Universal Serial Bus
Device Class Definition
for
MIDI Devices

Release 2.0

May 5, 2020

Scope of this Revision

This document is the Universal Serial Bus Device Class Definition for MIDI Devices Version 2. This version follows the Universal Serial Bus Device Class Definition for MIDI Devices Version 1.0. Version 1.0 supports all MIDI 1.0 specifications. The primary goal for this Version 2.0 update is add support for MIDI 2.0 specifications published in 2020 by the MIDI Manufacturers Association and the Association of Musical Electronics Industry. Other enhancements are also included in this Version 2.0. Backward compatibility is given careful consideration.

Version 1.0 Specification Contributors

Gal Ashour	IBM Corporation
Billy Brackenridge	Microsoft Corporation
Mike Kent	Roland Corporation
Geert Knapen	Philips ITCL-USA
Oren Tirosh	Altec Lansing

Version 2.0 Specification Contributors

Franz Detro	Native Instruments
Harumichi Hotta	Yamaha Corporation
Mike Kent	Roland Corporation – Project Chair, mikekent@mk2audio.com
Geert Knapen	Knowles Corporation
Makoto Komorita	Roland Corporation
Daisuke Miura	Yamaha Corporation
Matt Mora	Apple Corporation
Torrey Walker	Apple Corporation
Doug Wyatt	Apple Corporation
Ichiro Yazawa	Roland Corporation

Revision History

Revision	Date	Description
1.0	Nov. 1, 1999	First revision of Universal Serial Bus Device Class Definition for MIDI Devices. Supports MIDI 1.0.
2.0	May 5, 2020	Second revision of Universal Serial Bus Device Class Definition for MIDI Devices. Adds support for MIDI 2.0, MIDI-CI, and the Universal MIDI Packet.

USB Device Class Definition for MIDI Devices

Please send comments via electronic mail to audio-chair@usb.org

Table of Contents

1 Introduction 8
1.1 Background: MIDI 1.0, MIDI 2.0, and USB 8
1.2 Purpose 8
Figure 1: Simple USB MIDI Interface 9
Figure 2: Simple USB MIDI Synthesizer 9
1.3 Related Documents 10
1.4 Terms and Abbreviations 10
1.5 Reserved Words and Specification Conformance 11
Table 1-1: Words Relating to Specification Conformance 11
Table 1-2: Words Not Relating to Specification Conformance 11
2 Management Overview 12
2.1 Overview of what is new or changed from Version 1.0 class specification 12
3 Functional Characteristics 14
3.1 Device and Topology 14
3.1.1 MIDI Streaming Interface with Two Alternate Settings: Backward Compatibility 15
3.2 Data Format: Universal MIDI Packet (UMP) 15
3.2.1 Basic Packet Format 16
3.2.2 UMP Messages in a USB Packet: Byte Ordering 17
3.2.3 UMP Message Type Field and Packet Size 17
Table 3-1: Packet Sizes based on Message Types 17
3.2.4 UMP Group Field and Routing 18
3.3 MIDI Streaming Interface 18
3.3.1 USB MIDI Converter 18
3.3.2 MIDI Streaming Data Endpoints 18
3.3.3 Group Terminals and UMP Groups 19
3.3.4 Group Terminal Blocks 19
4 Operational Model 21
4.1 Communication from Host to USB MIDI Function 22
4.2 Communication from USB MIDI Function to Host 22
5 Configuration Descriptors 23
5.1 Core Descriptors 23
5.2 MIDIStreaming Interface Descriptors 23
5.2.1 Standard MS Interface Descriptor 23
Table 5-1: Standard MIDIStreaming Interface Descriptor 23
5.2.2 Class-Specific MS Interface Descriptor 24
Table 5-2: Class-Specific MS Interface Header Descriptor 24
5.3 MIDI Streaming Endpoint Descriptors 24
5.3.1 Standard MIDI Streaming Data Endpoint Descriptor 24
Table 5-3: Standard MS Data Endpoint Descriptor 25

5.3.2 Class-Specific MIDI Streaming Data Endpoint Descriptor 25
Table 5-4: Class-specific MS Data Endpoint Descriptor 26
5.4 Class-Specific Group Terminal Block Descriptors – Retrievable by a Separate Get Request 27
5.4.1 Class Specific Group Terminal Block Header Descriptor 27
Table 5-5: Class-Specific Group Terminal Header Descriptor 27
5.4.2 Group Terminal Block Descriptor 27
Table 5-6: Group Terminal Block Descriptor 29
6 Class Specific Command: Group Terminal Blocks Descriptors Request 31
Appendix A. Audio Device Class Codes: MIDIStreaming 32
A.1 MS Class-Specific Interface Descriptor Types 32
A.1 MS Class-Specific Interface Descriptor Subtypes 32
A.2 MS Class-Specific Endpoint Descriptor Subtypes 32
A.3 MS Class-Specific Group Terminal Block Descriptor Subtypes 32
A.4 MS Interface Header MIDIStreaming Class Revision 33
A.5 MS MIDI IN and OUT Jack types 33
A.6 Group Terminal Block Type 33
A.7 Group Terminal Default MIDI Protocol 33
A.8 Group Terminal Number (Universal MIDI Packet Group) 33
Appendix B. Example 1: Simple MIDI Instrument (Informative) 35
B.1 Example 1 Device and Configuration Descriptor 36
B.1.1 Device Descriptor 36
Table B-1: Device Descriptor 36
B.1.2 Configuration Descriptor 36
Table B-2: Configuration Descriptor 36
B.2 Example 1 AudioControl Interface Descriptors 37
B.2.1 Standard AC Interface Descriptor 37
B.2.2 Class-specific AC Interface Descriptor 37
B.3 Example 1 MIDIStreaming Interface Descriptors (On Alternate Setting 0x00 for USB MIDI 1.0) 38
B.3.1 Standard MS Interface Descriptor 38
B.3.2 Class-specific MS Interface Descriptor 38
B.3.3 MIDI IN Jack Descriptor 38
B.3.4 MIDI IN Jack Descriptor 39
B.3.5 MIDI OUT Jack Descriptor 39
B.3.6 MIDI OUT Jack Descriptor (External) 39
B.4 Example 1 Bulk OUT Endpoint Descriptors (On Alternate Setting 0x00 for USB MIDI 1.0) 40
B.4.1 Standard Bulk OUT Endpoint Descriptor 40
B.4.2 Class-specific MS Bulk OUT Endpoint Descriptor 40
B.5 Example 1 Bulk IN Endpoint Descriptors (On Alternate Setting 0x00 for USB MIDI 1.0) 40
B.5.1 Standard Bulk IN Endpoint Descriptor 40
B.5.2 Class-specific MS Bulk IN Endpoint Descriptor 41

B.6 Example 1 MIDIStreaming Interface Descriptors (On Alternate Setting 0x01 for USB MIDI 2.0) 41
B.6.1 Standard MS Interface Descriptor 41
B.6.2 Class-specific MS Interface Header Descriptor 41
B.7 Example 1 Bulk OUT Endpoint Descriptors (On Alternate Setting 0x01 for USB MIDI 2.0) 42
B.7.1 Standard Bulk OUT Endpoint Descriptor 42
B.7.2 Class-specific MS Bulk OUT Endpoint Descriptor 42
B.8 Example 1 Interrupt IN Endpoint Descriptors (On Alternate Setting 0x01 for USB MIDI 2.0) 42
B.8.1 Standard Interrupt IN Endpoint Descriptor 42
B.8.2 Class-specific MS Interrupt IN Endpoint Descriptor 43
B.9 Example 1 Group Terminal Block Descriptors (On Alternate Setting 0x01 for USB MIDI 2.0) 43
B.9.1 Class Specific Group Terminal Block Header Descriptor 43
B.9.2 Group Terminal Block Descriptor 44
B.10 Example 1 Get Descriptor Requests 44
B.11 Example 1 String Descriptors 45
Appendix C. MIDI-CI Topology Limitation (Informative) 47

List of Tables

Table 1-1: Words Relating to Specification Conformance 11
Table 1-2: Words Not Relating to Specification Conformance 11
Table 3-1: Packet Sizes based on Message Types 17
Table 5-1: Standard MIDIStreaming Interface Descriptor 23
Table 5-2: Class-Specific MS Interface Header Descriptor 24
Table 5-3: Standard MS Data Endpoint Descriptor 25
Table 5-4: Class-specific MS Data Endpoint Descriptor 26
Table 5-5: Class-Specific Group Terminal Header Descriptor 27
Table 5-6: Group Terminal Block Descriptor 29
Table B-1: Device Descriptor 36
Table B-2: Configuration Descriptor 36
Table B-3: Standard AC Interface Descriptor 37
Table B-4: Class Specific AC Interface Descriptor 37
Table B-5: Standard MS Interface Descriptor 38
Table B-6: Class-specific MS Interface Header Descriptor 38
Table B-7: MIDI IN Jack Descriptor (Embedded) 38
Table B-8: MIDI IN Jack Descriptor (External) 39
Table B-9: MIDI OUT Jack Descriptor (Embedded) 39

Table B-10: MIDI OUT Jack Descriptor (External) 39
Table B-11: Standard Bulk OUT Endpoint Descriptor 40
Table B-12: Class Specific Bulk OUT Endpoint Descriptor 40
Table B-13: Standard Bulk IN Endpoint Descriptor 40
Table B-14: Class Specific Bulk IN Endpoint Descriptor 41
Table B-15: Standard MS Interface Descriptor 41
Table B-16: Class-specific MS Interface Header Descriptor 41
Table B-17: Standard Bulk OUT Endpoint Descriptor 42
Table B-18: Class-specific Bulk OUT Endpoint Descriptor 42
Table B-19: Standard Interrupt IN Endpoint Descriptor 42
Table B-20: Class Specific Interrupt IN Endpoint Descriptor 43
Table B-21: Class-Specific Group Terminal Block Header Descriptor 43
Table B-22: Group Terminal Block Descriptor 44
Table B-25: String Descriptor ID 0 45
Table B-26: String Descriptor ID 1 45
Table B-27: String Descriptor ID 2 45
Table B-28: String Descriptor ID 3 46
Table B-29: String Descriptor ID 4 46

List of Figures

Figure 1: Simple USB MIDI Interface 9
Figure 2: Simple USB MIDI Synthesizer 9
Figure 3: USB MIDI 2.0 Device and Function Topology 14
Figure 4: Basic Message Format 16
Figure 5: Example Message Layout 16
Figure 6: Example Byte Ordering 17
Figure 7: Examples of Group Terminals Blocks 20
Figure 8: USB MIDI 2.0 Device – Interleaving up to 16 Groups on Each Endpoint 21

1 Introduction

Following is Version 2 of the USB Device Class Definition for MIDI Devices. It is designed to cover the widest range of possible MIDI applications and products. This document must be considered an integral part of the USB Audio Device Class Definition.

