Scalable Polyphony MIDI

Specification and Device Profiles

Document Version 1.0a

May 24, 2002

Scalable Polyphony MIDI Specification

SP-MIDI Device 5-24 Note Profile for 3GPP

Published By:
The MIDI Manufacturers Association
Los Angeles, CA

This document is a combination of the “Scalable Polyphony MIDI Specification” and “SP-MIDI Device 5-24 Note Profile for 3GPP” Recommended Practices adopted by AMEI/MMA in 2002:

RP-034 Scalable Polyphony MIDI Specification
RP-035 SP-MIDI Device 5-24 Note Profile for 3GPP

February 2002: First printing.
May 2002: Version 1.0a (incorporating Scalable MIDI Specification errata 1.0a)

Printed 2002

MMA
PO Box 3173
La Habra CA 90632

Scalable Polyphony MIDI Specification

Version 1.0a

May 24, 2002

Published By:

The MIDI Manufacturers Association

Los Angeles, CA

PREFACE

The Scalable Polyphony MIDI Specification (SP-MIDI) provides the content creator with a mechanism to define the MIDI playback at different levels of polyphony, and includes additional guidelines for tool development and playback device behavior appropriate to scalable polyphony implementations.

This specification is derived from work of the MMA's Scalable MIDI Working Group in cooperation with AMEI and was created initially to address some unique requirements for using MIDI for ring tones in mobile phones. However, scalable playback may be equally important in other applications of MIDI in mobile devices (such as games), and so this specification is intentionally written to be as broad as possible.

What this specification does not include is a definition of all features (other than polyphony) that need to be supported by a sound generator for Scalable Polyphony content to play predictably. However, the various MMA/ AMEI specifications in the "General MIDI" and "Downloadable Sounds" families are good candidates for that purpose, when modified to accommodate SP's requirements. AMEI/MMA may also create additional specifications (or variations of existing specifications) to be used in conjunction with SP-MIDI to address various markets.

The AMEI/MMA's initial recommendations for using SP-MIDI in 3GPP (3rd generation mobile phone) applications are discussed in a separate document, titled "Scalable Polyphony MIDI Device 5-24 Note Profile for 3GPP. Additional levels of polyphony (lower and/or higher) for 3GPP are expected to be jointly developed by AMEI/MMA and 3GPP.Looking forward, the goal of the SMWG is to continue to extend the scalability of MIDI, eventually leading to a mechanism for the layering of MIDI content not just for coexistent levels of playback polyphony, but also for other synthesis features as well. These future scalability extensions will help support the development of interoperable music applications and services for devices operating in a networked environment.

Scalable Polyphony MIDI Specification
RP-034

February 2002: First printing.
May 2002: Version 1.0a (correcting previous errors in sections 2.2.1 and 3.1.3)

Printed 2002

MMA
PO Box 3173
La Habra CA 90632-3173

Table of Contents

Table of Figures

1. Introduction

1.1 Background

The Scalable Polyphony (SP) specification defines a mechanism for the flexible presentation of MIDI data to a wide range of playback devices. The specification is intended for mobile phones, PDAs, palm-top computers and other personal appliances that operate in an environment where users may create, purchase and/or exchange MIDI music with other users having different device capabilities, specifically in terms of available polyphony.

This scalable polyphony approach is an alternative to the more typical approach taken in AMEI/MMA specifications, such as General MIDI Lite (GML) [6], where device compatibility is assured by specifying a fixed polyphony. Scalable MIDI provides flexibility to the system operator and mobile terminal manufacturer to address differing customer needs, rather than forcing all customers to adopt the same requirements. For example, lower-cost phones may be offered that have only 8-note polyphony, vs. higher priced models that have 32-note polyphony, yet the same content will play on either phone. Customers in that scenario can upgrade their phones and still play all of the content that they have obtained previously, and they can share content with friends and relatives who may have different configurations.

Scalable MIDI can also help to mitigate some unique situations that might occur in wireless and battery powered systems. For example, a multi-purpose Scalable Polyphony MIDI phone or PDA could automatically drop back from 16-notes to 4-notes when more power was needed for some other application, such as decoding a video stream. Similarly, reducing polyphony would be a reasonable means for conserving battery power in some implementations.

The Scalable Polyphony MIDI specification only defines those MIDI messages that need to be supported by the playback device to provide scalable playback. In order to also provide compatibility of content with devices (interoperability), additional specifications must be used in conjunction with this one, to define all required MIDI messages and how the MIDI device will respond to them. (See Section)

1.2 Objectives

For the implementation of scalable-polyphony systems the following fundamental aspects have to be satisfied:

- The content creator has to be able to create MIDI content with embedded information about polyphony rendering requirements. Different playback devices can thus play the content in accordance with the composer's requirements.
- The MIDI messaging solution for polyphony requirements has to be supported for both locally played and streamed MIDI.
- The playback device has to be able to interpret the polyphony requirement of the content and play the content according to the polyphony requirements at the highest polyphony level it can support.
- Priority must be specified for MIDI Channels so that note generator allocation can be prioritized between Channels.
- Scalable MIDI has to be able to support several coexistent levels of polyphony so that a range of devices with different polyphony capabilities can be supported with the same content format.
- The note generator allocation for both melodic and rhythmic instruments has to be scalable, requiring more than one Rhythm Channel in some instances.

1.3 Terminology

Note is used to refer a note that begins with a Note On MIDI message and ends with a Note Off MIDI message.

Note Generator refers to rendering hardware or software needed to play one note, e.g. to respond to one Note On message and the corresponding Note Off message.

Note Stealing refers to terminating a Note and reallocating its sound generator(s) before the corresponding Note Terminating Message is received.

Note Terminating Message: a Note Off message or message with corresponding effect, e.g. Note On with velocity 0, All Notes Off, or Hold 1(Damper) with damper values between 0 and 63)

Channel Priority defines the priority order of MIDI Channels for Channel Masking and dynamic Note allocation.

Polyphony Level defines the maximum number of Notes that a Scalable Polyphony synthesizer can play simultaneously.

Channel Masking means muting (i.e., masking, or ignoring) a number of lowest-priority MIDI Channels during playback, and only playing the highest-priority Channels. The number of Channels to play is decided before playback, based on the MIP information included in the Scalable Polyphony song.

Maximum Instantaneous Polyphony (MIP) specifies the total number of Notes required for proper playback of the respective MIDI Channel together with all the higher-priority MIDI Channels (see Channel Priority). The MIP table, consisting of 16 individual MIP values, represents a cumulative polyphony requirement for the corresponding Channel Priority order defined in the Channel Priority table.

Sound Module refers to a sound-generating device that receives MIDI data in real time, typically from a **File Player** (MIDI sequencer). In mobile phone systems, for example, the terminal (handset) will contain the equivalent of a Sound Module, and might also contain a File Player.

2. Maximum Instantaneous Polyphony (MIP)

An important goal of the SP-MIDI design is to minimize the occurrence of Note Stealing and the resulting partial randomization of the music playback. The mechanism for accomplishing this is called Channel Masking, the operation of which is controlled by the Maximum Instantaneous Polyphony (MIP) message. The MIP message is a MIDI System Exclusive message for use in song data for SP-MIDI devices, indicating note usage and priority for each MIDI Channel. SP-MIDI devices use this information when processing subsequent MIDI events, to adapt to the Sound Module's current polyphony (total number of simultaneously available notes). The message indicates those layers of a song that the composer has defined as optimal for playback with different levels of polyphony. Thus it is possible to incorporate multiple versions of the same high-polyphony piece of music inside the same MIDI file. For instance, the composer can arrange a Scalable Polyphony MIDI file having maximum polyphony of 24 so that it can be played correctly, on 8, 16, and 24 note polyphony devices.

In SP-MIDI players, Channel Masking functionality filters MIDI events on a Channel-by-Channel basis in such a way as to lock out Channels which could not be played properly with the Sound Module's Polyphony Level. This Channel filtering is controlled by the MIP table (see Section 2.2), which may be shaped by the composer/arranger's taste. This is in contrast to conventional Note Stealing allocation, where notes are arbitrarily "stolen" without respect to the composer/arranger's wishes. If the SP-MIDI content and the MIP table are properly created, Note Stealing can be avoided.

2.1 MIP Message Syntax

[UNIVERSAL REALTIME SYSTEM EXCLUSIVE]

```
F0 7F <device ID> 0B 01 {cc vv} {cc vv} {cc vv} ... {cc vv} {cc vv} F7

       F0 7F         Universal Real Time SysEx header
       <device ID>   ID of target device (7F = all devices)
       0B            sub-ID#1 = Scalable Polyphony MIDI
       01            sub-ID#2 = MIP Message
       cc            MIDI Channel number (0x00 = ch1, 0x01 = ch2 etc.)
       vv            Maximum Instantaneous Polyphony (MIP) value of the
                     previous cc value
       F7            EOX
```

2.2 MIP Message Usage

The MIP message contains both a MIDI Channel Priority table and a MIP table. The MIDI Channel Priority is given by the order of the Channels in the MIP message.

