SMAF/Phrase L1 Specification

Ver. 1.5.3

02/22/2002

Yamaha Corporation

Contents

1 OUTLINE 4

2 FUNDAMENTAL IDEA OF SMAF/PHRASE 4

3 REPLAY ENVIRONMENT MODEL OF SMAF/PHRASE 5

4 SPECIFICATION OF SMAF/PHRASE 6

4.1 MINIMUM SMAF FILE CONFIGURATION AS SMAF/PHRASE 6
4.1.1 Copy management 7
4.1.2 Character codes that are used for management of music information 7
4.2 PHASE CHUNK STRUCTURE 8
4.2.1 Phrase chunk 9
4.2.2 Contents Info chunk 9
4.2.3 Voice chunk 9
4.2.4 Sequence chunk 9
4.3 SMAF/PHRASE TONE GENERATION MODEL 10
4.4 SMAF/PHRASE VOICES 10
4.5 SMAF/PHRASE SEQUENCE MESSAGE 11
4.5.1 Essential messages 11
4.5.2 Optional messages 14
4.5.3 Messages that are not supported 16
4.6 TREATMENT OF DEFAULT VALUES OF SMAF/PHRASE SEQUENCE MESSAGE 16

5 APPENDIX 17

5.1 DEFAULT SOUND SET 17
5.2 SMAF/PHRASE MESSAGE MAP 19
5.3 GENERATION OF CRC 21
5.4 DIFFERENCE FROM SMAF/MA2 SEQUENCE DATA 23

Notes:

The description of definition of structures uses the following types of definition.

Definition type	Meaning	Definition type	Meaning
UINT8	8 bits without code	SINT8	8 bits with code
UINT16	16 bits without code	SINT16	16 bits with code
UINT32	32 bits without code	SINT32	32 bits with code

< History >

Ver.	Date	Description
1.0	12/28/2000	First edition
1.1	1/9/2001	Default voice chunk size was corrected. TimeBase value was corrected.
1.2	1/19/2001	Omission of ContentsType that is specified as essential by SMAF specification was corrected.
1.3	2/14/2001	UserEvent was extended. SMAF/Phrase ContentsType = 0xF0 was defined.
1.31	3/5/2001	Items that are different between SMAF/Phrase and SMAF/MA2 were added.
1.32	3/12/2001	Change of classification of reserved messages.
1.4	4/4/2001	Change of tone generation method when no voices are designated Addition of default of ProgramChange/OctaveShift/ Modulation Addition of description about default value setting
1.41	4/23/2001	Correction of wrong expression
1.5	6/4/2001	Addition of description about Contents Info Chunk Correction of wrong voice name Change of type definition
1.5.1	9/10/2001	Addition of description about Optional Data Chunk
1.5.2	1/28/2002	Addition of Gatetime=0 prohibition of NoteMessage. Clerical error correction.
1.5.3	2/22/2002	Clerical error correction.

1 Outline

SMAF/Phrase, a format derived from SMAF standard, that is assumed to be used by applications such as Java is defined here. This has been designed to achieve a performance that can reproduce four SMAF/Phrase sequences by using synthesizer hardware compatible with standard SMAF/MA2 (SMAF sub-standard that mainly covers 16 chord ringing melody).
The definition here is described as "SMAF/Phrase L1" meaning the first installation of SMAF/Phrase.