1.1 Background: MIDI 1.0, MIDI 2.0, and USB

MIDI, introduced in 1983, is a mature standard with many existing products and applications. It is a standard that defines data protocol and communication mechanisms for exchange of musical control information. The original USB Device Class Specification for MIDI Devices Version 1.0, adopted by the USB-IF in 1999, provides USB transport for MIDI 1.0 messages (commonly called USB MIDI). USB MIDI adoption has grown over the past 20 years and USB MIDI is now the most widely used transport for MIDI.

In 2020 the Association of Musical Electronics Industry and the MIDI Manufacturers Association adopted a set of MIDI 2.0 specifications. The MIDI 2.0 specifications define a new Universal MIDI Packet for both MIDI 1.0 and MIDI 2.0 protocols. Therefore, this new USB Device Class Specification for MIDI Devices Version 2.0 is defined to support new MIDI 1.0 and MIDI 2.0 devices using the Universal MIDI Packet. Transferring the Universal MIDI Packet over USB via this specification will be called USB MIDI 2.0 through the rest of this document.

1.2 Purpose

This specification for Universal MIDI Packet exchange over USB is designed to be an elegant method to enable a wide range of MIDI 1.0 and MIDI 2.0 system configurations, from the simplest to the most complex. Furthermore, this specification expands on MIDI, allowing applications not possible with non-USB MIDI 2.0 configurations.

Figure 1 shows a simple MIDI interface, which would allow many existing non-USB MIDI devices to connect to USB.

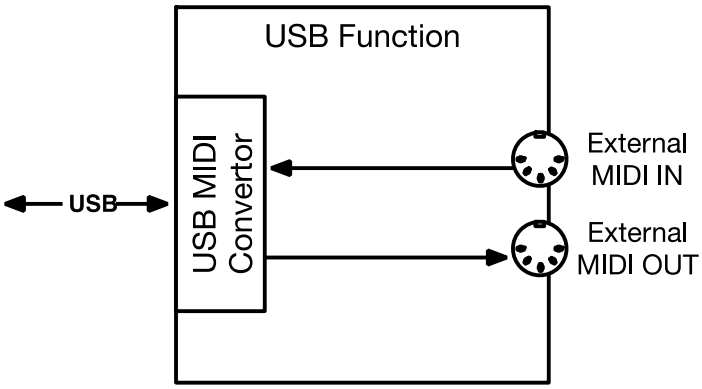


Figure 1: Simple USB MIDI Interface

Figure 2 shows a MIDI synthesizer with a keyboard and a tone generator.

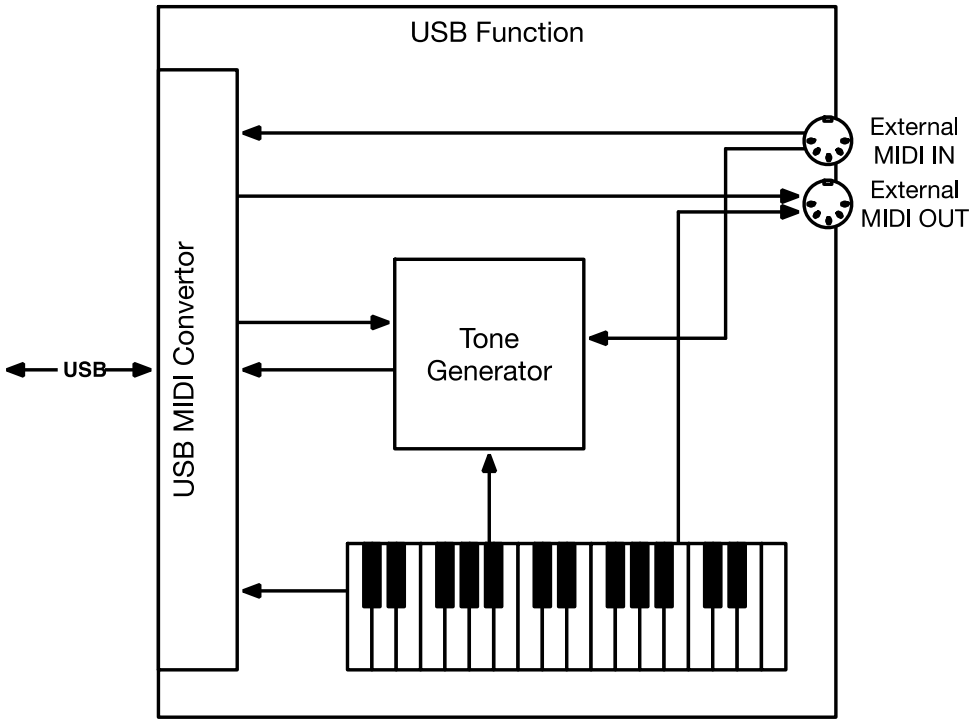


Figure 2: Simple USB MIDI Synthesizer

1.3 Related Documents

- *Universal Serial Bus Specification*, 1.0 final draft revision (also referred to as the *USB Specification*). In particular, see Chapter 9, "USB Device Framework".
- *Universal Serial Bus Specification*, Revision 2.0 (referred to in this document as the *USB 2 Specification*). In particular, see Chapter 5, "USB Data Flow Model" and Chapter 9, "USB Device Framework."
- *Universal Serial Bus 3.2 Specification*, Revision 1.0 (referred to in this document as the *USB 3 Specification*). This document covers details specific to SuperSpeed and SuperSpeed+ Devices.
- *Universal Serial Bus Device Class Definition for Audio Devices Version 1.0*.
- *Universal Serial Bus Device Class Definition for MIDI Devices Version 1.0*.
- *The Complete MIDI 1.0 Detailed Specification, Document Version 96.1, Third Edition,* (available from http://www.midi.org).
- *MIDI 2.0 Specification Overview* (available from http://www.midi.org).
- *MIDI Capability Inquiry (MIDI-CI) Specification* (available from http://www.midi.org).
- *Universal MIDI Packet Format and MIDI 2.0 Protocol* (available from http://www.midi.org).

MIDI is a standard for musical instruments and audio devices to communicate with each other and with computers. For more information contact MIDI Manufacturers Association at https://www.midi.org/ or Association of Musical Electronics Industry (Japan) at http://www.amei.or.jp/.

1.4 Terms and Abbreviations

Group: In every Universal MIDI Packet there is a 4-bit **Group** field following the Message Type field, addressing every UMP MIDI Message (and every UMP comprising any given MIDI Message) to one of 16 Groups. The 16 Groups are interleaved into one MIDI stream.

MIDI-CI: MIDI Capability Inquiry. A MIDI mechanism for bidirectional queries and negotiations. Requires a pair of a MIDI Input and a MIDI Output. When using the Universal MIDI Packet, as is the format of this USB MIDI 2.0 specification, MIDI-CI requires a pair of an IN Group Terminal and an OUT Group Terminal.

MIDI Message: (1) A complete MIDI 1.0 message as specified in ***The Complete MIDI 1.0 Detailed Specification***, including any updates, addenda, or errata); or (2) A complete MIDI 1.0 message or a complete MIDI 2.0 message as specified in the ***Universal MIDI Packet Format and MIDI 2.0 Protocol*** specification. Note that for the USB in this specification, a single MIDI Message might span multiple transport packets, or a single transport packet might contain multiple MIDI Messages.

Message Type: The most significant 4 bits in every Universal MIDI Packet contain the **Message Type** field. It indicates the message's general functional area (e.g., Utility, MIDI 1.0 Channel Voice Messages, MIDI 2.0 Channel Voice Messages), as well as the UMP's size.

MS: MIDI Streaming.

UMP: Universal MIDI Packet.

Universal MIDI Packet: The Universal MIDI Packet is a data container format which supports all MIDI 1.0 Protocol messages and all MIDI 2.0 Protocol messages and is intended for use on all defined transports. See ***Universal MIDI Packet Format and MIDI 2.0 Protocol Specification*** for detailed definition.

USB MIDI 1.0: The Universal Serial Bus Device Class Definition for MIDI Devices, Version 1.0

USB MIDI 2.0: The Universal Serial Bus Device Class Definition for MIDI Devices, Version 2.0 (this specification).

1.5 Reserved Words and Specification Conformance

In this document, the following words are used solely to distinguish what is required to conform to this specification, what is recommended but not required for conformance, and what is permitted but not required for conformance:

Table 1-1: Words Relating to Specification Conformance

Word	Reserved For	Relation to Spec Conformance
shall	Statements of requirement	Mandatory. A conformant implementation conforms to all 'shall' statements.
should	Statements of recommendation	Recommended but not mandatory. An implementation that does not conform to some or all 'should' statements is still conformant, providing all 'shall' statements are conformed to.
may	Statements of permission	Optional. An implementation that does not conform to some or all 'may' statements is still conformant, providing all 'shall' statements are conformed to.