The MIP value attached to each Channel is cumulative. For example, the MIP value attached to the second MIDI Channel is the total maximum instantaneous polyphony for both the first and second Channels combined. Technically, then, the MIP table represents the cumulative polyphony of all MIDI Channels containing MIDI data.

It is essential that the MIP message defines MIP table values for all Channels containing MIDI data. If the MIP message does not define MIP values for all of the player's MIDI Channels then the SP-MIDI player masks the Channels that were not received in the MIP message, i.e., sets those Channels to muted state.The content creator is responsible for defining the MIDI Channel Priority table, computing the respective polyphony as each additional MIDI Channel is used, and embedding the MIP message in the beginning of the Scalable Polyphony MIDI file. However, in practice this might also be accomplished semi-automatically using, e.g., some content creation software.

During playback, the number of Channels to be played is decided with the help of the MIP table and MIDI Channel Priority table. The recommended Algorithm for muting and un-muting MIDI Channels according to the MIP table is shown in Figure 1.

This algorithm works by traversing through all the Channels in priority order, muting all Channels with MIP greater than the supported polyphony, and un-muting all other Channels.

In the pseudo-code, vector indexing is one-based: for example, the MIP table is stored in `mip[1]–mip[ mip_length]`.

```
Inputs
polyphony:   the maximum number of Notes the player can play
             simultaneously
mip_length:  the number of entries in the MIP table
mip[]:       a vector filled with MIP values
pri[]:       a vector of the MIDI Channel numbers in
             priority order

Outputs
mute[]:      a vector of 16 Boolean values specifying
             whether to mute the corresponding MIDI Channel

Temporary variables
i:     index variable
ch:    Channel number

for i := 1 to 16 do
    mute[i] := TRUE
end for

for i := 1 to mip_length do
    ch := pri[i]
    if mip[i] <= polyphony then
        mute[ch] := FALSE
    end if
end for
```

Figure 1: Channel Masking Algorithm

The intention of the MIP mechanism is to give the composer better control over the playback of the music on various platforms. The composer can now more freely decide how different SP*nn* synthesizers should react to the content. He could, e.g., put four piano Notes to MIDI Channel 1, four percussion Notes to MIDI Channel 10 etc. A 4-note polyphonic synthesizer would then play only the piano part, an 8-note polyphonic synthesizer would play the piano plus drums, and so on. If the MIP table were made correctly, potentially disturbing Note Stealing will not take place at all.

2.2.1 Example MIP Calculation

An example of how to compute the MIP can be made using the case of a composition using MIDI Channels 1–11. It should be noted that the following MIP example is provided to illustrate the functionality of the polyphony scalability, and that the actual utilization of scalability is dependent on the musical content and interoperability requirements for supported applications or services.

In this example, the MIDI Channel Priority for the first eleven Channels is {1, 10, 2, 3, 4, 11, 5, 9, 6, 8, 7}

and the corresponding MIP figures are {4, 9, 10, 12, 12, 16, 17, 20, 26, 26, 26}, as shown in the following table.

Name	Channel	Number of Notes (over time)	MIP
Piano	1	4 Notes … 3 Notes	4
Drums	10	2 Notes, 5 Notes … 5 Notes	9
Bass	2	1 Note … 1 Note	10
Guitar	3	2 Notes … 3 Notes	12
Synth pad	4	3 Notes … 4 Notes	12
Percussion	11	4 Notes	16
Orch hit	5	1 Note	17
Guitar 2	9	3 Notes	20
Organ	6	6 Notes	26
Alien FX	8	3 Notes	26
–	7	(none)	26
…	…	…	…

Figure 2: Example Maximum Instantaneous Polyphony (MIP) calculation.

When this MIDI file is played on a 4-polyphony device, only Channel 1 is played: an 8-polyphony device also plays only Channel 1; a 16-polyphony device plays Channels 1–4 and 10-11; a 24-polyphony synthesizer plays Channels 1–5, 9 and 10–11; and finally, a 32-polyphony device plays all the Channels.

Note that the playback device should play as many Channels as it can support according to the MIP table. Therefore, using the above example, a 12-note synthesizer should play the Channels 1–4 and 10 instead of playing only the Channels 1, 10, 2, and 3, although both of these combinations have a Maximum Instantaneous Polyphony of 12 notes.

Figure 3 helps to illustrate how different Scalable Polyphony devices will play the example.

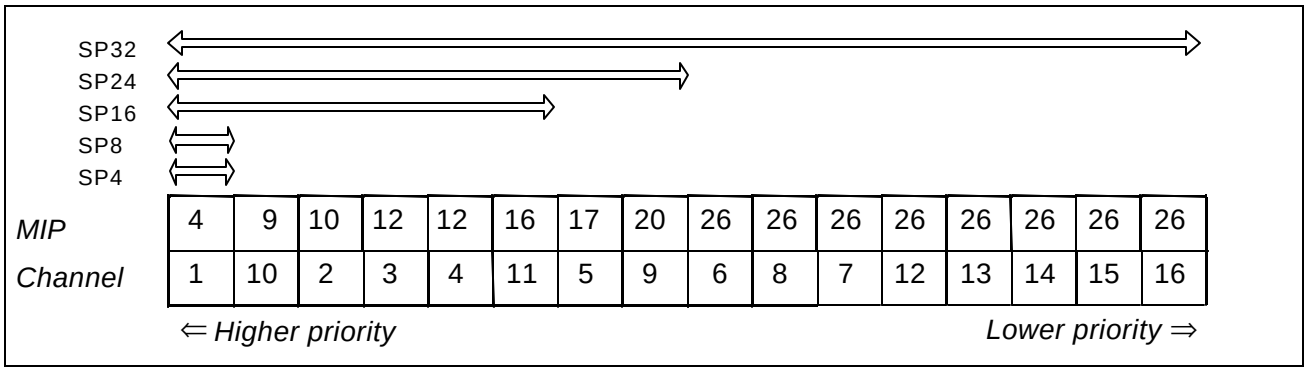

Figure 3: How different Scalable Polyphony devices play the example from Figure 2.

The actual MIP (Real Time System Exclusive) message that would be constructed from the above MIP table is given here in hexadecimal format:

```
F0 7F 7F 0B 01 00 04 09 09 01 0A 02 0C 03 0C 0A 10 04 11 08 14 05 1A 07 1A 06
1A 0B 1A 0C 1A 0D 1A 0E 1A 0F 1A F7
```

2.2.2 Manual MIP Customization

The MIP values in the example of Figure 2 were computed by counting the highest number of simultaneous Notes for each of the possible Channel combinations. This could easily be done by a content creation application, but may not result in maximizing the playback ability of target devices, so composers should also consider manual MIP customization.

As was shown in Figure 3, an 8-note polyphonic synthesizer playing the content in Figure 2 would ignore all but Channel 1, and thus not play the content any differently than a 4-note synthesizer. If the composer could expect an 8-note synthesizer to be a common target, then it might make more sense to adjust the MIP value for Channel 10 to "8". This would allow the 8-note synthesizer to play Channel 10, using its own note-stealing algorithm to reconcile the measure where 9 notes will be called to play. Editing the MIP values by hand and then playing the content on a Scalable Polyphony device (at different polyphony levels) can verify a suitable MIP value.

In Figure 4, the asterisks show how the MIP values might be changed specifically to optimize the playback for 8, 16 and 24-note polyphonic synthesizers.

Name	Channel	Number of Notes (over time)	MIP
Piano	1	4 Notes … 3 Notes	4
Drums	10	2 Notes, 5 Notes, 5 Notes	8*
Bass	2	1 Note … 1 Note	16*
Guitar	3	2 Notes … 3 Notes	16*
Synth pad	4	3 Notes, 4 Notes	16*
Percussion	11	4 Notes	16
Orch hit	5	1 Note	17
Guitar 2	9	3 Notes	20
Organ	6	6 Notes	26
Alien FX	8	3 Notes	26
–	7	(none)	26
…	…	…	…

Figure 4: MIP Values Manually Optimized (*) for 8-note and 16-note Target Devices

It is also possible to group sequential Channels together in a 'cluster' so that either all or none of them are played at specific target polyphony levels. For example, in Figure 4, the Channels {2, 3, 4, 11} are clustered together by means of setting the same MIP level for both of them. During playback, if there is not enough polyphony available in order to play Channel 11, Channels {2,3,4} are not played either.