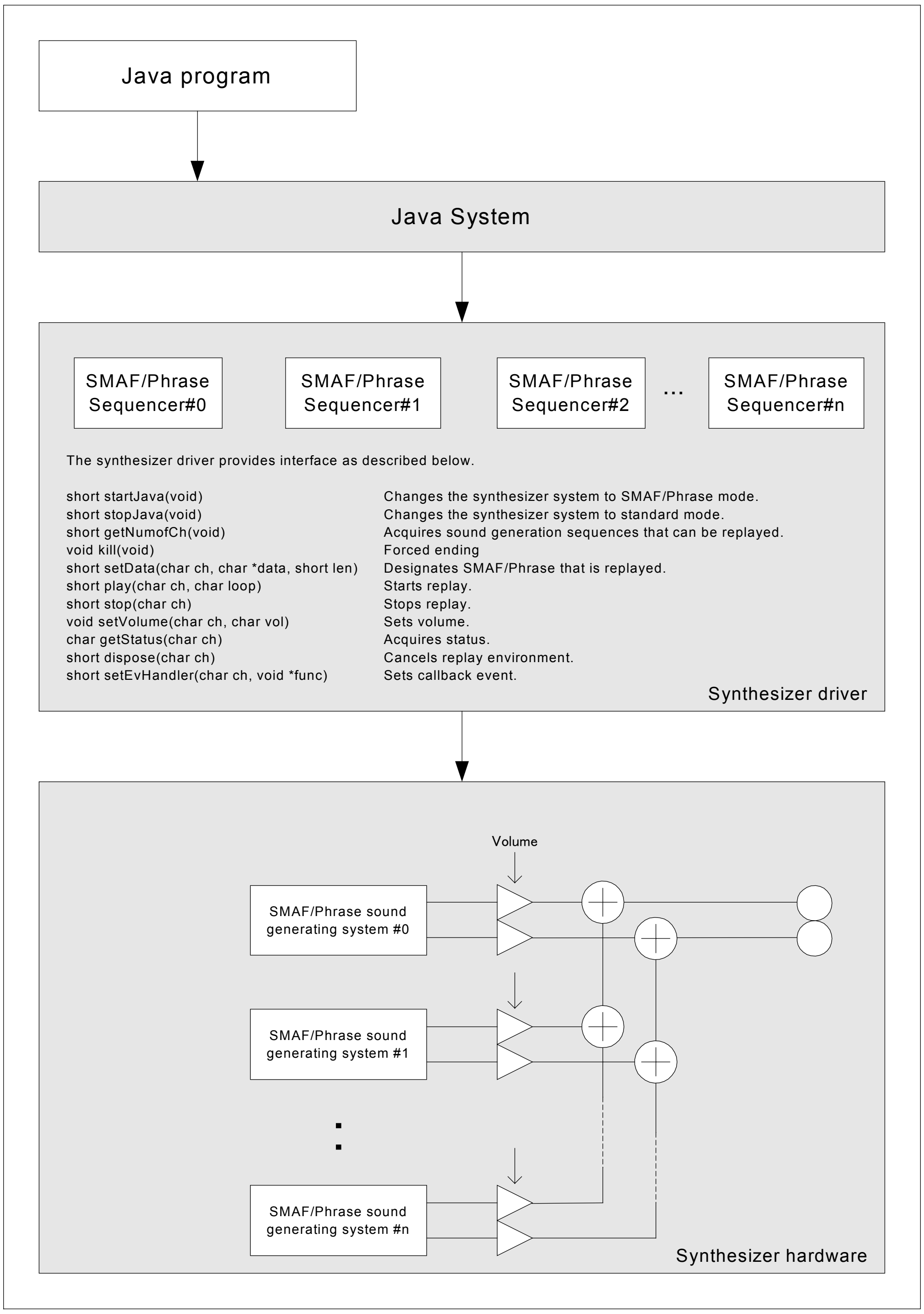
2 Fundamental Idea of SMAF/PHRASE

SMAF/Phrase has been designed based on SMAF/MA2, and thus treats sequential events that conform to SMAF specification, but does not always support all of them. Taking into consideration that it can perform reproduction of four SMAF/Phrase sequences on the hardware compatible with SMAF/MA2 standard, it is given features as described below.

· Sequential format that uses fixed time unit.
· Uses sequential events that are described in SMAF specification (however, supports only some of the events).
· Interprets mono-phonic tones sequentially.
· Allows sequence of up to 4 channels (4 tone simultaneous generation).
· Allows definition of up to 4 voices.
· Allows change of voices freely in a sequence.
· Has no idea of drum bank that is defined in MIDI. Melody bank only.
· Definition of standard default voices
· Definition of user events

3 Replay Environment Model of SMAF/Phrase

An example of replay environment is shown below.



The synthesizer driver provides interface as described below.

short startJava(void)	Changes the synthesizer system to SMAF/Phrase mode.
short stopJava(void)	Changes the synthesizer system to standard mode.
short getNumofCh(void)	Acquires sound generation sequences that can be replayed.
void kill(void)	Forced ending
short setData(char ch, char *data, short len)	Designates SMAF/Phrase that is replayed.
short play(char ch, char loop)	Starts replay.
short stop(char ch)	Stops replay.
void setVolume(char ch, char vol)	Sets volume.
char getStatus(char ch)	Acquires status.
short dispose(char ch)	Cancels replay environment.
short setEvHandler(char ch, void *func)	Sets callback event.

4 Specification of SMAF/Phrase

4.1 Minimum SMAF file configuration as SMAF/PHRASE

Since SMAF/Phrase is a sub-standard of SMAF, it has an SMAF structure as a whole. An example of minimum structure is shown below. It requires at least two chunks, Contents Info chunk that is essential in SMAF standard, and Phrase chunk that contains the definition of SMAF/ Phrase. Operation of Contents Info chunk is to conform to SMAF standard. Since SMAF/ Phrase belongs to SMAF standard, it is necessary to install Contents Info chunk in accordance with SMAF operation specification of SMAF specification to which the applicable unit belongs. Only one Phrase chunk is to exist presently. However, execute processing that does not cause problems even if two or more Phrase chunks are placed. When two or more Phrase chunks are placed, execute the processing for the chunk that is found first as viewed from the head of the file.
Since SMAF standard allows inclusion of chunks other than those shown here, it is necessary to ignore unknown chunks when they are found. Since a chunk is to have a 4 byte identifier and 4 byte size, it can be omitted easily when reading them. The alignment is made by byte.
When corresponds to multi-national and adds music information, we recommend to use Optional Data Chunk added from SMAF Ver 3.00 rather than it sets of Option of Contents Info Chunk.

```
<Main chunk>
UINT8          ID[4] = “MMMD”
UINT32         Size                      // whole size [bytes] (BigEndian)

     <Contents Info chunk>              // To conform to the specification of category to
                                           which the units belong.
       UINT8          ID[4] = “CNTI”
       UINT32         Size               // size of chunk [bytes]
       UINT8          Class = 0          // data class (0: YAMAHA)
       UINT8          Type = 0xF0        // data type(0xF0: SMAF/Phrase data)
       UINT8          CodeType           // codes of characters that are used as option
       UINT8          CopyStatus         // copy definition (0: free, 1: Copy not allowed)
       UINT8          CopyCount          // copy count(0..255)
       UINT8          Option[]           // Storing music information(option)

       <Phrase chunk>
       UINT8          ID[4] = “MMMG”
       UINT32         Size               // size of data    [bytes](BigEndian)
         :

UINT16  CRC                              //CRC code for checking legality    (BigEndian)
<End>
```

4.1.1 Copy management

Contents Info Chunk has CopyStatus and CopyCount that show possibility of copying.

4.1.1.1 CopyStatus

Contains information that copying can be made, transfer can be made and storage can be made. The definition is as described below when the bits are numbered as 0, 1, 2, .., 7 from LSB side.

bit 0 :	transfer (0: allowed, 1: not allowed)
bit 1 :	save (0: allowed, 1: not allowed)
bit 2 :	edit (0: allowed, 1: not allowed)
bit 3..7 :	reserved (fixed to "0")

4.1.1.2 CopyCount

This is a counter for managing the number of times of copying. Sends out data of CopyCount incremented by one every time a copying (transfer) is performed. However, CopyCount is not incremented when it is 255.