By contrast, in this document, the following words are never used for specification conformance statements; they are used solely for descriptive and explanatory purposes:

Table 1-2: Words Not Relating to Specification Conformance

Word	Reserved For	Notes
must	Statements of unavoidability	Describes an action to be taken that, while not required (or at least not directly required) by this specification, is unavoidable. Not used for statements of conformance requirement (see 'shall' above).
will	Statements of fact	Describes a condition that as a question of fact is necessarily going to be true, or an action that as a question of fact is necessarily going to occur, but not as a requirement (or at least not as a direct requirement) of this specification. Not used for statements of conformance requirements (see 'shall' above).
can	Statements of capability	Describes a condition or action that a system element is capable of possessing or taking. Not used for statements of conformance permission (see 'may' above).
might	Statements of possibility	Describes a condition or action that a system element is capable of electing to possess or take. Not used for statements of conformance permission (see 'may' above).

2 Management Overview

The USB Device Class Specification for MIDI Devices Version 1.0, adopted by the USB-IF in 1999, provides USB transport for MIDI 1.0 messages (commonly called USB MIDI). This version 2.0 specification adds support for MIDI 2.0 messages and other future messages that are contained in a Universal MIDI Packet.

USB is well suited for connecting MIDI Interfaces and MIDI instruments to computers. MIDI is a recognized protocol for music control that is serving the marketplace very well. USB MIDI adoption has grown over the past 20 years and USB MIDI is now the most widely used transport for MIDI.

This version 2.0 of the USB Device Class Specification for MIDI Devices adds support for the new features defined by MIDI 2.0, primarily by adopting the Universal MIDI Packet defined by MIDI 2.0. It also adds some additional features beyond those found in the original version of the class specification and deprecates some features that have been replaced over the years by superior mechanisms.

In principle, a versatile bus like USB provides many ways to handle MIDI data. For the industry, however, it is very important that MIDI transport mechanisms be well defined and standardized on USB. Only in this way can we predict interoperability, guarantee reliable performance, and maintain the good market image of MIDI itself. Standardized MIDI transport mechanisms also help keep software drivers as generic as possible. The MIDIStreaming Interface Class described in this document satisfies those requirements. It is written and revised by experts in the MIDI field. Other device classes that address MIDI in some way should refer to this document for their MIDI specification.

This document contains all necessary information for a designer to build a USB-compliant device that incorporates MIDI 1.0 and/or MIDI 2.0 functionality. It specifies the standard and class specific descriptors that must be present in each USB MIDI Function. It further explains the transfer of MIDI events, using the Universal MIDI Packet for standardized transfer over the USB and for easy handling by MIDI devices. The MIDI data itself is transferred transparently, without any changes. Furthermore, if the MIDI 2.0 specification is updated, new MIDI events or definitions that use the Universal MIDI Packet are fully supported.

2.1 Overview of what is new or changed from Version 1.0 class specification

- Replaced the 32-bit USB MIDI Event Packet with the Universal MIDI Packet to support both MIDI 1.0 Protocol and MIDI 2.0 Protocol.
- MIDI IN/OUT Jacks replaced by IN/OUT Group Terminals
- IN/OUT Group Terminals represented in Blocks to show association between multiple IN/OUT Group Terminals.
- Added Bandwidth descriptors for more predictable use of higher speeds.
- Added support for Interrupt transactions as well as Bulk (USB MIDI 1.0 uses Bulk only) for more deterministic control over jitter and throughput.
- Removed the Element functional entity. The features of the Element were rarely supported and have been replaced by better mechanisms in MIDI-CI Profile Configuration and MIDI-CI Property Exchange. This removal has a side effect of removing the Element's association to Audio Terminals.

- Removed the Transfer Endpoints. These were never supported and the target application is better achieved by newer USB mechanisms including Mass Storage class.
- Expose any fixed pairings of IN/OUT Groups/Ports to Host OS for use with MIDI-CI.
- Define how MIDI 2.0 devices may also connect to legacy systems with only USB MIDI 1.0 support. Devices will be limited to MIDI 1.0 functionality when connected to a USB MIDI 1.0 Host.

3 Functional Characteristics

USB MIDI functionality resides at the interface level in the USB Device Framework hierarchy. The MIDI Streaming interface represents the entire functionality of the USB MIDI function. It is defined as a subclass of the Audio Interface Class. But this subclass assignment remains for legacy compatibility to the prior USB MIDI class only.

3.1 Device and Topology

USB MIDI 2.0 devices use a MIDI Streaming Interface to support USB MIDI 1.0 Functions and USB MIDI 2.0 Functions.

- Topology, descriptors and other details of the USB MIDI 2.0 Function are defined in this specification.
- Topology, descriptors and other details of the USB MIDI 1.0 Function are defined in the USB Device Class Specification for MIDI Devices Version 1.0.

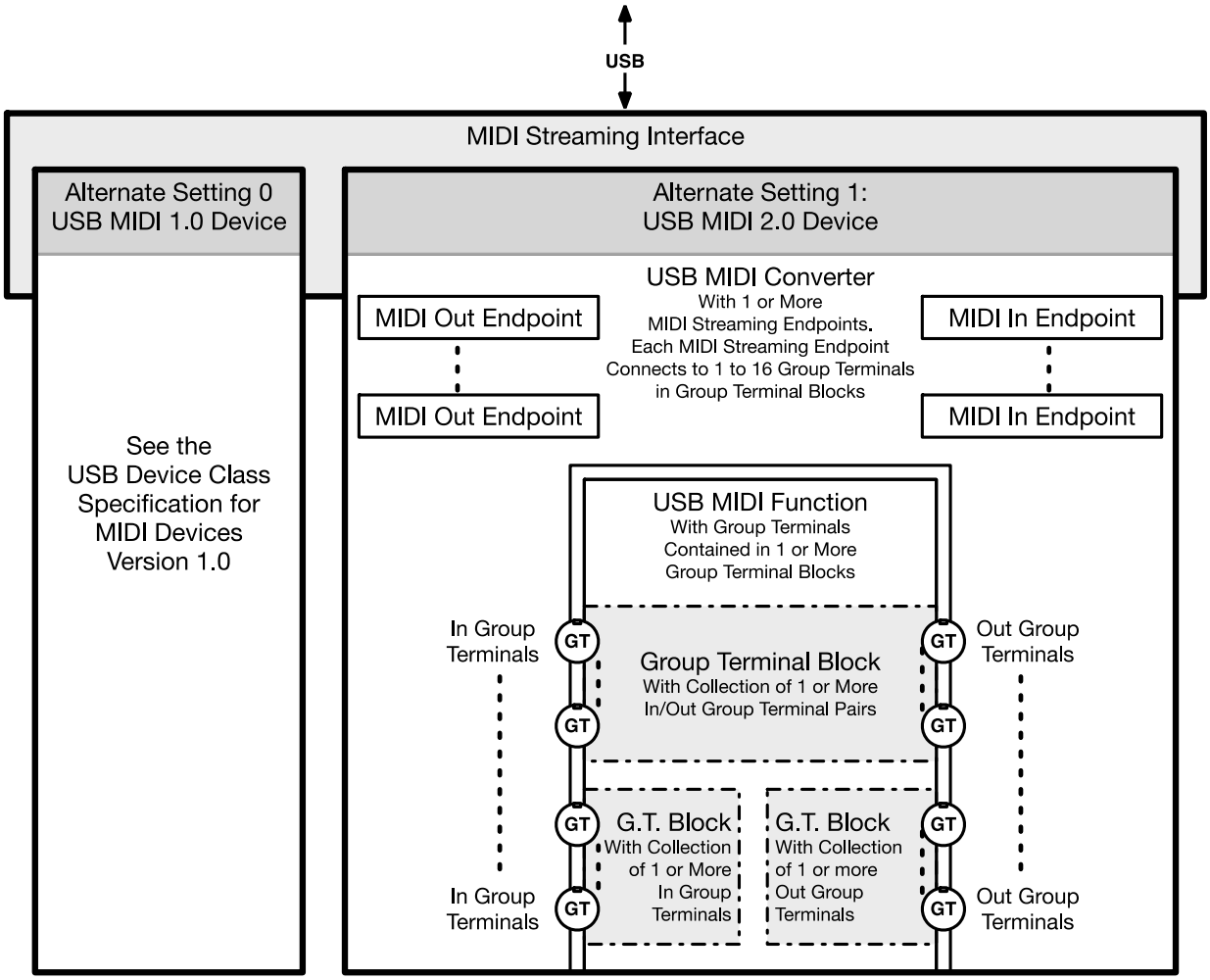


Figure 3: USB MIDI 2.0 Device and Function Topology

A device declares whether each Alternate Setting includes descriptors and functionality for either version 1.0 or version 2.0 of the USB Device Class Specification for MIDI Devices by setting the

value of the bcdMSC field in the Class-Specific MIDI Streaming Interface Header to either 0x0100 (version 1.0) or 0x0200 (version 2.0).

3.1.1 MIDI Streaming Interface with Two Alternate Settings: Backward Compatibility

This 2.0 version of the USB Device Class Specification for MIDI Devices is not backwards compatible with any of the previous versions of the same specification.
However, to ensure the broadest possible interoperability among new USB MIDI 2.0 Devices and legacy USB MIDI 1.0 Hosts, this specification defines that a USB MIDI 2.0 compliant Device should do the following:

The first MIDI Streaming Interface exposed by the Device should contain an Alternate Setting 0 which is compliant with USB Device Class Specification for MIDI Devices Version 1.0. In other words, if Host software selects the first indexed Alternate Setting on the MIDI Streaming Interface, the Device should expose a MIDI Function that is compliant with USB MIDI 1.0. The bcdMSC field in the Class-Specific MIDI Streaming Interface Header shall be set to 0x0100. This requirement allows the MIDI Function to interoperate with existing Hosts exactly the same as current legacy MIDI 1.0 Functions.

The first MIDI Streaming Interface exposed by the Device should contain an Alternate Setting 1 which contains the preferred device implementation which conforms to the new design defined by this USB MIDI 2.0 specification. The bcdMSC field in the Class-Specific MIDI Streaming Interface Header shall be set to 0x0200. Further Alternate Settings which conform to the new design defined by this USB MIDI 2.0 specification may be included in Alternate Setting 2 or higher.

