syn.Drum
119	RevCymbl
120	FretNoiz
121	BrthNoiz
122	SeaShore
123	Tweet
124	Telphone
125	Helicptr
126	Applause
127	Gunshot

drum voice

note#	Name
32	
33	
34	
35	Bass Drum M
36	Bass Drum H
37	Closed Rim Shot
38	Snare M
39	Hand Clap
40	Snare H
41	Floor Tom L
42	Hi-Hat Closed
43	Floor Tom H
44	Hi-Hat Pedal
45	Low Tom
46	Hi-Hat Open
47	Mid Tom L
48	Mid Tom H
49	Crash Cymbal
50	High Tom
51	Ride Cymbal 1
52	Chinese Cymbal
53	RideCymbal Cup
54	Tamboulin
55	Splash Cymbal
56	Cowbell
57	Crash Cymbal 2
58	Vibraslap
59	Ride Cymbal 2
60	Bongo H
61	Bongo L
62	Conga H Mute
63	Conga H Open

note#	Name
64	Conga L
65	Timbale H
66	Timbale L
67	Agogo H
68	Agogo L
69	Cabasa
70	Maracas
71	Samba Whistle H
72	Samba Whistle L
73	Guiro Short
74	Guiro Long
75	Claves
76	Wood Block H
77	Wood Block L
78	Cuica Mute
79	Cuica Open
80	Triangle Mute
81	Triangle Open
82	
83	
84	
85	
86	
87	
88	
89	
90	
91	
92	
93	
94	
95	