2.3 Channel Masking Implementation Considerations

Channel Masking functionality as defined in section 2.2 must be implemented in all SP-MIDI devices. Channel Masking should be performed between the Player (MIDI Sequencer) and Sound Module (Synthesizer/Tone Generator), as shown below:

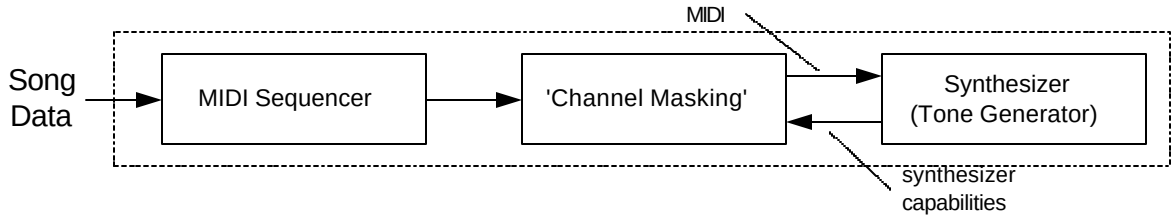

Figure 5: Integrated Player / Sound Module

Figure 5 (above) shows an integrated SP-MIDI device containing both Player and Sound Module elements. The device implementation might involve a Player program running on a CPU of some kind and a Sound Module implemented using an ASIC. Alternatively, the Sound Module might be implemented as a DSP program (or however the designer prefers). In either case, Channel Masking would probably be implemented as an extension of the Player element. However, it is also possible to implement a Player and Sound Module as distinctly separate units. A MIDI Player and Sound Module are considered to be implemented separately only if there is a means provided through which other sources (e.g. other MIDI players, devices, or programs) can send MIDI messages to the Sound Module. Special care is required for such designs, as discussed below.

2.3.1 Separate Implementation of Player and Sound Module

As shown in Figure 6 (below), Channel Masking functionality must be implemented by the Sound Module when the MIDI Player and Sound Module are implemented separately:

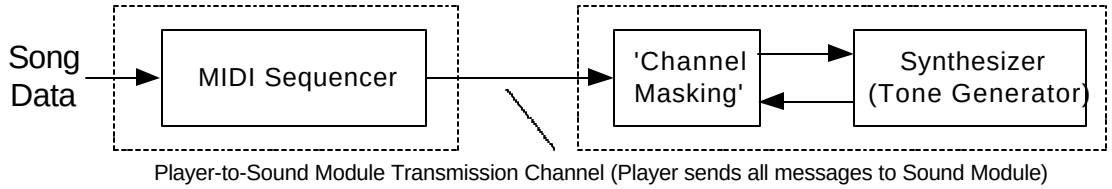

Figure 6: Separate Player and Sound Module (Basic Player)

This is required so that the Sound Module will continue to function correctly even when a nonconformant Player is connected to the Sound Module. (Note: this requirement also applies to an integrated SP-MIDI device with a Sound Module that accepts MIDI messages from other sources, as discussed in the preceding paragraph.)

Furthermore, it is recommended (but not required) that a separately-implemented MIDI Player should also implement Channel Masking functionality, as shown in Figure 7

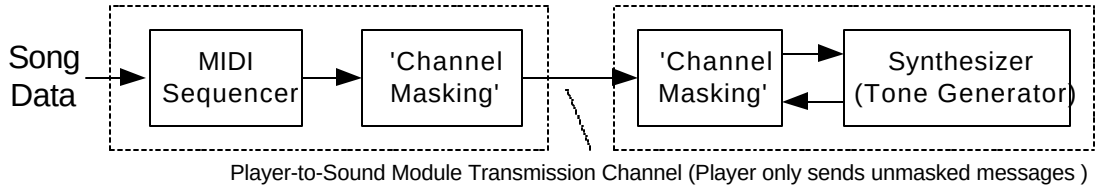

Figure 7: Separate Player and Sound Module (Enhanced Player)

The reason for implementing Channel Masking in both the MIDI Player and the Sound Module, per Figure 7, is to avoid potential timing errors caused by unnecessary message traffic to the Sound Module. (Since Channel Masking is computationally inexpensive, this should have little or no cost impact.) Noticeable timing errors can occur if the message rate exceeds the capacity of the Player-to-Sound Module transmission channel. For example, some low-polyphony designs may have a very low-bandwidth connection between the Player (implemented as software on a single-chip CPU) and the Sound Module (implemented as a separate ASIC connected to a low-speed serial channel). When a number of messages are scheduled to occur at the same time, the Player must send them sequentially rather than simultaneously. The resulting inter-message timing skew can produce audible errors, depending on the peak message load and the transmission channel capacity. Reducing message traffic (by masking irrelevant messages) is an effective way to minimize such potential timing errors.

It is worth noting that timing errors are often most apparent with percussive sounds. Therefore, it is also recommended (but not required) that messages for percussion Channels (e.g. Channels 10 and 11) be transmitted from the Player to the Sound Module before transmitting any messages for other Channels that are scheduled to occur at the same time.

Finally, when Player and Sound Module are implemented separately, the Player must always forward MIP messages to the Sound Module, even when the Player implements Channel Masking. This is required so that the Sound Module can correctly implement Channel Masking (e.g. to terminate any sounding notes on a previously-unmasked Channel which has just been masked).

2.3.2 Dynamic Channel Masking and Controller State

It is possible that previously-masked Channels may become active (unmasked) in the middle of a particular piece of music content. This could be caused by reception of a new MIP message, or even by reception of a Program Change message (because the effective available polyphony of a Sound Module may increase above the stated minimum Polyphony Level when a less-complex sound program is requested).

Unfortunately, such dynamically unmasked Channels may not be rendered as expected by the content author. This occurs when the sonic character of the rendered material depends on controller messages that were ignored because they occurred while the Channel was masked. When Channels are unmasked because of an authored MIP message, the content author can compensate for this by sending the current values of the appropriate controllers along with the MIP message. However, when Channels are unmasked because of an increase in effective available polyphony, no such remedy is possible.

To avoid such problems, it is strongly recommended that an SP-MIDI device should always retain and update the values of relevant controllers, even for Channels that are currently masked. Then, when a Channel is unmasked dynamically, the device can automatically send appropriate controller messages for that Channel to the Sound Module.

Alternatively, the Player element could quickly scan or “chase” the unmasked Channel(s) to establish the current effective controller state, and then send appropriate messages to the Sound Module to re-establish the intended controller state. (Note that when multiple messages of a given type and Channel (e.g. volume on Channel 1, expression on Channel 5) are detected within the scanned region, only the last message detected for each type and Channel should be sent to the Sound Module). However, this approach may be impractical due to processing time or memory constraints (e.g. if content data is discarded after it is played). Therefore, maintaining current controller state for all Channels, masked or unmasked, is often more practical.

It is not necessary to track (chase or maintain the state of) all possible MIDI controllers, since many are often unused. At minimum, it is strongly recommended that all controllers required by a particular SP-MIDI profile (or device specification) should be tracked. It is also recommended that other controllers in common use should be tracked, at the discretion of the manufacturer. (However, content authors are strongly advised not to count on the availability of any controllers not specified as part of the target profile.)

3. Requirements for Scalable Polyphony Devices

3.1 Response to MIP Message

A Scalable Polyphony device updates its internal Channel Priority and MIP Tables in response to a MIP Message. A device must wait until receiving an EOX (0xF7) for the MIP message before committing the update as it is necessary to first validate the contained Channels and contents of the MIP message.

Upon receiving a MIP Message, the Channel Priority and MIP Tables are updated. If the device is currently playing notes when the MIP Message arrives, it should mask and unmask Channels and restore controller states as specified in Section 2.3.2, and release notes as specified in Section 3.5.3.

The MIP message should not result in a full reset of an SP-MIDI device. Instead, each SP-MIDI device should reset only when it has received an appropriate device Reset Message (see Section 3.1.2.).

3.1.1 Initialized State of MIP and Channel Tables

When a Scalable Polyphony device is powered on or reset, it will initialize all of its MIP Table values to be equal to the device's maximum polyphony. For example, a device capable of 24 note polyphony would set each MIP Table value to 24. This effectively bypasses any MIP limitations per Channel and allows a device to allocate note generators according to its own Channel prioritization methods.

The initialized (default) state of an SP-MIDI Device's Channel Priority table is defined by the Profile or Device Specification, not the SP-MIDI Specification. For example, an SP-MIDI device based upon GM2 will set its Channel priorities however it chooses (if at all), because GM2 does not specify Channel priority. In this example, , the device may decide to ignore Channel priority altogether until receiving its first MIP message.

3.1.2 Device Reset

All SP-MIDI content should begin by sending an appropriate device Reset Message.

The appropriate Reset Message and its behavior are defined within each SP-MIDI Profile or Device Specification. For example, a device Profile based upon GM2 would specify the GM2 System On message for performing a device reset.