USB MIDI Hosts capable of supporting a USB MIDI 2.0 device may look for a MIDI Function that complies with the USB Device Class Specification for MIDI Devices Version 2.0 on Alternate Setting 1. If such an Alternate Setting is found in the device, the Host may install the device as a USB MIDI 2.0 device using the descriptors found on Alternate Setting 1.
The USB Device Class Definition for MIDI Devices Version 1.0 implemented on Alternate Setting #0 supports MIDI 1.0 devices using MIDI 1.0 messages.
The USB Device Class Definition for MIDI Devices Version 2.0 implemented on Alternate Setting #1 supports MIDI 1.0 devices using MIDI 1.0 messages and MIDI 2.0 devices using MIDI 1.0 messages or MIDI 2.0 messages.
It must be clearly understood that the part of the discussion involving Host behavior is merely there to complete the full picture and is not imposing any implementation rules on Host software. This specification is limited to specifying USB MIDI function behavior only.

3.2 Data Format: Universal MIDI Packet (UMP)

MIDI data is transferred over USB using Universal MIDI Packets as defined by MIDI 2.0 specifications. The Universal MIDI Packet contains two 4 bit fields which are critical to the design of USB MIDI 2.0:

1. Message Type Field
2. Group Field

3.2.1 Basic Packet Format

Each MIDI message is contained in a Universal MIDI Packet. The Universal MIDI Packet consists of one to four 32 bit words. Each Universal MIDI Packet contains one MIDI 1.0 Protocol message or one MIDI 2.0 Protocol message.
Diagrams of the Universal MIDI Packets are presented with one 32 bit word per line.

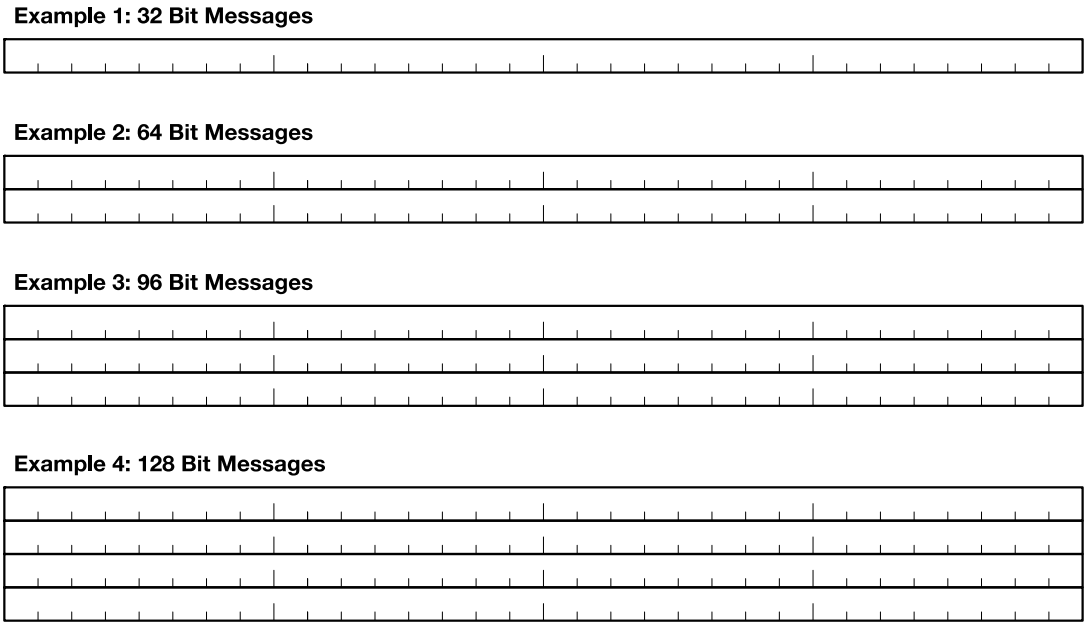


Figure 4: Basic Message Format

The diagrams in this specification show a logical view where the leftmost bits are the most significant bits in the word and in each field.

3.2.1.1 Example MIDI Messages in the Universal MIDI Packet Format

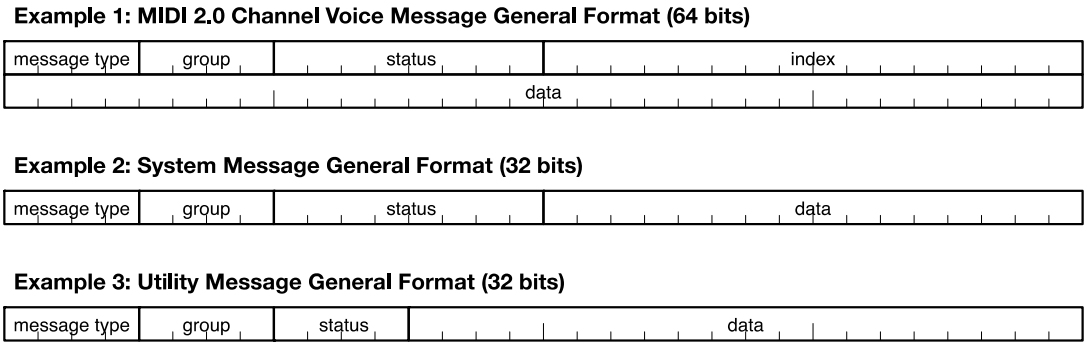


Figure 5: Example Message Layout

The USB MIDI 2.0 only needs to know about the Message Type and Group fields of messages in the Universal MIDI Packet in order to transport MIDI messages. USB MIDI 2.0 does not

understand or interact with the contents of any other field in a Universal MIDI Packet such as Status, Channel, Index, and Data fields of messages.

3.2.2 UMP Messages in a USB Packet: Byte Ordering

Each 32 bit word of a Universal MIDI Packet is sent with the least significant byte first. If a Universal MIDI Packet is longer than one 32 bit word, then the order of the words is retained.

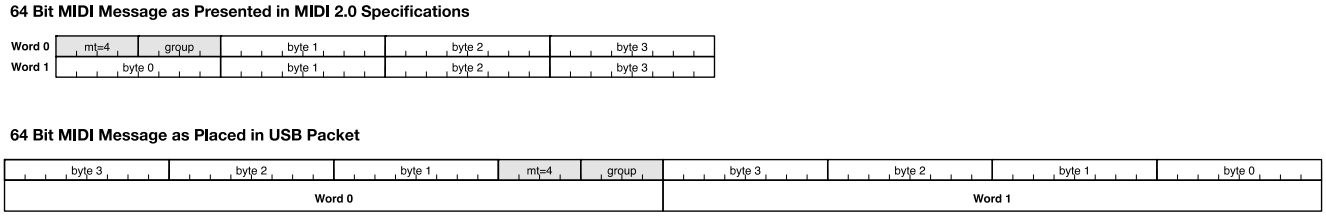


Figure 6: Example Byte Ordering

3.2.3 UMP Message Type Field and Packet Size

The most significant 4 bits of every message contain the Message Type (MT). All messages within a Message Type have the same Universal MIDI Packet size. The USB MIDI 2.0 needs to know about Packet Size based on the Message Type.

When a Universal MIDI Packet is sent on the USB and whenever possible, the entire Universal MIDI Packet should be kept together and sent within the same USB Bulk Transfer Transaction or within the same USB Interrupt Transfer Transaction.

When the USB is using burst transfers, splitting Universal MIDI Packets across USB Transactions is generally beyond of the control of the device (and class specific driver). But devices with support for burst transfers are generally more powerful processors capable of holding part of a Universal MIDI Packet until the remainder of the Universal MIDI Packet arrives.

Table 3-1: Packet Sizes based on Message Types

Message Type	Packet Size	Description
0x0	32 bits	Utility Messages
0x1	32 bits	System Real Time and System Common Messages (except System Exclusive)
0x2	32 bits	MIDI 1.0 Channel Voice Messages
0x3	64 bits	Data Messages (including System Exclusive)
0x4	64 bits	MIDI 2.0 Channel Voice Messages
0x5	128 bits	Data Messages
0x6	32 bits	Reserved
0x7	32 bits	Reserved
0x8	64 bits	Reserved
0x9	64 bits	Reserved
0xA	64 bits	Reserved

Message Type	Packet Size	Description
0xB	96 bits	Reserved
0xC	96 bits	Reserved
0xD	128 bits	Reserved
0xE	128 bits	Reserved
0xF	128 bits	Reserved

3.2.4 UMP Group Field and Routing

Every packet has a 4-bit Group field following the Message Type field. The Universal MIDI Packet format uses 16 Groups as a mechanism to create sixteen interleaved streams of MIDI data. For the sake of clarity, this specification refers to these Groups as UMP Groups.

- 0x0 to 0xF = Group 1 to Group 16

USB MIDI 2.0 uses the Group field of MIDI messages in an interleaved stream to route MIDI messages from a source to an associated destination. Each active UMP Group is represented by a Group Terminal at the boundary between the USB MIDI Function and the USB MIDI Converter. The value of the Group field in each Universal MIDI Packet indicates the associated Group Terminal Number where the Universal MIDI Packet is sent or received.

3.3 MIDI Streaming Interface

USB MIDI 2.0 Devices contain several device components. All USB MIDI functions must contain some MIDI Streaming Data Endpoints connected to some Group Terminals. Group Terminals include IN Group Terminals and OUT Group Terminals. The USB MIDI Function contains some Group Terminal Blocks. Each Group Terminal Block declares Group Terminals as members of that Group Terminal Block to indicate that those Group Terminals are associated to each other. A MIDI Streaming Data Endpoint connects to a Group Terminal by declaring the ID of the Group Terminal Block which has that Group Terminal as a member. See the previous device topology diagram (Figure **3**: USB MIDI 2.0 Device and Function Topology).