4.1.2 Character codes that are used for management of music information

Optionally,. data that shows the information of music can be placed in "Option" of Contents Info Chunk. The character codes to be used here are designated with CodeType of Contents Info Chunk, and it is recommended to use 0x00:S-JIS, 0x01:Latin-1(ISO 8859-1) or 0x23:UTF-8(Unicode).

4.1.2.1 Management of music information

Music information is managed by using tag name (ASCII) in "Option" of Contents Info Chunk. The information is presented in the order of "tag(2byte ASCII)", "0x3A", "data string", and "0x2C". A special procedure is needed when a delimiter (0x2c) is included in the data string. For the special procedure, refer to specification for operation.

Identifier	Meaning
VN	Vender's name
CN	Carrier's name
CA	Category name
ST	Title
AN	Artist's name
WW	Writer
SW	Composer
AW	Arranger
CR	Copyright(c)
CD	Date of preparation
UD	Date and time of update
GR	Name of copyright management party
MI	Information about copyright management party

4.2 Phase chunk structure

The structure of Phrase chunk that is the main body of SMAF/Phrase is shown in this section. Designation of numerical values by using two or more bytes is to conform to the rule of BigEndian.

```
< Phrase chunk>
UINT8           ID[4] = "MMMG"
UINT32          Size                    // size of data    [bytes]
UINT8           Vers                    // format version (fixed to 1)
UINT8           TimeBase                // time unit [ms] (fixed to 20)

      <Contents Info chunk>
      UINT8           ID[4] = "INFO"
      UINT32          Size              // size of data    [bytes]
      UINT8           Info[]            // information character data string
                                          (CString)

      <Voice chunk>
      UINT8           ID[4] = "VOIC"
      UINT32          Size              // size of data    [bytes]

            //Up to four   <DeVoice chunk> and/or <ExVoice chunk> can be placed as follows.
            // Fifth one and after can be ignored.
            // Voice numbers (0 to 3) are assigned starting with 0.

            <DeVoice chunk>             // case of default voices
            UINT8       ID[4] = "DEVO"
            UINT32      Size            // size of data    [bytes](fixed to 1)
            UINT8       Program         // default voice numbers (0..127)

            <ExVoice chunk>             // case of user voices (HW dependent)
            UINT8       ID[4] = "EXVO"
            UINT32      Size            // size of data    [bytes]
            UINT8       VoiceData[]     // sets voices of SystemEx type of SMAF
                                        // 0xff, 0xf0, ..., 0xf7

      <Sequence chunk>
      UINT8           ID[4] = "SEQU"
      UINT32          Size = n          // size of data    [bytes]
      UINT8           Data[n]           // SMAF sequential data string

<End>
```

4.2.1 Phrase chunk

Phrase chunk has "MMMG" at the head as an ID for showing that it is for Phrase. Phrase chunk is able to include one Contents Info chunk, one Voice chunk and one Sequence chunk. The order of the sub-chunks is as designated here. However, any sub-chunk can be omitted. The Vers settings that are present here must be checked, and tones of those that do not correspond are not to be produced.
In this document, only Vers=1 (SMAF/Phrase L1) is defined as the first installation of SMAF/Phrase. When Vers=1, TimeBase is fixed to 20[ms], different values are to be ignored even if they are set.

4.2.2 Contents Info chunk

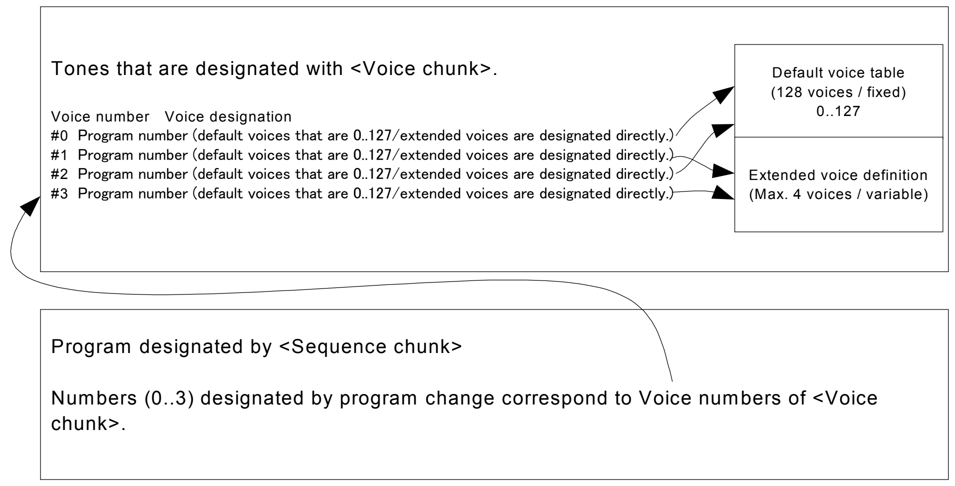
This is an area in which information about data is placed. Any character strings can be stored by using CString format (ending with NULL). Designation of character code system is not defined. This area is used as a memo. It is recommended to place information to be displayed in Contents Info chunk of the SMAF main unit structure side or in Optional Data Chunk. Since the main unit side information is to designate character codes, character strings are determined uniformly.

4.2.3 Voice chunk

Voice chunk sets the relationship between program numbers and voices that are used in voice generation sequence. The setting can be made for up to four voices. When five or more voices are designated, fifth one and after are ignored. The voices are defined with DeVoice sub-chunk for setting default voices and ExVoice sub-chunk for setting extended voices that depend on models. The voice numbers of default voices are shown on the appendix.