When a Scalable Polyphony device is reset, the device must set its MIP and Channel Tables to an initialized state as described in Section 3.1.1.

3.1.3 Setting the MIP Table

The playback device's Maximum Instantaneous Polyphony (MIP) values are set according to the 'cc vv' combinations of the message that are the MIDI Channels and corresponding MIP values in the order of priority. The message is invalid if any of the following occurs:

- The number of MIDI Channels and MIP value number pairs is higher that the number of supported MIDI Channels (i.e. 16, typically).
- Any MIDI Channel is presented more than once
- Any MIP value is smaller than the previous MIP value (this condition doesn't apply to the first MIP value)

A Scalable Polyphony playback device shall implement the pseudo-code Algorithm 1 shown in Figure 1 for muting and un-muting MIDI Channels according to the MIP table. This shall be done each time the MIP message is received. If the device cannot support one or more of the required Polyphony Levels, the device should inform the user that the content containing the MIP message is incompatible content. How (and if) the device will play incompatible content is left to the manufacturer to decide, unless otherwise stated in some applicable specification.

3.2 Number of Rhythm Channels

All Scalable Polyphony synthesizers shall support at least one Rhythm Channel, which is on Channel 10. Scalable Polyphony Profiles may specify additional requirements with respect to the number of Rhythm Channels. If a Profile requires two rhythm Channels, they are Channels 10 and 11, where Channel 10 defaults to Rhythm Channel and Channel 11 defaults to Melody Channel. If the Profile requires two Rhythm Channels, Channel 11 can be set as a Rhythm Channel via a Bank Select and Program Change messages, as defined in General MIDI 2. This allows SP-MIDI content to provide split percussion tracks that work both on a low and a high polyphony SP-MIDI device.

The recommended mechanism for supporting a second Rhythm Channel is via a Bank Select message, as described in the General MIDI 2 Specification.

3.3 Maximum and Minimum Polyphony

Conceptually, there is no lower or upper limit for polyphony that SP-MIDI devices must support, however polyphony range must be defined by a Device Profile. The level of polyphony supported by a device is expressed in terms of "SPn" where n is the maximum polyphony. Since MIP values are transmitted as 7-bit numbers, polyphony is represented in a range of 1-127. A MIP value of zero (0) is reserved and must not be used.

3.4 Number of MIDI Channels

An SP-MIDI device must be able to respond on all 16 MIDI Channels (not necessarily simultaneously). The actual Channels to be used in a specific instance are controlled with the MIP table.

3.5 Note Generator Allocation and Behavior

In general, MIDI rendering devices respond to a Note On message by attempting to allocate a note generator on which to play the new note. If any note generators are idle (currently unused) when the Note On message is received, any idle note generator may be used for the new note. This section describes the note generator allocation behavior requirements for SP-MIDI renderers in cases when no note generators are idle ('note stealing'). These requirements apply only to those Note On messages that have not been muted by the Channel Masking mechanism.

3.5.1 Note Stealing Behavior

Under normal conditions with MIDI content which is carefully prepared for SP-MIDI renderers, note stealing should occur relatively rarely, because one aim of MIP tables and Channel Masking is to reduce the frequency of note stealing. However, if no note generators are idle when a Note On message is received, the device may need to 'steal' a note generator from one of the notes that have not finished playing. The device must decide which one of those notes to steal, and there are many possible ways to make this selection.

In SP-MIDI devices, the note stealing decision should always take into account the Channel Priority Table and MIP Table from the most recently received MIP message. This is an essential aspect of the SP-MIDI Channel priority concept, since new notes on low-priority Channels should not generally steal active note generators from high-priority Channels. However, it is the responsibility of each SP-MIDI device manufacturer to design an appropriate note stealing algorithm, and there are many possible ways of using Channel Priority and MIP information in the note stealing algorithm.

The simplest recommended note stealing algorithm is as follows. First, use the Channel Priority Table and MIP Table to identify which Channel it would be best to steal a note from. This can be determined by selecting the lowest-priority Channel with active note generators in excess of its MIP value. Second,

select the best note to steal from within that Channel. Usually this will be the note whose disappearance will be the least noticeable (it is up to the manufacturer to determine which note this is).

However some manufacturers may prefer to use a more advanced algorithm, where the Channel Priority Table and MIP Table are used together with other factors to determine the best note to steal. These factors may for example include the levels, ages, or held vs. released states of the other active notes. When using such advanced algorithms, the Channel of the selected note might be different from the one selected by the simpler algorithm described in the previous paragraph (e.g. in some cases the Channel priority table may be overridden by other factors).

Example 1

MIP Message

Channel	MIP Value	Notes actually Playing on Channel	MIP Limited Notes	Notes Exceeding MIP
1	3	4	3	1
10	6	3	6	0
2	9	2	8	0
3	12	6	12	2
4	15	4	15	1
…	…	…	…	
Total		**19**	**15**	**4**

Begin Note Stealing here

Figure 8: Note Stealing Example

The example above shows a Channel Priority Table and MIP Table indicating 15 notes of polyphony, whereas the actual song contains areas using up to 19 note generators (including release periods after Note Off messages). A device capable of less than 19 notes of polyphony would use the Channel Priority and MIP Tables to determine which notes should not be played. It is recommended that the notes be stolen from the lowest-priority Channel for which the number of active note generators exceeds the Channel's MIP Value. When multiple notes within a Channel are candidates for note stealing, it is left up to the manufacturer to implement a suitable algorithm to steal the least noticeable note.

Example 2

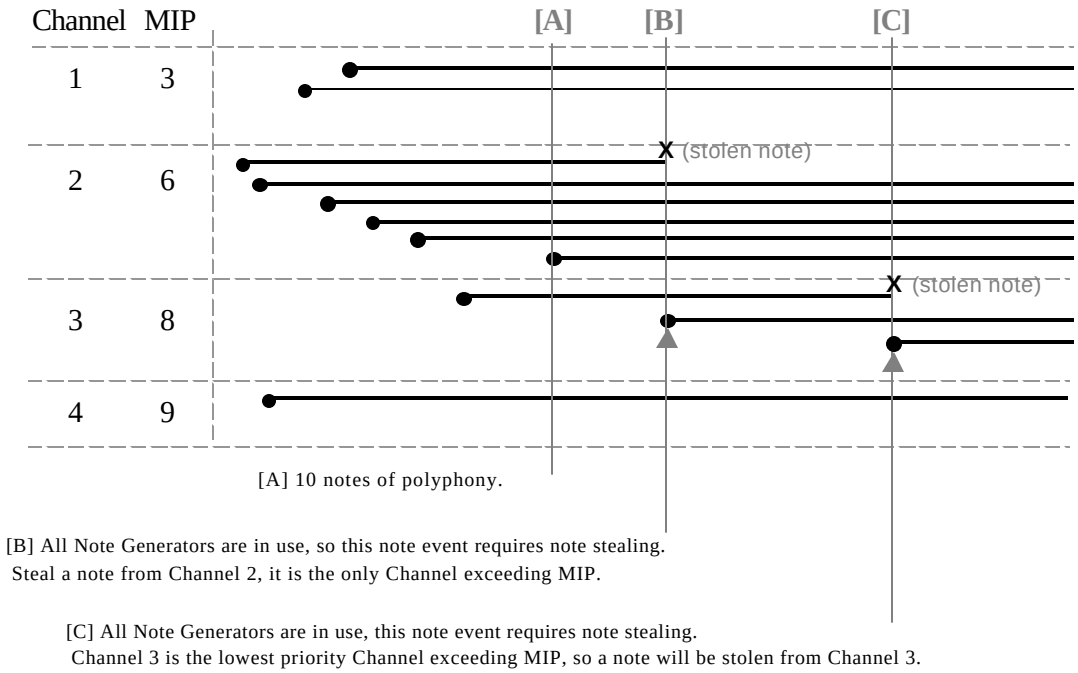

Figure 9: Note Stealing Example for a 10-note Polyphonic Synthesizer

The example in Figure 9 shows a timeline of subsequent Note On events and how Channels are prioritized for a synthesizer with 10-note polyphony. The Channels are listed in descending order of Priority (Channel 1 being the highest, Channel 4 the lowest).

Point [A] is denoted to show where the synthesizer is using all 10 of its Note Generators, any additional Notes after this point will require Note Stealing. Notice that even though Channel 2 is exceeding its MIP value, the synthesizer has enough polyphony to play all of the notes in the sequence at this point.

At Point [B] a new Note On appears on Channel 3 and there are no idle Note Generators, so a Note Generator must be stolen. With 2 notes on Channel 1 and 6 notes on Channel 2, Channel 2 is exceeding its MIP limit by 2 notes and it is currently the only Channel exceeding MIP, so it is the only Channel selected for Note Stealing. In this example, the oldest note was arbitrarily stolen, however it is expected that a manufacturer will determine which note in the selected Channel is best to steal.