3.3.1 USB MIDI Converter

The USB MIDI Converter is the heart of every USB MIDI function since it provides the link between the Host and the USB MIDI function. Therefore, it is a mandatory building block. On one side, it interfaces with the USB pipes that are used to exchange MIDI data streams between the Host and the USB MIDI function's MIDI data endpoints. On the other side, it connects to a number of Group Terminals. The Group Terminal is a logical concept to represent a single UMP Group of Data for true MIDI connectivity to the USB MIDI function. The USB MIDI Converter provides the link between a MIDI OUT Endpoint and associated IN Group Terminals. Likewise, it converts between an OUT Group Terminal and the corresponding MIDI IN Endpoint.

3.3.2 MIDI Streaming Data Endpoints

The USB MIDI Converter contains one or more MIDI Streaming Data Endpoints which function as MIDI IN and/or MIDI OUT Endpoints. Each MIDI Streaming Data Endpoint transfers a single Universal MIDI Packet stream with up to sixteen UMP Groups of MIDI Data.

A Class-Specific MIDI Streaming Data Endpoint Descriptor declares the unique identification number of one or more associated Group Terminal Blocks in the **bAssoGrpTrmBlkID** field. The set of all Group Terminal Blocks associated to the MIDI Streaming Data Endpoint have a collective total of one to sixteen Group Terminals as members to represent one to sixteen UMP Groups in the Universal MIDI Packet stream.
Only a single MIDI IN Endpoint may associate to any specific Group Terminal Block. Only a single MIDI OUT Endpoint may associate to any specific Group Terminal Block. However, a single MIDI IN Endpoint and a single MIDI OUT Endpoint may simultaneously associate to a shared Group Terminal Block for the purposes of creating MIDI-CI In/Out pairs of Group Terminals.
MIDI Streaming Data Endpoints use bulk or interrupt transfers to exchange data with the Host. Consequently, a large quantity of MIDI data can simultaneously be sent by an application without missing any MIDI events. Therefore, music applications can perform complex MIDI operations, including sending many MIDI Note On messages at the same time to more smoothly play the most complex music.
The MIDI Streaming Data Endpoint descriptors are further detailed in Section 5.3 of this document.

3.3.3 Group Terminals and UMP Groups

Group Terminals are entities that pass MIDI Data In/Out of the USB MIDI Function to/from the MIDI Streaming Endpoints in the USB MIDI Converter.
Each Group Terminal is a logical concept to represent a connection for a single UMP Group of MIDI Data. Inside the USB MIDI 2.0 Function, each UMP Group is treated as a separately routable stream to provide the greatest flexibility of routing to various components of the Device and to MIDI applications in the Host.
Each UMP Group of data coming out of the USB MIDI Function is represented by an OUT Group Terminal.
Each UMP Group of data going into the USB MIDI Function is represented by an IN Group Terminal.
The number for each Group Terminal is the UMP Group Number of MIDI messages on that connection:

- 0x0 to 0xF = Group Terminal 1 to Group Terminal 16
- Group Terminal 1 to Group Terminal 16 carry respectively Group 1 to Group 16 data.

Devices should implement Group Terminals starting with Group 1 and incrementing from there (i.e. a device with only one Group Terminal should use data of Group 1).
Each Group Terminal is a member of a block of Group Terminals, a Group Terminal Block. Group Terminals are declared in a Group Terminal Block descriptor. See Section 3.3.4.

3.3.4 Group Terminal Blocks

Group Terminal Blocks are entities in the USB MIDI Function.
A Group Terminal Block has one or more Group Terminals as members. If a Group Terminal Block has more than one Group Terminal as members, those Group Terminals have some functional relationship. Functional relationships include:

1. Matching Input/Output pair(s) of Group Terminals for use in MIDI-CI Transactions
2. More than one Group Terminal in use by a component of the USB MIDI Device.
3. A combination of 1) and 2) above within the same Block.

The relationship between the Group Terminals should be described in a string descriptor for the Group Terminal Block. The relationship described is most often the MIDI source or sink such as a synthesizer or controller.

Each Group Terminal Block within the USB MIDI function is assigned a unique identification number, contained in the **bGroupTrmBlkID** field of the descriptor. The value 0x00 is reserved for undefined ID, effectively restricting the total number of addressable Group Terminal Blocks in the USB MIDI Function to 255.

Besides uniquely identifying all addressable Group Terminal Blocks in a USB MIDI function, the IDs also serve to describe data connections between MIDI Streaming Data Endpoints and Group Terminal members of each Group Terminal Block.

Figure 7 shows example implementations of Group Terminal Blocks:

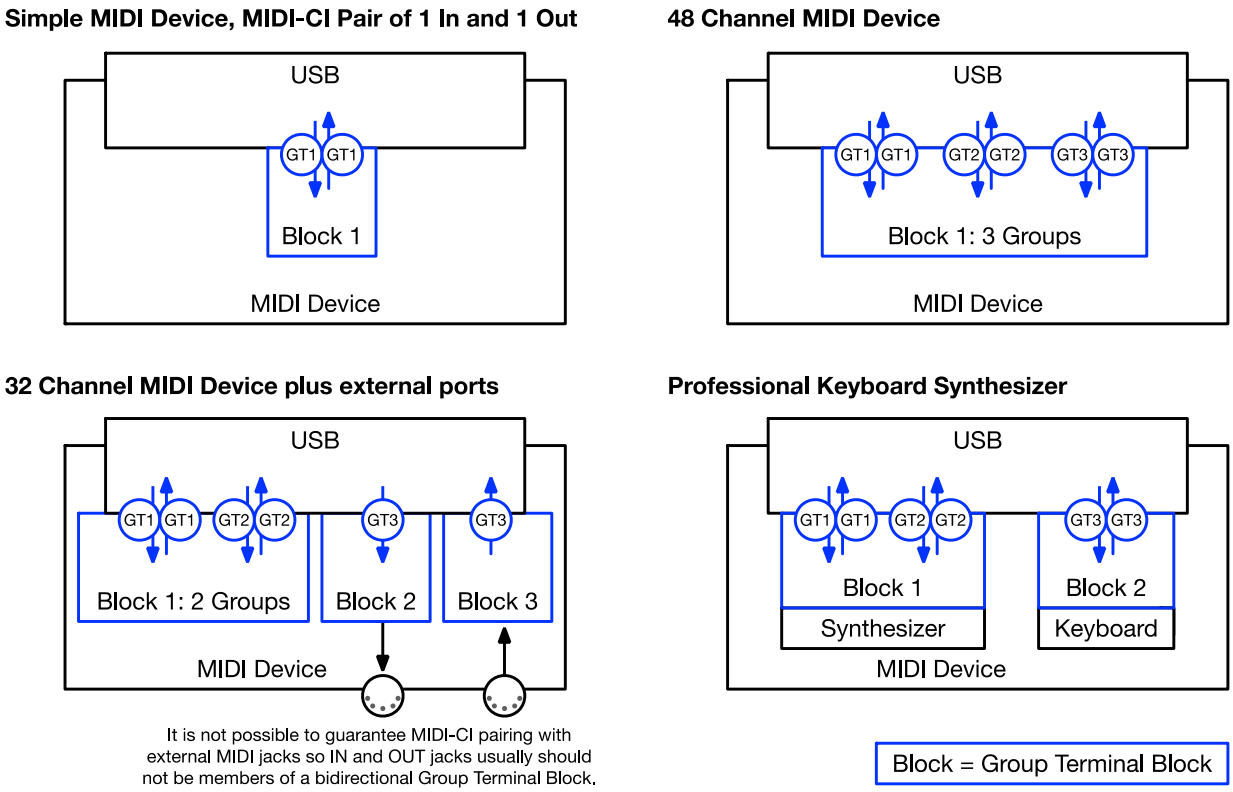


Figure 7: Examples of Group Terminals Blocks

The Group Terminal Block descriptor is further detailed in Section 5.4 of this document.

4 Operational Model

The USB MIDI function exposes a single MIDIStreaming interface that is used by Host software to interact and control the entire MIDI functionality of the function. Since all control messages are performed in-band (buried into the MIDI data stream) there is no need for a separate control and status interface. The MIDIStreaming interface contains one or more MIDI Endpoints, which all use bulk or interrupt transfers to exchange MIDI data with the Host.

The following paragraphs describe the data paths involved when communication is taking place between multiple MIDI applications on the Host and multiple USB MIDI components or external MIDI appliances, connected to the USB MIDI function. However, it must be clearly understood that the part of the discussion involving Host behavior is merely there to complete the full picture and is not imposing any implementation rules on Host software. This specification is limited to specifying USB MIDI function behavior only.