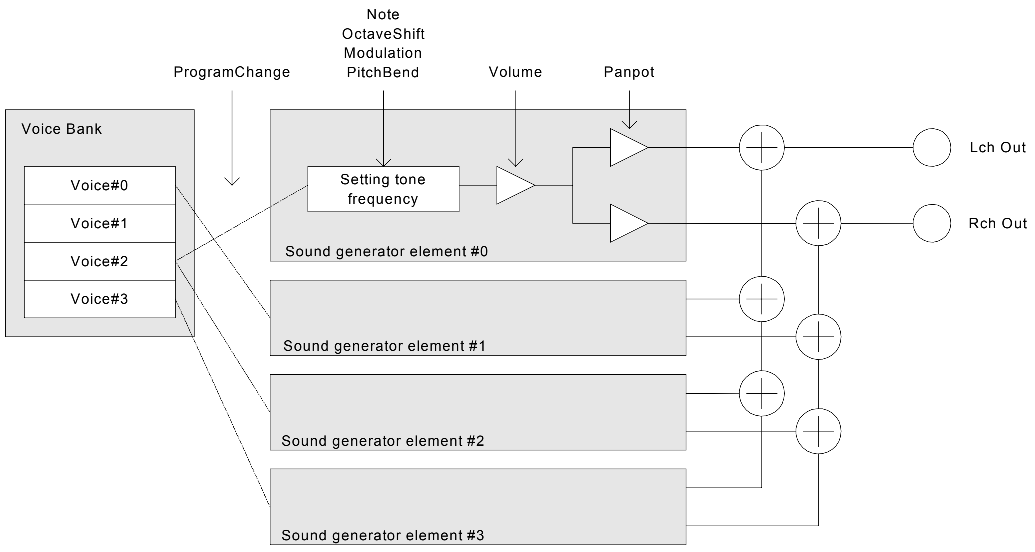
4.2.4 Sequence chunk

Sequence chunk places SMAF sequence. SMAF sequence has a message structure that allows four tone mono-phonic control. The voices that are generated when the voice numbers designated by the program are not defined by Voice chunk are to be default number zero (grand piano) voice.


Tones that are designated with <Voice chunk>.
Voice number Voice designation
#0 Program number (default voices that are 0..127/extended voices are designated directly.)
#1 Program number (default voices that are 0..127/extended voices are designated directly.)
#2 Program number (default voices that are 0..127/extended voices are designated directly.)
#3 Program number (default voices that are 0..127/extended voices are designated directly.)
Default voice table (128 voices / fixed) 0..127
Extended voice definition (Max. 4 voices / variable)
Program designated by <Sequence chunk>
Numbers (0..3) designated by program change correspond to Voice numbers of <Voice chunk>.

4.3 SMAF/Phrase tone generation model

SMAF/Phrase is data for executing four timber of 4 tone generation and mono-phonic tone generation. Maximum volume of each sound generator element is to be 0 dB (Panpot at the center, and Lch/Rch are -3 dB). Therefore, if no distortion is expected under any condition, it is necessary to set the maximum volume of each sound generator element to 12 dB or lower.

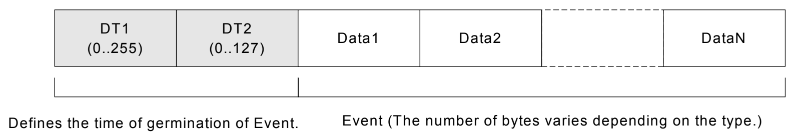


4.4 SMAF/Phrase voices

Default voices are basically the same as GM melody voices. Although they are to be tones of equal temperament, installation in accordance with a rule that is deemed optimum by the person who installs is acceptable for percussive voices and SE voices (Prog#115~127). For generation of extended voices, the installation in accordance with a rule that is deemed optimum by the person who installs is also acceptable. For voices that are not set and extended voices that are not supported, voice of default number 0 (grand piano) is to be produced.

4.5 SMAF/Phrase sequence message

One SMAF data consists of a combination of DT (1 or 2-byte) and Event (>1-byte). DT designates the time of startup of Event. Judgement whether the length of DT is 1 byte or 2 bytes uses MSB of the first byte. When MSB is 0, DT is expressed 0..127 with 1-byte. When MSB is 1, DT is expressed with the value of 14 bits (7 bits of low ranks of the next byte are connected with 7 bits of low ranks of the first byte) and +128.
The unit of DT is fixed to 20 [ms]. If a message that is not included in the essential messages, optional message or messages that are not supported is encountered, an error occurs and the tone generation is stopped.


DT1
(0..255)
DT2
(0..127)
Data1
Data2
DataN
Defines the time of germination of Event.
Event (The number of bytes varies depending on the type.)

DT2 is added only whenMSB of DT1 is "1".
Time = DT1 * TimeBase (MSB of DT1 = 0) TimeBase = 20[ms]
Time = (((DT1 & 0x7f) << 7) + DT2 + 128) * TimeBase (MSB of DT1 = 1)

4.5.1 Essential messages

Units that conform to SMAF/Phrase must interpret the following message.