At Point [C] another Note On appears on Channel 3 and again, all Note Generators are in use, so a Channel for Note Stealing must be selected. At this point, Channel 2 is exceeding MIP by 1 note and the new Note On will cause Channel 3 to exceed MIP by 1 note. Channel 3 is the lowest priority of the two, so Channel 3 is selected for Note Stealing. In this example, the oldest note was arbitrarily stolen, however it is expected that a manufacturer will determine which note in the selected Channel is best to steal.

3.5.2 MIP Message Update Behavior

If a MIP Message is received during song playback, some MIDI Channels that had not been muted before receipt of the MIP Message may become muted when the MIP Message is processed. All note generators that had been playing on the newly masked Channels should be immediately put into a "Release Note State" so that the ends of the notes will continue to sound, as though corresponding Note Off messages had been received. They should not be immediately returned to the idle or unallocated state. Notes playing on all other MIDI Channels must not be affected by such behavior. Notes received after the updated MIP message will be subject to normal Note Generator Allocation as described in section 3.5 and 3.5.1.

3.6 Future “Profiles” and Specifications

This specification does not define all of the performance features of the MIDI sound generator necessary to establish the rules for content compatibility. Instead, it is assumed that appropriate SP-MIDI content/device specifications will be developed according to market requirements.

However AMEI/MMA already have any number of device/content specifications (such as GM2) that can be used as the basis for a Scalable Polyphony content/device specification with a few modifications to enable scalable playback. A specification that modifies another specification for scalable playback is called an Scalable Polyphony MIDI ‘Device Profile’.

Specifications or Device Profiles referencing this specification must define at minimum the following:

- Minimum and maximum polyphony
- MIDI renderer specification and set of MIDI messages that must be supported. This may be expressed as a reference to an existing renderer specification such as GM1, GM2, GML, etc., or as an annotated list of MIDI messages.
- Melody and Rhythm Channel requirements and behavior
- Program Change and Bank Select requirements and behavior, including definition of required instruments.
- Device Reset and System On message and behavior

3.6.1 Interoperability of Profiles

Where more than one profile or specifications is intended to operate in the same market, care should be taken to avoid conflicting requirements that would introduce interoperability problems.

For example, the *SP-MIDI Device 5–24 Note Profile for 3GPP* is expected to be extended to support higher levels of polyphony, and to maintain upward compatibility of content, the rendering features and supported MIDI messages of the higher-polyphony profile(s) should must be a superset of the lower-polyphony profile. Observing this recommendation will allow scalable content to be created (content that plays consistently and well on more than one Profile).

If a content creator wishes to create scalable content, it is their responsibility to make sure that the low polyphony Channels that will be played on lower-polyphony profile players are free of messages not supported by that profile.

4. References

[1] *"MIDI 1.0 Detailed Specification, Document Version 4.2."* February 1996, In "The Complete MIDI 1.0 Detailed Specification, Document Version 96.1." The MIDI Manufacturers Association., Los Angeles, CA, USA.

[2] *"General MIDI System Level 1."* 1994, MMA0007/RP003, In "The Complete MIDI 1.0 Detailed Specification, Document Version 96.1." The MIDI Manufacturers Association, Los Angeles, CA, USA.

[3] *"General MIDI Level 2 Specification (Recommended Practice)."* November 1999, RP-024, The MIDI Manufacturers Association, Los Angeles, CA, USA.

[4] *"General MIDI System Level 1 Developer Guidelines."* September 1996, The MIDI Manufacturers Association, Los Angeles, CA, USA.

[5] *"The Complete MIDI 1.0 Detailed Specification, Document Version 96.1."* 1996, The MIDI Manufacturers Association, Los Angeles, CA, USA.

[6] *"General MIDI Lite, Specification for GM Lite and Guidelines for Use In Mobile Applications"* October 5, 2001, RP-033, The MIDI Manufacturers Association, Los Angeles, CA, USA.

[7] *"Scalable Polyphony MIDI Device 5-24 Note Profile for 3GPP"* May 2002, RP-035, The MIDI Manufacturers Association, Los Angeles , CA, USA.

Scalable Polyphony MIDI Device 5–24 Note Profile for 3GPP

Version 1.0

May 24, 2002

Published by:

The MIDI Manufacturers Association

Los Angeles, CA

PREFACE

The AMEI/MMA Scalable Polyphony MIDI Specification (SP-MIDI) provides a mechanism to define MIDI playback at different levels of polyphony. This 5–24 Note SP-MIDI Device Profile was created by the MMA's Scalable MIDI Working Group in cooperation with AMEI to address the unique requirements for using SP-MIDI for ring tones and more in 3rd generation mobile phones.

This specification is to be used in conjunction with the SP-MIDI Specification, and defines not just a scalable implementation, but also the means to achieving interoperability of MIDI content with devices. This Profile is intended for 3GPP devices offering 5–24 Notes of Polyphony. Additional levels of polyphony (lower and/or higher) for 3GPP are expected to be jointly developed by AMEI/MMA and 3GPP.

Scalable Polyphony MIDI Device 5–24 Note Profile For 3GPP
RP-035

February 2002: First printing.
May 2002: Changed publication date on reference to SP-MIDI specification.

Printed 2002

MMA
PO Box 3173
La Habra CA 90632-3173

Table of Contents

Table of Figures

1. Overview

This document defines a Scalable Polyphony MIDI (SP-MIDI) Profile recommendation for the 3GPP. The Profile is based on the Scalable Polyphony MIDI Specification [7] which defines the general requirements applicable to all Scalable Polyphony (SP) synthesizers. This document specifies all characteristics unique to the 3GPP Profile, and is consistent with the guidelines described in Section 3.8 of the SP specification.

The general requirements for 5–24 Note 3GPP SP-MIDI Devices are as follows:

1. The MIDI messages that devices are required to respond to are listed in Section 1.2.
2. The manner of response for those messages is defined in the AMEI/MMA General MIDI 2 Specification [3]
3. Content should not contain any MIDI messages that are not listed.
4. Devices should ignore any message that is not listed.
5. MIDI content that is not intended for scalable playback should not contain the SP-MIDI "MIP" message.

1.1 Background

The Scalable Polyphony MIDI and 3GPP MIDI Profile specifications have been created to standardize the use of MIDI data (for ring tones as well as other purposes) in 3rd generation mobile phones, and support the unique requirements of this environment. Due to diversity of terminal designs and applications, interoperability of content creates some interesting challenges, which Scalable MIDI aims to resolve.

1.1.1 Challenges:

For example, in the case of a simple content downloading service, content can certainly be stored in multiple formats for different playback devices, in which case the device accessing the service will download content in the most suitable format. But while this allows optimizing the content for each format, it does so at the expense of increased server complexity and content management costs.

A common alternative to multiple content formats is transcoding. In the case of synthetic music and audio, however, transcoding is more likely to create undesirable changes in the musical score. It is preferable to content author s that their content is maintained and distributed in its original format, to be faithful to their original musical arrangement.

Moreover, in commercial applications content protection and encryption methods may be needed, and would increase the complexity of the content format conversion procedures or the required infrastructure. In this type of environment it is preferable that the applications do not have to rely on infrastructure providing the content adaptation. Furthermore, it is desirable that the total number of different content formats can be kept small by avoiding the proliferation of content formats in the course of technology development.

1.1.2 The Solution:

This specification provided a scalable solution for a broad range of applications. This approach enables content creation for high and low polyphony content, as well as scalable content supporting multiple levels of polyphony, using the same content format.

The scalability feature can be utilized in several ways:

1. Device manufacturers can profile their products for different product categories based on customer needs and expectations.

2. Content providers can capitalize on their investment in content databases for a longer time because of the longer expected lifetime of the format.
3. Technology renewal can be supported using the same content format framework.
4. Technological transition from one complexity level to another, as well as interoperability between systems with different interoperability requirements can be supported by the scalable presentation of MIDI content. In some cases the MIDI rendering capabilities may also vary for different applications, e.g., a device having a general purpose CPU for multimedia applications may have a reduced MIDI rendering capability during a game or a multimedia show compared to idle time MIDI playback. The concept of minimum requirement loses its meaning if playback devices can satisfy the requirements only in situations where any additional CPU load is minimal and therefore most of the time fails to allocate sufficient CPU resources for MIDI playback.

From the content creator's point of view, it would be beneficial if the same content could be used in varying operating conditions and in devices belonging to different product categories. However, and especially in the case of synthesizer implementations with low polyphony capabilities, content adaptation cannot be based on the utilization of synthesizers' note-stealing technologies. The only source of reliable information about optimal polyphony scaling for arbitrary musical content is the content creator.