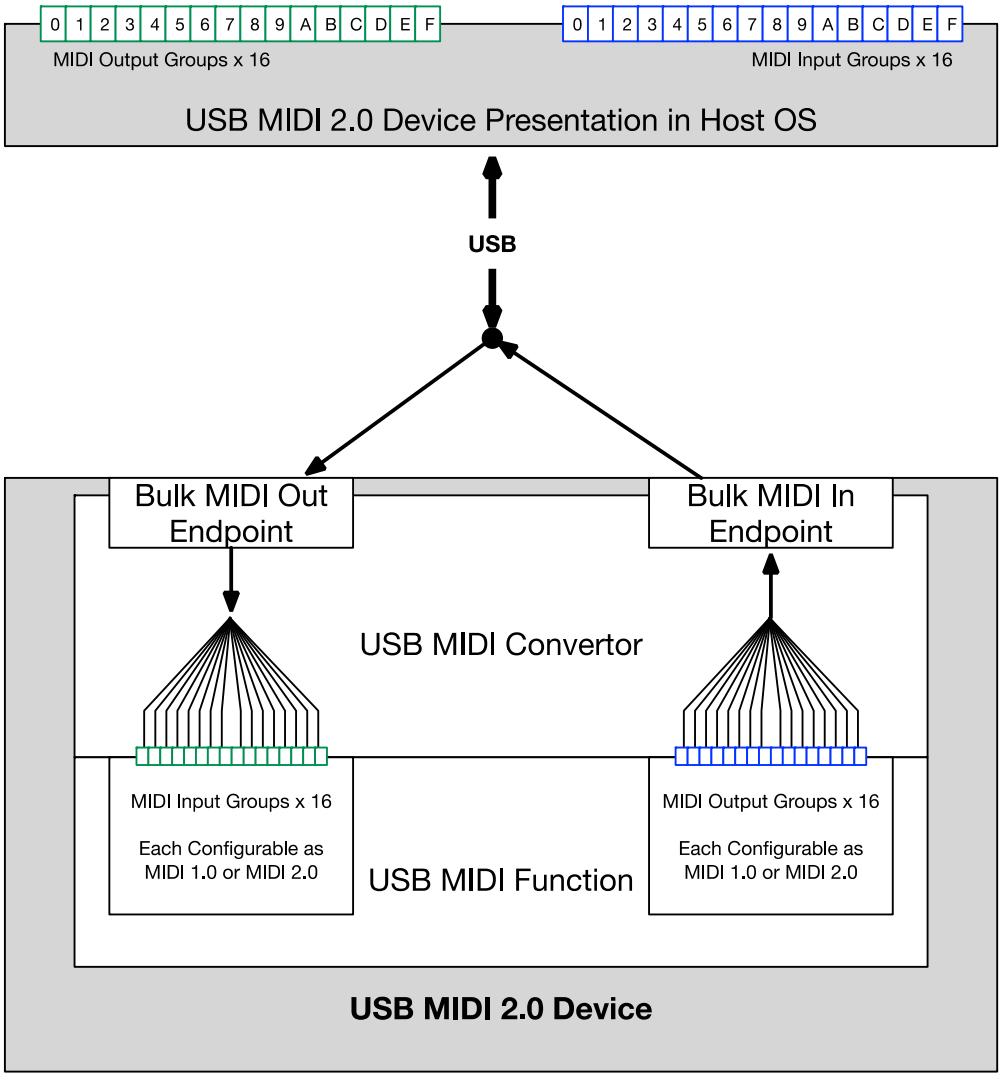


Figure 8: USB MIDI 2.0 Device – Interleaving up to 16 Groups on Each Endpoint

4.1 Communication from Host to USB MIDI Function

MIDI Data sent from applications in the Host to MIDI output connections in the Host is transported by USB to IN Group Terminals in the USB MIDI Function:

Multiple applications may want to send MIDI data streams to the USB MIDI function. The data format used for communication is the Universal MIDI Packet which supports up to 16 UMP Groups on the device. In order to send MIDI data streams to the USB MIDI function, Host applications connect to any IN Group Terminal exposed by the device via a Host MIDI output port. The USB MIDI Converter receives the incoming Universal MIDI Packets from the Host. It routes the original MIDI data streams to the proper IN Group Terminal, according to the Group field embedded in each Universal MIDI Packet. There is a one to one relationship between the value of the Group field in the Universal MIDI Packet and the IN Group Terminal. The IN Group Terminal is a member of a Group Terminal Block which is associated with a single MIDI OUT Endpoint. Once present at the IN Group Terminal, the USB MIDI Function routes and uses the data according to its internal design.

4.2 Communication from USB MIDI Function to Host

MIDI Data sent from functions in the USB MIDI Function to OUT Group Terminals in the USB MIDI Function is transported by USB to MIDI input connections in the Host:

Multiple entities in the USB MIDI function may want to send MIDI data streams to the Host. In general, multiple UMP Groups of MIDI data streams, originating from different components within the USB MIDI function according to the Group field embedded in each Universal MIDI Packet, arrive at the different OUT Group Terminals. There is a one to one relationship between the value of the Group field in the Universal MIDI Packet and the OUT Group Terminal. The OUT Group Terminal is a member of a Group Terminal Block which is associated with a single MIDI IN Endpoint. The USB MIDI Converter multiplexes all the MIDI data streams into a single MUX MIDI data stream. This MUX MIDI data stream is sent over a USB pipe to the Host. The Host software then inspects the incoming MUX-MIDI USB MIDI Data Packets, extracts the MIDI data from the packets and routes the original MIDI data streams to the proper applications via Host MIDI input ports, according to the value of the Group field in the Universal MIDI Packets.

5 Configuration Descriptors

5.1 Core Descriptors

A set of core USB descriptors describe the type of device and base level class of functionality. All devices shall include these mandatory descriptors:

1. Device Descriptor
    - As defined by USB Specification
2. Configuration Descriptor
    - As defined by USB Specification

If a Device contains an Alternate Setting which is compliant with USB Device Class Specification for MIDI Devices Version 1.0, then the Device shall also include these descriptors:

3. Standard AudioControl (AC) Interface Descriptor
    - As defined by USB Audio Class Specifications
    - For legacy purposes, MIDI is defined as a SubClass of Audio Class.
4. Class Specific AudioControl (AC) Interface Descriptor
    - As defined by USB Audio Class Specifications
    - Declares the MIDIStreaming (MS) Interface which belongs to this Audio Control Interface.

5.2 MIDIStreaming Interface Descriptors

5.2.1 Standard MS Interface Descriptor

The standard MIDIStreaming (MS) interface descriptor is identical to the standard interface descriptor defined in Section 9.6.3 of the *USB Specification*, except that some fields have now dedicated values.

Table 5-1: Standard MIDIStreaming Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 9
1	bDescriptorType	1	Constant	INTERFACE descriptor type
2	bInterfaceNumber	1	Number	Number of interface. A zero-based value identifying the index in the array of concurrent interfaces supported by this configuration.
3	bAlternateSetting	1	Number	Value used to select an alternate setting for the interface identified in the prior field.
4	bNumEndpoints	1	Number	Number of MIDI endpoints used by this interface (excluding endpoint 0).
5	bInterfaceClass	1	Class	AUDIO. Audio Interface Class code. Set to 0x01
6	bInterfaceSubClass	1	Subclass	MIDISTREAMING. Audio Interface Subclass code. Set to 0x03.

Offset	Field	Size	Value	Description
7	bInterfaceProtocol	1	Protocol	Not used. Must be set to 0x00.
8	iInterface	1	Index	Index of a string descriptor that describes this interface.

5.2.2 Class-Specific MS Interface Descriptor

The class-specific MS interface descriptor contains only* the Class-Specific MS Interface Header Descriptor.

> **In USB MIDI 1.0 this contains the concatenation of all the descriptors that are used to fully describe the interface.*

5.2.2.1 Class-Specific MS Interface Header Descriptor

The **bcdMSC** field identifies the release of the MIDIStreaming SubClass Specification with which this USB MIDI function and its descriptors are compliant. The **bDescriptorType** field identifies the descriptor as being a class-specific interface descriptor. The **bDescriptorSubtype** field further qualifies the exact nature of the descriptor. The **wTotalLength** field fills a critical function in USB MIDI 1.0 so the field is included for conformity although it has no function in USB MIDI 2.0.
The following table defines the class-specific MS interface header descriptor.

Table 5-2: Class-Specific MS Interface Header Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 7
1	bDescriptorType	1	Constant	CS_INTERFACE descriptor type.
2	bDescriptorSubtype	1	Constant	MS_HEADER descriptor subtype.
3	bcdMSC	2	BCD	MIDIStreaming SubClass Specification Release Number in Binary-Coded Decimal. Set to 0x0200.
5	wTotalLength	2	Number	Set to 7. This field is not used in this 2.0 version of USB MIDI class but is here for conformity with USB MIDI 1.0 class. In this version, set to match the value of the bLength field of this Class-Specific MS Interface Header. (See Section 5.2.2)

5.3 MIDI Streaming Endpoint Descriptors

The following paragraphs outline the descriptors that fully characterize the endpoint(s) used for transporting MIDI data streams to and from the USB MIDI function. Devices may use Bulk or Interrupt Endpoints for transporting MIDI data streams.

5.3.1 Standard MIDI Streaming Data Endpoint Descriptor

The Standard MIDI Streaming Data Endpoint descriptor is identical to the standard endpoint descriptor defined in Section 9.6.4, “Endpoint,” of the *USB Specification* and further expanded as defined in the *Universal Serial Bus Class Specification*. D7 of the **bEndpointAddress** field indicates whether the endpoint is a MIDI data source (D7 = 1) or a MIDI data sink (D7 = 0). The

bmAttributes Field bits are set to declare whether the endpoint type is bulk or interrupt. The synchronization type is indicated by D3..2 and must be set to None.

Table 5-3: Standard MS Data Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 7
1	bDescriptorType	1	Constant	ENDPOINT descriptor type
2	bEndpointAddress	1	Endpoint	The address of the endpoint on the USB device described by this descriptor. The address is encoded as follows: D7: Direction. 0 = OUT endpoint 1 = IN endpoint D6..4: Reserved, reset to zero D3..0: The endpoint number, determined by the designer.
3	bmAttributes	1	Bit Map	D7..4: Reserved D3..2: Synchronization type 00 = None D1..0: Transfer type 10 = Bulk D1..0: Transfer type 11 = Interrupt
4	wMaxPacketSize	2	Number	Maximum packet size this endpoint is capable of sending or receiving when this configuration is selected.
6	bInterval	1	Number	Interval for polling endpoint for Interrupt data transfers. For bulk endpoints this field is ignored and must be reset to 0.

Note: In the USB MIDI 1.0 specification, the Standard MIDI Streaming Data Endpoint Descriptor includes two additional fields, bRefresh, and bSynchAddress. These fields are not included in USB MIDI 2.0.

5.3.2 Class-Specific MIDI Streaming Data Endpoint Descriptor

Every MIDI Streaming Data Endpoint shall report data streaming connections to Group Terminals by declaring association to Group Terminal Blocks. The **bNumGrpTrmBlock** field contains the number of Group Terminal Blocks associated with this MIDI Streaming Data Endpoint.