Volume	0x00, 0x3B+(NN<<6), XX
NN :	#channel (0..3)
XX :	volume setting (0..127) Vol[dB] = 20 * log(XX^2/127^2) default: 100

Shortened type Volume		0x00, 0x00+(NN<<6)+XX
NN :	#channel (0..3)	
XX :	Shortened type format for volume setting (1..14)	
	0x1	Mute
	0x2	20 * log(31^2/127^2)
	0x3	20 * log(39^2/127^2)
	0x4	20 * log(47^2/127^2)
	0x5	20 * log(55^2/127^2)
	0x6	20 * log(63^2/127^2)
	0x7	20 * log(71^2/127^2)
	0x8	20 * log(79^2/127^2)
	0x9	20 * log(87^2/127^2)
	0xA	20 * log(95^2/127^2)
	0xB	20 * log(103^2/127^2)
	0xC	20 * log(111^2/127^2)
	0xD	20 * log(119^2/127^2)
	0xE	0dB

Panpot 00, 0x3A+(NN<<6), XX

NN :	#channel (0..3)
XX :	volume setting (0..127)
	Lch[dB] = 20 * log(Cos(PAI/2 * XX/127) , < Lch = MUTE if vv = 127>
	Rch[dB] = 20 * log(Sin(PAI/2 * XX/127) , < Rch = MUTE if vv = 0>
	default: 64

NOP 0xff, 0x00

Nothing occurs. This is used to extend blank time.

UserEvent 0xff, 0x1n

n :	#Event (0x0..0xf)

Performs callback for the system.

Program change 0x00, 0x30+(NN<<6), XX

NN :	#channel (0..3)
XX :	#voice (0..3)
	default: 0

Selects a voice from four voices that can be set. The relationship between #VoiceID and actual voices are set by the voice setting side. The result of voice generation when program is changed during NoteOn is not guaranteed. It is expected that the program is changed after NoteOff.

Octave shift 0x00, 0x32+(NN<<6), XX

NN :	#channel (0..3)	
XX :	ocrtave change (0..4, 0x81..0x84)	
	0x00	Normal
	0x01	+1oct
	0x02	+2oct
	0x03	+3oct
	0x04	+4oct
	0x81	-1oct
	0x82	-2oct
	0x83	-3oct
	0x84	-4oct
	default: 0	

Changes the range of tone generation key that is designated by Note message. The change becomes valid from the issue of message after this message.

Note message KEY+(OCT<<4)+(NN<<6), G1, <G2>

NN : #channel (0..3)

KEY : Setting hight of key (1..12)

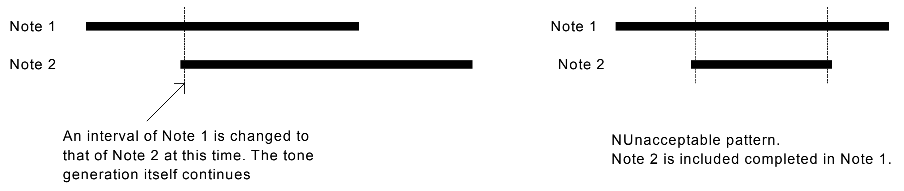
1	C#	8	G#
2	D	9	A
3	D#	10	A#
4	E	11	B
5	F	12	C
6	F#		
7	G		

OCT : octave setting (0..3)

0	0 oct
1	+1 oct
2	+2 oct
3	+3 oct

G1/G2 : GateTime
No G2 if MSB of G1 is "0"
When MSB of G1 is "1", lower 7 bits of G1 and G2 are coupled in upper bits of G1 and then 128 is added to obtain Gate Time. GateTime=0 is forbidden.

Designates NoteOn/NoteOff. Tone generation is executed in accordance with mono-phonic rule. When OCT = 2 / KEY = 9, A4=440Hz. Actual tone generation key is affected also by Octave shift message. However, when the tone generation period overlaps for 1 or more, the tone is judged as continuous tone, and becomes tie / slur tone generation at the time the tone overlaps. The result of tone generation when Note which is included completely in the tone generation time of previous Note is not guaranteed.


Note 1
Note 2
An interval of Note 1 is changed to that of Note 2 at this time. The tone generation itself continues
Note 1
Note 2
NUnacceptable pattern.
Note 2 is included completed in Note 1.

(Reference)

MIDI_Key(0..127) = KEY(1..12) + (OCT(0..3) + OctShift(-4..+4) + 3) * 12

Note: The range of MIDI_Key may be exceeded depending on the setting of KEY/OCT/OctShift.

4.5.2 Optional messages

Messages that do not necessarily have to be supported. When they are not supported, these are treated as NOP.

Modulation 0x00, 0x33+(NN<<6), XX

NN :	channel number (0..3)		
XX :	vibrato setting value (0x00..0x7f)		
	vibrato depth change =	OFF	(XX=0x00)
		1.0	(XX=0x01)
		2^(XX/32)	(XX>0x01)
	default: OFF		

Sets the amount of vibrato. Shows multiplying factor in [cents]. For example, when XX=64, the modulation is 2^2 times, resulting in 4 times. If the depth of vibrato of original voice is 10[cents], the modulation is 40[cents]. Waveform of LFO depends on the person who performs installation.

Shortened type Modulation 0x00, 0x20+(NN<<6)+XX

NN : #channel (0..3)

XX : vibrato setting value (1..14)

0x01	vibrato OFF
0x02	standard setting x 2^(8/32) [cents]
0x03	standard setting x 2^(16/32) [cents]
0x04	standard setting x 2^(24/32) [cents]
0x05	standard setting x 2^(32/32) [cents]
0x06	standard setting x 2^(40/32) [cents]
0x07	standard setting x 2^(48/32) [cents]
0x08	standard setting x 2^(56/32) [cents]
0x09	standard setting x 2^(64/32) [cents]
0x0a	standard setting x 2^(72/32) [cents]
0x0b	standard setting x 2^(80/32) [cents]
0x0c	standard setting x 2^(96/32) [cents]
0x0d	standard setting x 2^(112/32) [cents]
0x0e	standard setting x 2^(127/32) [cents]

PitchBend 0x00, 0x34+(NN<<6), XX

NN :	channel number (0..3)
XX :	Amount of bend (0x00..0x40..0x7f)
	0x00 (-200-cent)..0x40 (0-cent)..0x7f (+200-cent)
	Cent linear change curve
	default: 0x40

Designates the change of replay pitch.