The Scalable MIDI standard promotes the use of MIDI by making consistent playback and content creation for this type of devices possible.

1.2 Definitions

In this document, all features are described as being either [required], [recommended], or [optional]:

[required] — The feature must be implemented as described in order to meet this Profile specification.

[recommended] — Implementation is suggested for best results, but is not required. Content authors should be aware and consider taking advantage of this feature, but since it is not guaranteed to exist, authors are advised to verify that content will play correctly with and without this feature, if desired.

[optional] — The feature may be implemented where desired. Content authors should be aware and consider taking advantage of this feature, but since it is not guaranteed to exist, authors are advised to verify that content will play correctly with and without this feature, if desired.

2. Requirements

2.1 Number of Rhythm Channels

[required]

Scalable Polyphony synthesizers conformant to this Profile shall provide at least two MIDI Channels that can function as Rhythm Channels, to enable a fluent scalable polyphony implementation. The two Channel numbers are 10 and 11, of which Channel 10 shall default to a Rhythm Channel and Channel 11 to a Melody Channel. Channel 11 can be set to a Rhythm Channel via a Bank Select and a Program Change message, as defined in General MIDI 2.

2.2 Sound Sets

2.2.1 3GPP Minimum Sound Set

[required]

Synthesizer must support the GM1 Sound Set, as listed in Figure 1, Figure 2, and Figure 3. Since the GM1 Specification only defines sound names and not their actual characteristics (timbres), manufacturers may choose to reuse timbres for musically similar sounds, but should do this with extreme caution.

If manufacturers can not support the full GM1 sound set, they can provide a subset of the GM1 sounds but this could compromise the musical performance of the synthesizer. However at minimum they must provide all of the timbres highlighted in Figure 4 and Figure 5, and shall map any unsupported GM1 sounds to the provided timbres as defined in Section 2.2.2.

2.2.2 Mapping of GM1 Sounds to the Minimum Sound Set

Figures 6 and 7 show an example of how to map GM1 instrument numbers onto the 13 minimally required instrument numbers of the Minimum Sound Set. Manufacturers are encouraged to provide more than the minimum timbres required as not doing so can cause inconsistency in the playback experience between different devices. The full GM1 bank is divided into groups of instruments that share significant characteristics in timbre, envelope, and musical function. The idea is that the instruments in a given group can replace each other without completely losing the basic musical feel of the song. The most generic instrument from each group is then selected to replace the other instruments in the same group. The boundaries of the traditional GM grouping are crossed in cases where similarities exist between instruments from different GM groups. For example, in terms of timbre and envelope, guitar and banjo are more similar to each other than banjo and bagpipe, although guitar and banjo belong to different GM groups while banjo and bagpipe belong to the same GM group.

2.2.3 3GPP Extended Sound Set

[optional]

Synthesizer supports the full General MIDI 2 Sound Set, or some subset thereof.

[required]

If the synthesizer does not support the full General MIDI 2 Sound Set, the missing GM2 timbres shall be mapped to the GM1 Sound Set by substituting all GM2 Bank Select LSB requests (those within the range 01H–09H) with a Bank Select LSB of 00H, and mapping the instrument numbers further onto the instrument numbers of the minimum sound set. Exceptions to this requirement are the GM2 SFX sounds that are not included in GM1 SFX sounds—these sounds can be left silent.

PC#	Timbre	Recommended Key Range	PC#	Timbre	Recommended Key Range
1(00H)	Acoustic Grand Piano	21–108	33(20H)	Acoustic Bass	28–55
2(01H)	Bright Acoustic Piano	21–108	34(21H)	Electric Bass (finger)	28–55
3(02H)	Electric Grand Piano	21–108	35(22H)	Electric Bass (pick)	28–55
4(03H)	Honky-tonk Piano	21–108	36(23H)	Fretless Bass	28–55
5(04H)	Electric Piano1	28–103	37(24H)	Slap Bass 1	28–55
6(05H)	Electric Piano 2	28–103	38(25H)	Slap Bass 2	28–55
7(06H)	Harpsichord	41–89	39(26H)	Synth Bass 1	28–55
8(07H)	Clavi	36–96	40(27H)	Synth Bass 2	28–55
9(08H)	Celesta	60–108	41(28H)	Violin	55–96
10(09H)	Glockenspiel	72–108	42(29H)	Viola	48–84
11(0AH)	Music Box	60–84	43(2AH)	Cello	36–72
12(0BH)	Vibraphone	53–89	44(2BH)	Contrabass	28–55
13(0CH)	Marimba	48–84	45(2CH)	Tremolo Strings	28–96
14(0DH)	Xylophone	65–96	46(2DH)	Pizzicato Strings	28–96
15(0EH)	Tubular Bells	60–77	47(2EH)	Orchestral Harp	23–103
16(0FH)	Dulcimer	60–84	48(2FH)	Timpani	36–57
17(10H)	Drawbar Organ	36–96	49(30H)	String Ensembles 1	28–96
18(11H)	Percussive Organ	36–96	50(31H)	String Ensembles 2	28–96
19(12H)	Rock Organ	36–96	51(32H)	Synth Strings 1	36–96
20(13H)	Church Organ	21–108	52(33H)	Synth Strings 2	36–96
21(14H)	Reed Organ	36–96	53(34H)	Choir Aahs	48–79
22(15H)	Accordion	53–89	54(35H)	Voice Oohs	48–79
23(16H)	Harmonica	60–84	55(36H)	Synth Voice	48–84
24(17H)	Tango Accordion	53–89	56(37H)	Orchestra Hit	48–72
25(18H)	Acoustic Guitar (nylon)	40–84	57(38H)	Trumpet	58–94
26(19H)	Acoustic Guitar (steel)	40–84	58(39H)	Trombone	34–75
27(1AH)	Electric Guitar (jazz)	40–86	59(3AH)	Tuba	29–55
28(1BH)	Electric Guitar (clean)	40–86	60(3BH)	Muted Trumpet	58–82
29(1CH)	Electric Guitar (muted)	40–86	61(3CH)	French Horn	41–77
30(1DH)	Overdriven Guitar	40–86	62(3DH)	Brass Section	36–96
31(1EH)	Distortion Guitar	40–86	63(3EH)	Synth Brass 1	36–96
32(1FH)	Guitar Harmonics	40–86	64(3FH)	Synth Brass 2	36–96

Figure 1: Melody Channel Sound Set (PC# 1–64)

Note: PC# = Program Change Number

PC#	Timbre	Recommended Key Range	PC#	Timbre	Recommended Key Range
65(40H)	Soprano Sax	54–87	97(60H)	FX 1 (rain)	36–96
66(41H)	Alto Sax	49–80	98(61H)	FX 2 (soundtrack)	36–96
67(42H)	Tenor Sax	42–75	99(62H)	FX 3 (crystal)	36–96
68(43H)	Baritone Sax	37–68	100(63H)	FX 4 (atmosphere)	36–96
69(44H)	Oboe	58–91	101(64H)	FX 5 (brightness)	36–96
70(45H)	English Horn	52–81	102(65H)	FX 6 (goblins)	36–96
71(46H)	Bassoon	34–72	103(66H)	FX 7 (echoes)	36–96
72(47H)	Clarinet	50–91	104(67H)	FX 8 (sci-fi)	36–96
73(48H)	Piccolo	74–108	105(68H)	Sitar	48–77
74(49H)	Flute	60–96	106(69H)	Banjo	48–84
75(4AH)	Recorder	60–96	107(6AH)	Shamisen	50–79
76(4BH)	Pan Flute	60–96	108(6BH)	Koto	55–84
77(4CH)	Blown Bottle	60–96	109(6CH)	Kalimba	48–79
78(4DH)	Shakuhachi	55–84	110(6DH)	Bag pipe	36–77
79(4EH)	Whistle	60–96	111(6EH)	Fiddle	55–96
80(4FH)	Ocarina	60–84	112(6FH)	Shanai	48–72
81(50H)	Lead 1 (square)	21–108	113(70H)	Tinkle Bell	72–84
82(51H)	Lead 2 (sawtooth)	21–108	114(71H)	Agogo	60–72
83(52H)	Lead 3 (calliope)	36–96	115(72H)	Steel Drums	52–76
84(53H)	Lead 4 (chiff)	36–96	116(73H)	Woodblock	*
85(54H)	Lead 5 (charang)	36–96	117(74H)	Taiko Drum	*
86(55H)	Lead 6 (voice)	36–96	118(75H)	Melodic Tom	*
87(56H)	Lead 7 (fifths)	36–96	119(76H)	Synth Drum	*
88(57H)	Lead 8 (bass + lead)	21–108	120(77H)	Reverse Cymbal	*
89(58H)	Pad 1 (new age)	36–96	121(78H)	Guitar Fret Noise	*
90(59H)	Pad 2 (warm)	36–96	122(79H)	Breath Noise	*
91(5AH)	Pad 3 (polysynth)	36–96	123(7AH)	Seashore	*
92(5BH)	Pad 4 (choir)	36–96	124(7BH)	Bird Tweet	*
93(5CH)	Pad 5 (bowed)	36–96	125(7CH)	Telephone Ring	*
94(5DH)	Pad 6 (metallic)	36–96	126(7DH)	Helicopter	*
95(5EH)	Pad 7 (halo)	36–96	127(7EH)	Applause	*
96(5FH)	Pad 8 (sweep)	36–96	128(7FH)	Gunshot	*