The **baAssocGrpTrmBlkID()** array contains the IDs of the associated Group Terminal Blocks for this MIDI Streaming Endpoint. The number of Group Terminal Blocks within the USB MIDI function and the IDs of all Group Terminal Block are declared in the collection of all available Class-Specific MS Endpoint descriptors.

The number of Group Terminal members in all of the associated Group Terminal Blocks of one MIDI Streaming Endpoint shall be at least 1 and no more than 16 Group Terminals.

Table 5-4: Class-specific MS Data Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 4+n
1	bDescriptorType	1	Constant	CS_ENDPOINT
2	bDescriptorSubType	1	Constant	MS_GENERAL_2_0
3	bNumGrpTrmBlock	1	Number	Number of Group Terminal Blocks: n.
4	baAssoGrpTrmBlkID(1)	1	Number	ID of the first Group Terminal Block that is associated with this endpoint.
...	...	...	...	...
3 + n	baAssoGrpTrmBlkID(n)	1	Number	ID of the last Group Terminal Block that is associated with this endpoint.

5.4 Class-Specific Group Terminal Block Descriptors – Retrievable by a Separate Get Request

The following descriptors are retrieved by a Get Descriptors command (See Section 6).
The USB MIDI Function shall have one or more Group Terminal Blocks. Two types of Group Terminal Block descriptors represent and describe the Group Terminals of the USB MIDI Function.

1. A Group Terminal Blocks Header descriptor declares the total size of all Group Terminal Block descriptors.
2. Group Terminal Block descriptors declare which Group Terminals are members of that Block and describes shared functional characteristics of those member Group Terminals. The IDs of the Group Terminal Block are found in Class-Specific MS Data Endpoint Descriptors.

The first three fields are common to the Group Terminal Block Header Descriptors and all Group Terminal Block Descriptors. They contain the Descriptor Length, Descriptor Type, and Descriptor Subtype.

5.4.1 Class Specific Group Terminal Block Header Descriptor

The USB MIDI Function shall provide a Group Terminal Block Header descriptor which declares the total size of all Group Terminal Block descriptors.

Table 5-5: Class-Specific Group Terminal Header Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Number	Size of this descriptor, in bytes: 5
1	bDescriptorType	1	Constant	CS_GR_TRM_BLOCK
2	bDescriptorSubtype	1	Constant	GR_TRM_BLOCK_HEADER
3	wTotalLength	2	Number	Total number of bytes returned for the class-specific Group Terminal Block descriptors. Includes the combined length of this header descriptor and all Group Terminal Block descriptors.

5.4.2 Group Terminal Block Descriptor

The USB MIDI Function shall provide a Group Terminal Block Descriptor for each Group Terminal Block declared by the MIDI Streaming Data Endpoints. Each Group Terminal Block Descriptor describes the collection of Group Terminals which are members of that Group Terminal Block.
Each Group Terminal Block is assigned a unique identification number, the **bGrpTrmBlkID**. The value 0x00 is reserved for undefined ID, effectively restricting the total number of addressable Group Terminal Blocks in the USB MIDI function to 255
Group Terminal Block descriptors include the following class-specific information:

- **Group Terminal Block Type** (**bGrpTrmBlkType)** – declares whether all member Group Terminals are configured as:
    - 0x00 Bidirectional Connections used for MIDI-CI In/Out Pair. Every IN Group Terminal member has a matching OUT Group Terminal.

Note: If a bidirectional connection exists but does not function as a pair for the purposes of MIDI-CI, then the device shall use two separate Group Terminal Blocks, one Group Terminal Block for IN Group Terminals and a separate Group Terminal Block for OUT Group Terminals.

  - 0x01 IN Group Terminals Only
  - 0x02 OUT Group Terminals Only

- **First Group Terminal Number (nGroupTrm)** – Each Group Terminal Block declares a set of one or more Group Terminals as members of the Group Terminal Block. This value is the number of the Group Terminal, 0x0 to 0xF, with the lowest number which is a member of this Group Terminal Block.
  - 0x0 to 0xF = Group Terminal 1 to Group Terminal 16. Group Terminal 1 to Group Terminal 16 carry respectively Group 1 to Group 16 data.
- **Number of Group Terminal Numbers (nNumGroupTrm)** – The count of Group Terminal Numbers for members of the Group Terminal Block. The members of the Group Terminal Block are numbered from [nGroupTrm] to [nGroupTrm + (nNumGroupTrm - 1)].

  Note: The count of Group Terminals (vs. Group Terminal Numbers) in the Group Terminal Block equals this value when **bGrpTrmBlkType** is set to 0x01 IN Group Terminals Only or set to 0x02 OUT Group Terminals Only. When **bGrpTrmBlkType** is set to 0x00 Bidirectional Connections used for MIDI-CI In/Out Pair, then each In/Out pair is sharing a Group Terminal Number and the count of Group Terminals in the Group Terminal Block is (value x 2).

Examples using bGrpTrmBlkType, nGroupTrm and nNumGroupTrm

- **bGrpTrmBlkType = 0x01, nGroupTrm = 0x00, and nNumGroupTrm = 3:**
  There are three IN Group Terminal members, numbered 1, 2, and 3. Those Group Terminals receive Universal MIDI Packets with Group field values of 1, 2, and 3 respectively.
- **bGrpTrmBlkType = 0x02, nGroupTrm = 0x06, and nNumGroupTrm = 5:**
  There are five OUT Group Terminal Members are numbered 7, 8, 9, 10, and 11. Those Group Terminals send Universal MIDI Packets with Group field values of 7, 8, 9, 10, and 11 respectively.
- **bGrpTrmBlkType = 0x00, nGroupTrm = 0x00, and nNumGroupTrm = 3:**
  There are six Group Terminals. There are three IN Group Terminal members, numbered 1, 2, and 3. Those Group Terminals receive Universal MIDI Packets with Group field values of 1, 2, and 3 respectively. There are also three OUT Group Terminal members, numbered 1, 2, and 3. Those Group Terminals send Universal MIDI Packets with Group field values of 1, 2, and 3 respectively.

- **Default Protocol (bMIDIProtocol)** – The Group Terminals which are members of the Group Terminal Block may have a default MIDI Protocol. If a Group Terminal's default protocol is unknown, MIDI-CI Protocol Negotiation may be used to discover the current Protocol. MIDI-CI may also be used to set a different Protocol choice on that Group Terminal.

  - 0x00 = Unknown (Use MIDI-CI)
  - 0x01 = MIDI 1.0, Support UMP up to 64 bits in size. (default)
  - 0x02 = MIDI 1.0, Support UMP up to 64 bits in size. Uses Jitter Reduction Timestamps.
  - 0x03 = MIDI 1.0, Support UMP up to 128 bits in size.
  - 0x04 = MIDI 1.0, Support UMP up to 128 bits in size. Uses Jitter Reduction Timestamps.
  - 0x11 = MIDI 2.0
  - 0x12 = MIDI 2.0. Uses Jitter Reduction Timestamps.
- **Maximum Bandwidth Capability (wMaxInputBandwidth)** – This is bandwidth receiving capability of the MIDI components connected to the IN Group Terminals in this Group Terminal Block. Bandwidth is shared across all Group Terminals. This is independent of USB transport mechanisms that determine or influence bandwidth on the transport such as maximum packet size, service interval, NAK and retry, etc. This value usually reflects the parsing capabilities of the MIDI Function to help applications on the host optimize their use of MIDI data. Allowed values:
  - 0x0000 = Bandwidth capability is unknown or not a fixed value.
  - 0x0001 to 0xFFFF = number of 4KB/second
  - 0x0001 = Rounded representation of MIDI's classic baud rate of 31.25kb/s.
- **Maximum Bandwidth Capability (wMaxOutputBandwidth)** – This is bandwidth sending capability of the MIDI components connected to the OUT Group Terminals in this Group Terminal Block. Bandwidth is shared across all Group Terminals. This is independent of USB transport mechanisms that determine or influence bandwidth on the transport such as maximum packet size, service interval, NAK and retry, etc. This value reflects the generating capabilities of the MIDI Function to help applications on the host optimize their use of MIDI data. Allowed values:
  - 0x0000 = Bandwidth capability is unknown or not a fixed value.
  - 0x0001 to 0xFFFF = number of 4KB/second
  - 0x0001 = Rounded representation of MIDI's classic baud rate of 31.25kb/s.

5.4.2.1 Group Terminal Block Descriptor

Table 5-6: Group Terminal Block Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	Constant	Size of this descriptor, in bytes: 13
1	bDescriptorType	1	Constant	CS_GR_TRM_BLOCK
2	bDescriptorSubtype	1	Constant	GR_TRM_BLOCK
3	bGrpTrmBlkID	1	Constant	ID of this Group Terminal Block
4	bGrpTrmBlkType	1	Constant	Group Terminal Block Type: 0x00:bi-directional 0x01: IN Group Terminals only 0x02: OUT Group Terminals only

Offset	Field	Size	Value	Description
5	nGroupTrm	1	Number	The first member Group Terminal in this Block, with Number which is UMP Group field value for all Messages on that Group Terminal. 0x00 – 0x0F
6	nNumGroupTrm	1	Number	Number of member Group Terminals spanned. Group Terminals included have contiguously incrementing Numbers.
7	iBlockItem	1	Number	ID of STRING descriptor for UI representation of Block item
8	bMIDIProtocol	1	Constant	Default MIDI protocol, All Group Terminals. 0x00 = Unknown (Use MIDI-CI) 0x01 = MIDI 1.0, Support UMP up to 64 bits in size. 0x02 = MIDI 1.0, Support UMP up to 64 bits in size. Uses Jitter Reduction Timestamps. 0x03 = MIDI 1.0, Support UMP up to 128 bits in size. 0x04 = MIDI 1.0, Support UMP up to 128 bits in size. Uses Jitter Reduction Timestamps. 0x11 = MIDI 2.0 0x12 = MIDI 2.0. Uses Jitter Reduction Timestamps.
9	wMaxInputBandwidth	2	Number	Maximum Input Bandwidth Capability in 4KB/second. 0x0000 – 0xFFFF. Bandwidth is total for this Block, shared among all IN Group Terminals. 0x0000 = Unknown or Not Fixed. 0x0001 = Rounded version of 31.25kb/s
11	wMaxOutputBandwidth	2	Number	Maximum Output Bandwidth Capability in 4KB/second. 0x0000 – 0xFFFF. Bandwidth is total for this Block, shared among all OUT Group Terminals. 0x0000 = Unknown or Not Fixed. 0x0001 = Rounded version of 31.25kb/s

6 Class Specific Command: Group Terminal Blocks Descriptors Request

This request retrieves the complete set of all Group Terminal Block descriptors.
General application is as commonly defined by USB. A Host may send this request with wLength set to 0x05 to retrieve the Class-Specific Group Terminal Header Descriptor, which includes a wTotalLength value. Then the Host may use the same request with wLength set to the value of wTotalLength from the Class-Specific Group Terminal Header Descriptor to retrieve the whole set of all Group Terminal Block descriptors.

bmRequestType	0b10000001
bRequest	GET_DESCRIPTOR (0x06)
wValue	CS_GR_TRM_BLOCK in high byte (0x26), alternate setting number in low byte
wIndex	Interface Number
wLength	Length of data to retrieve.