SytemExMsg 0xff, 0xf0, LEN, MakerID, ..., 0xf7

LEN :	Number of bytes from MakerID to 0xf7 (0x03..0xff)
MakerID :	MakerID (0x00..0xff)
...	Command string

Extention message that is used for executing settings depending on the synthesizer hardware. Since MakerID is a value (0x00..0x7f)that conforms to MIDI standard, the use of this message requires the registration as MIDI. 0x80..0xff of MakerID is for preparatory use, and thus, we consider to release it for manufacturers who have not executed MIDI registration yet. (For the details, ask SMAF/Phrase management party for confirmation.)

4.5.3 Messages that are not supported

SMAF/Phrase treats the following messages as NOP.

Reserved	0xff, XX
XX :	(0x00..0xff) However, 0x00/0x10..0x1f/0xf0 are excluded.

Reserved area, presently undefined.

BankSelect	0x00, 0x31+(NN<<6), XX
NN :	channel number (0..3)
XX :	setting value (0x00..0xff)

Shortened type PitchBend	0x00, 0x10+(NN<<6)+XX
NN :	channel number (0..3)
XX :	setting value (0x01..0x0e)

ChVolume	0x00, 0x37+(NN<<6), XX
NN :	channel number (0..3)
XX :	setting value (0x00..0xff)

4.6 Treatment of default values of SMAF/Phrase sequence message

When the data of SMAF/Phrase is newly loaded, it is necessary to guarantee that each element of SMAF/Phrase tone generation mechanism has been set to the default value shown in 4.5. Concretely, the setting is as described below.

Volume	= 100	
Pan	= 64	
ProgramChange	= 0	
OctaveShift	= 0	
Modulation	= 0	(optional message)
PitchBend	= 64	(optional message)

5 Appendix

5.1 Default sound set

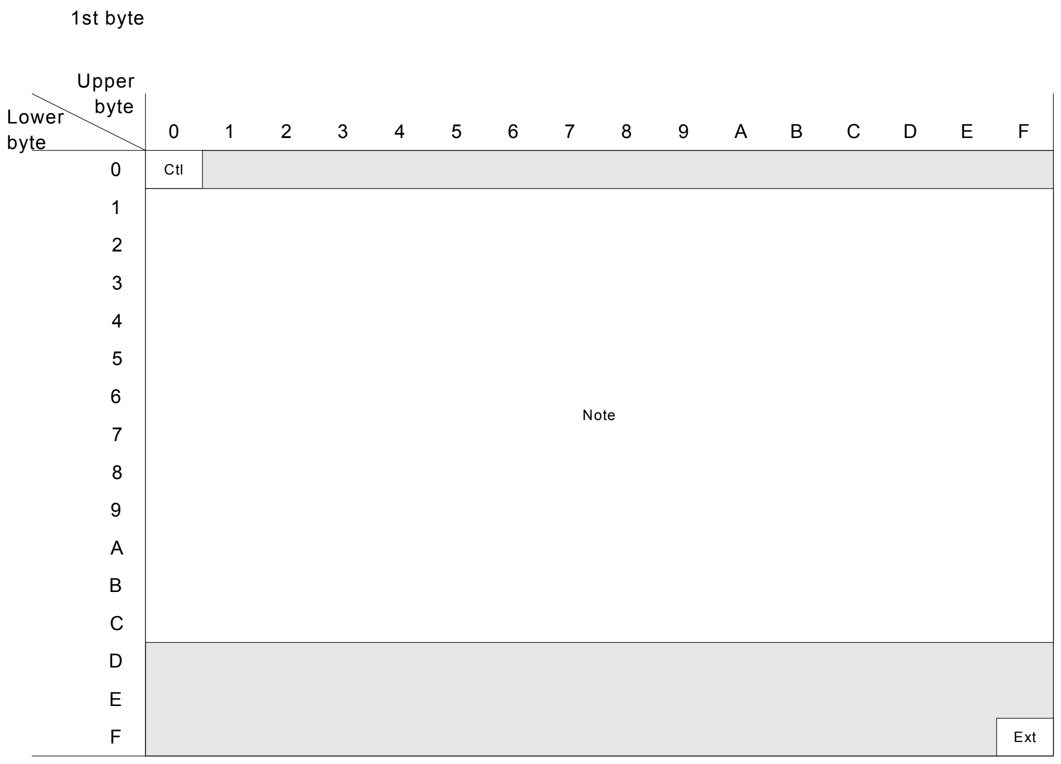
The Key Range is a recommended scale range, and if the range is exceeded, the voices are not guaranteed as designated instrument voices.
Key Range (which can be heard as difference instrument) is a Key number used in MIDI representation, and tone generation is to be an equal temperament. However, installation of the part of Key Range marked with * is to depend on the person who execute the installation.