Figure 2: Melody Channel Sound Set (PC# 65–128)

Note: PC# = Program Change Number

Note	Timbre	Pan	Note	Timbre	Pan
24			56	Cowbell	84
25			57	Crash Cymbal 2	44
26			58	Vibra-slap	29
27			59	Ride Cymbal 2	44
28			60	High Bongo	99
29			61	Low Bongo	99
30			62	Mute Hi Conga	39
31			63	Open Hi Conga	39
32			64	Low Conga	44
33			65	High Timbale	84
34			66	Low Timbale	84
35	Acoustic Bass Drum	64	67	High Agogo	29
36	Bass Drum 1	64	68	Low Agogo	29
37	Side Stick	64	69	Cabasa	29
38	Acoustic Snare	64	70	Maracas	24
39	Hand Clap	54	71	Short Whistle [EXC2]	99
40	Electric Snare	64	72	Long Whistle [EXC2]	99
41	Low Floor Tom	34	73	Short Guiro [EXC3]	94
42	Closed Hi-hat [EXC1]	84	74	Long Guiro [EXC3]	94
43	High Floor Tom	46	75	Claves	84
44	Pedal Hi-hat [EXC1]	84	76	Hi Wood Block	99
45	Low Tom	58	77	Low Wood Block	99
46	Open Hi-hat [EXC1]	84	78	Mute Cuica [EXC4]	44
47	Low-Mid Tom	70	79	Open Cuica [EXC4]	44
48	High Mid Tom	82	80	Mute Triangle [EXC5]	24
49	Crash Cymbal 1	84	81	Open Triangle [EXC5]	24
50	High Tom	94	82		
51	Ride Cymbal 1	44	83		
52	Chinese Cymbal	44	84		
53	Ride Bell	44	85		
54	Tambourine	74	86		
55	Splash Cymbal	54	87		

Figure 3: Percussion Channel Sound Set

[Note] EXC means mutually exclusive. Instruments that have the same EXC numbers do not sound simultaneously. Playing one such sound will immediately shut off any other sound with the same EXC number.

PROG#	BANK# (MSB LSB)	GM2 TIMBRE NAME	Recommended Key Range
### Piano			
1 (00H)	79H 00H	Acoustic Grand Piano	21–108
### Chromatic Percussion			
12 (0BH)	79H 00H	Vibraphone	53–89
### Organ			
17 (10H)	79H 00H	Drawbar Organ	36–96
### Guitar			
28 (1BH)	79H 00H	Electric Guitar (clean)	40–86
### Bass			
34 (21H)	79H 00H	Electric Bass (finger)	28–55
### Strings & Orchestral instruments			
41 (28H)	79H 00H	Violin	55–96
### Ensemble			
49 (30H)	79H 00H	String Ensembles 1	28–96
### Brass			
57 (38H)	79H 00H	Trumpet	58–94
### Reed			
67 (42H)	79H 00H	Tenor Sax	42–75
### Pipe			
74 (49H)	79H 00H	Flute	60–96
### Synth Lead			
82 (51H)	79H 00H	Lead 2 (sawtooth)	21–108
### Synth Pad			
90 (59H)	79H 00H	Pad 2 (warm)	36–96
### Percussive			
115 (72H)	79H 00H	Steel Drums	52–76

Figure 4: Minimum Sound Bank (Melodic Sound Set Timbres)

NOTE#	INSTRUMENT NAME	PAN	NOTE#	INSTRUMENT NAME	PAN
36 (C2)	Bass Drum	64	51 (D#3)	Ride Cymbal	44
40 (E2)	Electric Snare	64	54 (F#3)	Tambourine	74
42 (F#2)	Closed Hi-hat [EXC1]	84	62 (D4)	Mute High Conga	39
45 (A2)	Low Tom	58	64 (E4)	Low Conga	44
46 (A#2)	Open Hi-hat [EXC1]	84	70 (A#4)	Maracas	24
49 (C#3)	Crash Cymbal	84	75 (D#5)	Claves	84
50 (D3)	High Tom	94			

Figure 5: Minimum Sound Bank (Percussion Sound Set Timbres)

The first instrument in each group can be used to replace the rest of the instruments. All SFX sounds (programs #120 Reverse Cymbal, #121 Guitar Fret Noise, #122 Breath Noise, #123 Seashore, #124 Bird Tweet, #125 Telephone, #126 Helicopter, #127 Applause and #128 Gunshot) must be implemented or mapped to program numbers suitable for the manufacturer.

PROG#	Patch name
1	**Acoustic Piano**
2	Bright Acoustic Piano
3	Electric Grand Piano
4	Honky-tonk Piano
5	Electric Piano1
6	Electric Piano 2
7	Harpsichord
8	Clavi

PROG#	Patch name
12	**Vibraphone**
9	Celesta
10	Glockenspiel
11	Music Box
13	Marimba
14	Xylophone
15	Tubular Bells
16	Dulcimer
46	Pizzicato Strings
47	Orchestral Harp
99	FX 3 (crystal)
109	Kalimba

PROG#	Patch name
17	**Drawbar Organ**
18	Percussive Organ
19	Rock Organ
20	Church Organ
21	Reed Organ
22	Accordion
23	Harmonica
24	Tango Accordion
110	Bag pipe

PROG#	Patch name
28	**Electric Guitar (clean)**
25	Acoustic Guitar (nylon)
26	Acoustic Guitar (steel)
27	Electric Guitar (jazz)
29	Electric Guitar (muted)
30	Overdriven Guitar
31	Distortion Guitar
32	Guitar Harmonics
105	Sitar
106	Banjo
107	Shamisen
108	Koto

PROG#	Patch name
34	**Electric Bass (finger)**
33	Acoustic Bass
35	Electric Bass (pick)
36	Fretless Bass
37	Slap Bass 1
38	Slap Bass 2
39	Synth Bass 1
40	Synth Bass 2

PROG#	Patch name
41	**Violin**
42	Viola
43	Cello
44	Contrabass
111	Fiddle

PROG#	Patch name
49	**Strings**
45	Tremolo Strings
50	String Ensembles 2
51	Synth Strings 1
52	Synth Strings 2

PROG#	Patch name
57	**Trumpet**
58	Trombone
59	Tuba
60	Muted Trumpet
61	French Horn
62	Brass Section
63	Synth Brass 1
64	Synth Brass 2

PROG#	Patch name
67	**Tenor Sax**
65	Soprano Sax
66	Alto Sax
68	Baritone Sax
69	Oboe
70	English Horn
71	Bassoon
72	Clarinet
112	Shanai

PROG#	Patch name
74	**Flute**
73	Piccolo
75	Recorder
76	Pan Flute
77	Blown Bottle
78	Shakuhachi
79	Whistle
80	Ocarina

PROG#	Patch name
82	**Lead (Saw)**
81	Lead 1 (square)
83	Lead 3 (calliope)
84	Lead 4 (chiff)
85	Lead 5 (charang)
86	Lead 6 (voice)
87	Lead 7 (fifths)
88	Lead 8 (bass + lead)

PROG#	Patch name
90	**Pad**
53	Choir Aahs
54	Voice Oohs
55	Synth Voice
89	Pad 1 (new age)
91	Pad 3 (polysynth)
92	Pad 4 (choir)
93	Pad 5 (bowed)
94	Pad 6 (metallic)
95	Pad 7 (halo)
96	Pad 8 (sweep)
97	FX 1 (rain)
98	FX 2 (soundtrack)
100	FX 4 (atmosphere)
101	FX 5 (brightness)
102	FX 6 (goblins)
103	FX 7 (echoes)
104	FX 8 (sci-fi)

PROG#	Patch name
115	**Pitched Percussion**
48	Timpani
56	Orchestra Hit
113	Tinkle Bell
114	Agogo
116	Woodblock
117	Taiko Drum
118	Melodic Tom
119	Synth Drum

Figure 6: Optional Sound Mapping (Melodic Instruments)

The first instrument in each group can be used to replace the rest of the instruments.