PC#	Instrument	Key Range	PC#	Instrument	Key Range
0	GrandPno	21-108	32	AcoBass	28-55
1	BritePno	21-108	33	FngrBass	28-55
2	E.GrandP	21-108	34	PickBass	28-55
3	HnkyTonk	21-108	35	Fretless	28-55
4	E.Piano1	28-103	36	SlapBas1	28-55
5	E.Piano2	28-103	37	SlapBas2	28-55
6	Harpsi	41-89	38	SynBass1	28-55
7	Clavi	36-96	39	SynBass2	28-55
8	Celesta	60-108	40	Violin	55-96
9	Glocken	72-108	41	Viola	48-84
10	MusicBox	60-84	42	Cello	36-72
11	Vibes	53-89	43	Contrabs	28-55
12	Marimba	48-84	44	TremStr	28-96
13	Xylophon	65-96	45	PizzStr	28-96
14	TubulBel	60-77	46	Harp	23-103
15	Dulcimar	60-84	47	Timpani	36-57
16	DrawOrgn	36-96	48	Strings1	28-96
17	PercOrgn	36-96	49	Strings2	28-96
18	RockOrgn	36-96	50	Syn.Str1	36-96
19	ChrchOrg	21-108	51	Syn.Str2	36-96
20	ReedOrgn	36-96	52	ChoirAah	48-79
21	Acordion	53-89	53	VoiceOoh	48-79
22	Harmnica	60-84	54	SynVoice	48-84
23	TangoAcd	53-89	55	Orch.Hit	48-72
24	NylonGtr	40-84	56	Trumpet	58-94
25	SteelGtr	40-84	57	Trombone	34-75
26	JazzGtr	40-86	58	Tuba	29-55
27	CleanGtr	40-86	59	Mute.Trp	58-82
28	Mute.G.tr	40-86	60	Fr.Horn	41-77
29	Ovrdrive	40-86	61	BrasSect	36-96
30	Dist.Gtr	40-86	62	SynBras1	36-96
31	GtrHarmo	40-86	63	SynBras2	36-96

PC#	Instrument	Key Range	PC#	Instrument	Key Range
64	SprnoSax	54-87	96	Rain	36-96
65	AltoSax	49-80	97	SoundTrk	36-96
66	TenorSax	42-75	98	Crystal	36-96
67	Bari.Sax	37-68	99	Atmosphr	36-96
68	Oboe	58-91	100	Bright	36-96
69	Eng.Horn	52-81	101	Goblins	36-96
70	Bassoon	34-72	102	Echoes	36-96
71	Clarinet	50-91	103	Sci-Fi	36-96
72	Piccolo	74-108	104	Sitar	48-77
73	Flute	60-96	105	Banjo	48-84
74	Recorder	60-96	106	Shamisen	50-79
75	PanFlute	60-96	107	Koto	55-84
76	Bottle	60-96	108	Kalimba	48-79
77	Shakhchi	55-84	109	Bagpipe	36-77
78	Whistle	60-96	110	Fiddle	55-96
79	Ocarina	60-84	111	Shanai	48-72
80	SquareLd	21-108	112	TnklBell	72-84
81	SawLead	21-108	113	Agogo	60-72
82	CaliopLd	36-96	114	SteelDrm	52-76
83	ChiffLd	36-96	115	Wood Block	60-72*
84	CharanLd	36-96	116	Taiko Drum	60-72*
85	VoiceLd	36-96	117	Melodic Tom	60-72*
86	FifthLd	36-96	118	Synth Drum	60-72*
87	Bass&Ld	21-108	119	Reverse Cymbal	60-72*
88	NewAgePd	36-96	120	Guitar Fret Noise	60-72*
89	WarmPad	36-96	121	Breath Noise	60-72*
90	PolySyPd	36-96	122	Seashore	60-72*
91	ChoirPad	36-96	123	Bird Tweet	60-72*
92	BowedPad	36-96	124	Telephone Ring	60-72*
93	MetalPad	36-96	125	Helicopter	60-72*
94	HaloPad	36-96	126	Applause	60-72*
95	SweepPad	36-96	127	Gunshot	60-72*

5.2 SMAF/Phrase message map

Map of assignment of sequential messages that are defined by SMAF/Phrase is as shown below.


1st byte
Upper byte
Lower byte
0 1 2 3 4 5 6 7 8 9 A B C D E F
0 1 2 3 4 5 6 7 8 9 A B C D E F
Ctl
Note
Ext

2nd byte (when the first byte is Ctl)

Lower byte \ Upper byte	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0				Prog				Prog				Prog				Prog
1				Bank				Bank				Bank				Bank
2				Oct				Oct				Oct				Oct
3				Mod				Mod				Mod				Mod
4				Bend				Bend				Bend				Bend
5																
6																
7	Shortened Vol	Shortened Bend	Shortened Mod	Ch Vol	Shortened Vol	Shortened Bend	Shortened Mod	Ch Vol	Shortened Vol	Shortened Bend	Shortened Mod	Ch Vol	Shortened Vol	Shortened Bend	Shortened Mod	Ch Vol
8																
9																
A				Pan				Pan				Pan				Pan
B				Vol				Vol				Vol				Vol
C																
D																
E																
F																

2nd byte (when the first byte is Ext)

Lower byte \ Upper byte	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NOP															Sys Ex
1																
2																
3																
4																
5																
6																
7		User Evts						<<Reserved>>								
8																
9																
A																
B																
C																
D																
E																
F																