Key	Key #	Sound name
C2	**36**	**Bass drum**
	35	Acoustic Bass Drum
E2	**40**	**Electric Snare**
	38	Acoustic Snare
F#2	**42**	**Closed Hi-hat**
	44	Pedal Hi-hat [EXC1]
	71	Short Whistle [EXC2]
	80	Mute Triangle [EXC5]
A#2	**46**	**Open Hi-hat**
	55	Splash Cymbal
	58	Vibra-slap
	74	Long Guiro [EXC3]
	81	Open Triangle [EXC5]
	72	Long Whistle [EXC2]
D3	**50**	**High Tom**
	48	High Mid Tom
A2	**45**	**Low Tom**
	43	High Floor Tom
	41	Low Floor Tom
	47	Low-Mid Tom
C#3	**49**	**Crash**
	57	Crash Cymbal 2

Key	Key #	Sound name
D#3	**51**	**Ride**
	52	Chinese Cymbal
	53	Ride Bell
	59	Ride Cymbal 2
F#3	**54**	**Tambourine**
	39	Hand Clap
D4	**62**	**Mute High Conga**
	60	High Bongo
	65	High Timbale
	78	Mute Cuica [EXC4]
E4	**64**	**Low Conga**
	61	Low Bongo
	63	Open Hi Conga
	66	Low Timbale
	79	Open Cuica [EXC4]
A#4	**70**	**Maracas**
	69	Cabasa
	73	Short Guiro [EXC3]
D#5	**75**	**Claves**
C#2	37	Side stick
	56	Cowbell
	67	High Agogo
	68	Low Agogo
	76	High Wood Block
	77	Low Wood Block

Figure 7: Optional Sound Mapping (Percussion Instruments)

2.3 Telephone Ring Vibrator Control

[optional]

The implementation of MIDI control for the telephone ring vibrator is optional, but if the control is implemented, it must meet the specifications defined here.

The telephone ring vibrator is a standard functionality of mobile phones, used for silent alarms. One of the prime uses of SP-MIDI content is to compose telephone ring tones, and the ring vibrator may readily be used to enhance them. The telephone ring vibrator is played on a Melody Channel using the instrument bank and program numbers given below. The program and bank numbers have been selected not to clash with the GM2 sound set.

PROG#	**BANK# (MSB LSB)**	**PROGRAM NAME**	**Recommended Key Range**
### SFX			
125 (7CH)	79H 06H	Telephone Ring Vibrator	*

Figure 8: Optional Ring Vibrator

3. MIDI Messages

3.1 Supported Messages

This list is a subset of messages supported by GM2 Devices.

The manner in which the device is to respond to each of these MIDI messages is as defined in the General MIDI 2 Specification (version 1.1).

Controller messages	Supported messages	Notes
Response to MIDI Channel Messages		
Note On/ Note Off	■	
Program Change Message	■	
Control Change Messages		
Bank Select (cc#0/32)	■	
Modulation Depth (cc#1)	■	
Portamento Time (cc#5)		
Channel Volume (cc#7)	■	
Pan (cc#10)	■	Required for stereo synths only.
Expression (cc#11)	■	
Hold1 (Damper) (cc#64)	■	
Portamento ON/OFF (cc#65)		
Sostenuto (cc#66)		
Soft (cc#67)		
Filter Resonance (Timbre/Harmonic Intensity) (cc#71)		
Release Time (cc#72)		
Attack time (cc#73)		
Brightness (cc#74)		
Decay Time (cc#75)		
Vibrato Rate (cc#76)		
Vibrato Depth (cc#77)		
Vibrato Delay (cc#78)		
Reverb Send Level (cc#91)		
Chorus Send Level (cc#93)		
Data Entry (cc#6/38)	■	
RPN LSB/MSB (cc#100/101)	■	
RPN (Registered Parameter Numbers)		
00H / 00H Pitch Bend Sensitivity	■	
00H / 01H Channel Fine Tuning		
00H / 02H Channel Coarse Tuning		
00H / 05H Modulation Depth Range (Vibrato Depth Range)		
7FH / 7FH (RPN NULL)	■	
Channel Mode Messages		
All Sound Off (cc#120)	■	
Reset All Controllers (cc#121)	■	
All Notes Off (cc#123)	■	
Omni Mode Off (cc#124)		
Omni Mode On (cc#125)		
Mono Mode On (Poly Mode Off) (cc#126)		
Poly Mode On (Mono Mode Off) (cc#127)		
Pitch Bend	■	
Channel Pressure	■	

■ Requred implementation for interoperable 3GPP services supporting this profile

Figure 9: Required MIDI Messages (1 of 2)

Controller messages	Supported messages	Notes
Universal System Exclusive Messages		
Master Volume	■	
Master Fine Tuning		
Master Coarse Tuning		
Reverb Parameters		
Reverb Type		
Reverb Time		
Chorus Parameters		
Chorus Type		
Mod Rate		
Mod Depth		
Feedback		
Send to Reverb		
Controller Destination Setting		
Channel Pressure (Aftertouch)		
Controller (Control Change)		
Scale/Octave Tuning Adjust		
Key-Based Instrument Controllers		
MIP Message	■	
GM System Messages		
GM2 System On	■	
GM1 System On	■	
GM System Off		

Figure 10: Required MIDI Messages (2 of 2)

3.1.1 Pan Control (CC 10)

The required implementation of Pan (stereo panning) Control for the GM2 Specification has been modified from the original in order to align the GM2 response with GML and DLS response. Pan response in this Profile is intended to match the revised GM2 response, so implementers are cautioned to consult the updated version (1.1) of the GM Specification for the correct response.

3.1.2 Master Volume (Universal SysEx)

Default Value: 7FH/7FH

Sets the overall volume of the entire device. As with cc#7 and cc#11, the square of the value is proportional to the volume. Should follow the curve definition for cc#7 in GM2.

The Master Volume message must not be included in the song data content to achieve fade ins and fade outs. MIDI Master Volume messages are solely intended to be used as a system command for controlling the overall playback level in the mobile device.

3.2 Device Reset and System On Behavior

A device may be reset by sending either a GM2 System On or GM1 System On Message.

For GM2 song data, the device is reset by sending a GM2 System On Message

F0 7E <device ID> 09 03 F7

The device response to this message follows the GM2 specification for GM2 System On *and* the Scalable Polyphony MIDI specification for Device Reset.

For GM1 song data, the device is reset by sending a GM1 System On Message

F0 7E <device ID> 09 01 F7

The device response to this message follows the GM1 and GM2 specification for GM1 System On *and* the Scalable Polyphony MIDI specification for Device Reset.

When either of these messages is received, all currently sounding Notes immediately mute without producing a click, and the device is reset to the initial GM status corresponding to the System On message that was sent. Additionally, the Scalable Polyphony device will reset its MIP and Channel Tables to an initialized state.

The reset operation shall be completed within 100ms after receiving a GM2 or GM1 System On message.

3.3 Interoperability of Future Profiles

It is strongly recommended that any future profiles (e.g. those addressing higher or lower levels of polyphony) are prepared with care to avoid introducing interoperability problems between different profile specifications.

To maintain upward compatibility of MIDI content in the 3GPP environment, the rendering features and supported MIDI messages of higher-polyphony profiles should always be a superset of lower-polyphony profiles. Observing this recommendation will allow scalable content to be created, in other words content that plays consistently and well on more than one profile player.

If a content creator wishes to create scalable content, it is their responsibility to make sure that the low polyphony Channels that will be played on lower-polyphony profile players are free of messages not supported by that profile.

4. References

[1] *"MIDI 1.0 Detailed Specification, Document Version 4.2."* February 1996, In "The Complete MIDI 1.0 Detailed Specification, Document Version 96.1." The MIDI Manufacturers Association, Los Angeles, CA, USA.

[2] *"General MIDI System Level 1."* 1994, MMA0007/RP003, In "The Complete MIDI 1.0 Detailed Specification, Document Version 96.1." The MIDI Manufacturers Association, Los Angeles, CA, USA.

[3] *"General MIDI Level 2 Specification (Recommended Practice)."* November 1999, RP-024, The MIDI Manufacturers Association, Los Angeles, CA, USA.

[4] *"General MIDI System Level 1 Developer Guidelines."* September 1996, The MIDI Manufacturers Association, Los Angeles, CA, USA.

[5] *"The Complete MIDI 1.0 Detailed Specification, Document Version 96.1."* 1996, The MIDI Manufacturers Association, Los Angeles, CA, USA.

[6] *"General MIDI Lite, Specification for GM Lite and Guidelines for Use In Mobile Applications."* October 2001, RP-033, The MIDI Manufacturers Association, Los Angeles, CA, USA.

[7] *"Scalable Polyphony MIDI Specification."* May 2002 (corrected), RP-034, The MIDI Manufacturers Association, Los Angeles, CA, USA.