5.3 Generation of CRC

Method of generation of SMAF standard CRC is described here by using a C like presentation. The part that becomes the object of CRC calculation is from the head of data to the point immediately before the place where CRC code is placed. However, the value of Size included in SMAF has to be a figure that includes CRC.
For the sample function makeCRC() that is used here, pointer to the place were data are present and the size (including the size for CRC) are to be set in the argument.

```
unsigned short crctable[256] =
{
  0x0000U,0x1021U,0x2042U,0x3063U,0x4084U,0x50A5U,0x60C6U,0x70E7U,
  0x8108U,0x9129U,0xA14AU,0xB16BU,0xC18CU,0xD1ADU,0xE1CEU,0xF1EFU,
  0x1231U,0x0210U,0x3273U,0x2252U,0x52B5U,0x4294U,0x72F7U,0x62D6U,
  0x9339U,0x8318U,0xB37BU,0xA35AU,0xD3BDU,0xC39CU,0xF3FFU,0xE3DEU,
  0x2462U,0x3443U,0x0420U,0x1401U,0x64E6U,0x74C7U,0x44A4U,0x5485U,
  0xA56AU,0xB54BU,0x8528U,0x9509U,0xE5EEU,0xF5CFU,0xC5ACU,0xD58DU,
  0x3653U,0x2672U,0x1611U,0x0630U,0x76D7U,0x66F6U,0x5695U,0x46B4U,
  0xB75BU,0xA77AU,0x9719U,0x8738U,0xF7DFU,0xE7FEU,0xD79DU,0xC7BCU,
  0x48C4U,0x58E5U,0x6886U,0x78A7U,0x0840U,0x1861U,0x2802U,0x3823U,
  0xC9CCU,0xD9EDU,0xE98EU,0xF9AFU,0x8948U,0x9969U,0xA90AU,0xB92BU,
  0x5AF5U,0x4AD4U,0x7AB7U,0x6A96U,0x1A71U,0x0A50U,0x3A33U,0x2A12U,
  0xDBFDU,0xCBDCU,0xFBBFU,0xEB9EU,0x9B79U,0x8B58U,0xBB3BU,0xAB1AU,
  0x6CA6U,0x7C87U,0x4CE4U,0x5CC5U,0x2C22U,0x3C03U,0x0C60U,0x1C41U,
  0xEDAEU,0xFD8FU,0xCDECU,0xDDCDU,0xAD2AU,0xBD0BU,0x8D68U,0x9D49U,
  0x7E97U,0x6EB6U,0x5ED5U,0x4EF4U,0x3E13U,0x2E32U,0x1E51U,0x0E70U,
  0xFF9FU,0xEFBEU,0xDFDDU,0xCFFCU,0xBF1BU,0xAF3AU,0x9F59U,0x8F78U,
  0x9188U,0x81A9U,0xB1CAU,0xA1EBU,0xD10CU,0xC12DU,0xF14EU,0xE16FU,
  0x1080U,0x00A1U,0x30C2U,0x20E3U,0x5004U,0x4025U,0x7046U,0x6067U,
  0x83B9U,0x9398U,0xA3FBU,0xB3DAU,0xC33DU,0xD31CU,0xE37FU,0xF35EU,
  0x02B1U,0x1290U,0x22F3U,0x32D2U,0x4235U,0x5214U,0x6277U,0x7256U,
  0xB5EAU,0xA5CBU,0x95A8U,0x8589U,0xF56EU,0xE54FU,0xD52CU,0xC50DU,
  0x34E2U,0x24C3U,0x14A0U,0x0481U,0x7466U,0x6447U,0x5424U,0x4405U,
  0xA7DBU,0xB7FAU,0x8799U,0x97B8U,0xE75FU,0xF77EU,0xC71DU,0xD73CU,
  0x26D3U,0x36F2U,0x0691U,0x16B0U,0x6657U,0x7676U,0x4615U,0x5634U,
  0xD94CU,0xC96DU,0xF90EU,0xE92FU,0x99C8U,0x89E9U,0xB98AU,0xA9ABU,
  0x5844U,0x4865U,0x7806U,0x6827U,0x18C0U,0x08E1U,0x3882U,0x28A3U,
  0xCB7DU,0xDB5CU,0xEB3FU,0xFB1EU,0x8BF9U,0x9BD8U,0xABBBU,0xBB9AU,
  0x4A75U,0x5A54U,0x6A37U,0x7A16U,0x0AF1U,0x1AD0U,0x2AB3U,0x3A92U,
  0xFD2EU,0xED0FU,0xDD6CU,0xCD4DU,0xBDAAU,0xAD8BU,0x9DE8U,0x8DC9U,
  0x7C26U,0x6C07U,0x5C64U,0x4C45U,0x3CA2U,0x2C83U,0x1CE0U,0x0CC1U,
  0xEF1FU,0xFF3EU,0xCF5DU,0xDF7CU,0xAF9BU,0xBFBAU,0x8FD9U,0x9FF8U,
  0x6E17U,0x7E36U,0x4E55U,0x5E74U,0x2E93U,0x3EB2U,0x0ED1U,0x1EF0U
};
```

```
unsigned short makeCRC(unsigned char *pData, long n)
{
  unsigned short res;
  unsigned short res1;
  unsigned short res2;
  unsigned short crc;
  unsigned char data0;

  res = 0xFFFFU;
  res1 = 0xFFFFU;
  res2 = 0xFFFFU;
  crc = 0;
  //
  while( --n >= 2 ){
    data0 = *pData++;
    res2 = res1;
    res1 = res;
    res = (res << 8) ^ crctable[(BYTE)(res >> 8) ^ data0];
    crc = (crc << 8) | data0;
  }
  // Compare CRC
  return (~res & 0xFFFFU);
}
```

5.4 Difference from SMAF/MA2 sequence data

The sequence data included in SMAF/Phrase and those of SMAF/MA2 comply basically with the same SMAF specification, however, they have the differences as described below.

▪ Resolution is fixed to 20ms.▪

▪ SMAF/Phrase has commands that are optional;
shortened Modulation, PitchBend, Modulation, and SysExMessage.

▪ Some SMAF messages are ignored by SMAF/Phrase;
BankChange, ChVolume, and shortened PitchBend

▪ SMAF/Phrase has commands that SMAF/MA2 does not have (causing NOP);
Reserved section of UserEvent.

▪ For SMAF/Phrase, it is necessary to normalize ProgramChange to 0..3.

▪ SMAF/Phrase does not support EndOfSequence(0x00, 0x00, 0x00, 0x00).

▪ LED, Motor synchronization designation is not present.

▪ Maximum number of voices is 4.

▪ Maximum number of channels is 4. (only one track)

▪ Default voice definition is present.

▪ Start/Stop point is not present.

▪ EOS definition is not present.