Providing the corresponding Alternate Setting Number in the low byte allows for adjusted bandwidths in different alternate settings. (Bulk and Interrupt Endpoint Alternate Settings.)

Appendix A. Audio Device Class Codes: MIDIStreaming

A.1 MS Class-Specific Interface Descriptor Types

Descriptor Subtype	Value
CS_UNDEFINED	0x20
CS_DEVICE	0x21
CS_CONFIGURATION	0x22
CS_STRING	0x23
CS_INTERFACE	0x24
CS_ENDPOINT	0x25
CS_GR_TRM_BLOCK	0x26

A.1 MS Class-Specific Interface Descriptor Subtypes

Descriptor Subtype	Value
MS_DESCRIPTOR_UNDEFINED	0x00
MS_HEADER	0x01
MIDI_IN_JACK	0x02
MIDI_OUT_JACK	0x03
ELEMENT	0x04

A.2 MS Class-Specific Endpoint Descriptor Subtypes

Descriptor Subtype	Value
DESCRIPTOR_UNDEFINED	0x00
MS_GENERAL	0x01
MS_GENERAL_2_0	0x02

A.3 MS Class-Specific Group Terminal Block Descriptor Subtypes

Descriptor Subtype	Value
GR_TRM_BLOCK_UNDEFINED	0x00
GR_TRM_BLOCK_HEADER	0x01
GR_TRM_BLOCK	0x02

A.4 MS Interface Header MIDIStreaming Class Revision

MIDIStreaming Class Revision	Value
MS_MIDI_1_0	0x0100
MS_MIDI_2_0	0x0200

A.5 MS MIDI IN and OUT Jack types

MIDI IN and OUT Jack type	Value
JACK_TYPE_UNDEFINED	0x00
EMBEDDED	0x01
EXTERNAL	0x02

A.6 Group Terminal Block Type

Group Terminal Block Type	Value
BIDIRECTIONAL	0x00
INPUT_ONLY	0x01
OUTPUT_ONLY	0x02

A.7 Group Terminal Default MIDI Protocol

Default MIDI Protocol	Value
USE_MIDI_CI	0x00
MIDI_1_0_UP_TO_64_BITS	0x01
MIDI_1_0_UP_TO_64_BITS_AND_JRTS	0x02
MIDI_1_0_UP_TO_128_BITS	0x03
MIDI_1_0_UP_TO_128_BITS_AND_JRTS	0x04
MIDI_2_0	0x11
MIDI_2_0_AND_JRTS	0x12

A.8 Group Terminal Number (Universal MIDI Packet Group)

Group Terminal Number	Value
GROUP_1	0x00
GROUP_2	0x01
GROUP_3	0x02

Group Terminal Number	Value
GROUP_4	0x03
GROUP_5	0x04
GROUP_6	0x05
GROUP_7	0x06
GROUP_8	0x07
GROUP_9	0x08
GROUP_10	0x09
GROUP_11	0x0A
GROUP_12	0x0B
GROUP_13	0x0C
GROUP_14	0x0D
GROUP_15	0x0E
GROUP_16	0x0F

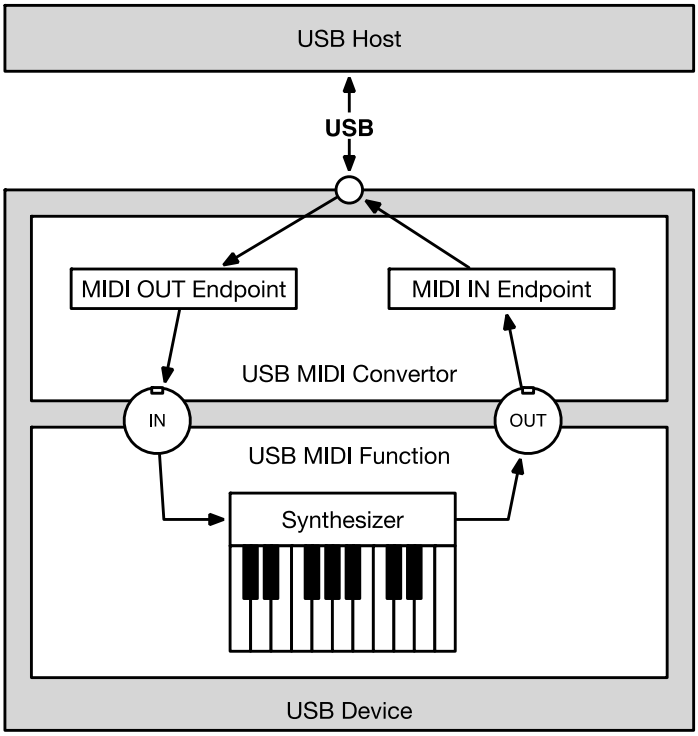
Appendix B. Example 1: Simple MIDI Instrument (Informative)

The following example shows the descriptors for a simple MIDI device, which with one set of descriptors, is designed to be connected to either:

1. A USB Host with a driver for USB MIDI 1.0, using Alternate Setting #0 on Interface #1

or

2. A USB Host with a driver for USB MIDI 2.0, using Alternate Setting #1 on Interface #1.



Descriptors for the USB MIDI 1.0 implementation on Alternate Setting 0:

1. Endpoint descriptors report connections to Embedded MIDI Jacks
2. Embedded MIDI Jack descriptors report topology connections to or from each Jack to the Synthesizer Element.
3. The Element descriptor reports the capabilities of the Synthesizer.

Descriptors for the USB MIDI 2.0 implementation on Alternate Setting 1:

1. Endpoint descriptors report associations to a Group Terminal Blocks which have the Group Terminals as members of the Group Terminal Blocks.
2. The Group Terminal Block descriptor reports association of the In and Out pair of Group Terminals. This association of Input and Output serves bidirectional MIDI-CI transactions. Capabilities of the Synthesizer may be discovered by the use of MIDI-CI messages exchanged in the MIDI streams to and from the associated Group Terminal.

B.1 Example 1 Device and Configuration Descriptor

B.1.1 Device Descriptor

Table B-1: Device Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x12	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x01	DEVICE descriptor.
2	bcdUSB	2	0x0110	1.10 - Revision of USB specification.
4	bDeviceClass	1	0x00	Device defined at Interface level.
5	bDeviceSubClass	1	0x00	Unused.
6	bDeviceProtocol	1	0x00	Unused.
7	bMaxPacketSize0	1	0x08	8 bytes.
8	idVendor	2	0xXXXX	Vendor ID.
10	idProduct	2	0xXXXX	Product ID.
12	bcdDevice	2	0xXXXX	Device Release Code.
14	iManufacturer	1	0x01	Index to string descriptor that contains the string <Your Name> in Unicode.
15	iProduct	1	0x02	Index to string descriptor that contains the string <Your Product Name> in Unicode.
16	iSerialNumber	1	0x03	Index to string descriptor that contains the device's serial number.
17	bNumConfigurations	1	0x01	One configuration.

B.1.2 Configuration Descriptor

Table B-2: Configuration Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x02	CONFIGURATION descriptor.
2	wTotalLength	2	0x00XX	Length of the total configuration block, including this descriptor, in bytes.
4	bNumInterfaces	1	0x02	Two interfaces. Audio Control Interface and MIDI Streaming Interface.
5	bConfigurationValue	1	0x01	ID of this configuration.
6	iConfiguration	1	0x00	Unused.
7	bmAttributes	1	0x80	Bus Powered device, not Self Powered, no Remote wakeup capability.
8	MaxPower	1	0x32	100 mA Max. power consumption.

B.2 Example 1 AudioControl Interface Descriptors

The AudioControl interface describes the device structure (audio function topology) and is used to manipulate the Audio Controls. This device has no audio function incorporated. However, the AudioControl interface is mandatory and therefore both the standard AC interface descriptor and the class-specific AC interface descriptor must be present. The class-specific AC interface descriptor only contains the header descriptor.

B.2.1 Standard AC Interface Descriptor

The AudioControl interface has no dedicated endpoints associated with it. It uses the default pipe (endpoint 0) for all communication purposes. Class-specific AudioControl Requests are sent using the default pipe. There is no Status Interrupt endpoint provided.

Table B-3: Standard AC Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor.
2	bInterfaceNumber	1	0x00	Index of this interface.
3	bAlternateSetting	1	0x00	Index of this setting.
4	bNumEndpoints	1	0x00	0 endpoints.
5	bInterfaceClass	1	0x01	AUDIO.
6	bInterfaceSubclass	1	0x01	AUDIO_CONTROL.
7	bInterfaceProtocol	1	0x00	Unused.
8	iInterface	1	0x00	Unused.

B.2.2 Class-specific AC Interface Descriptor

The Class-specific AC interface descriptor is always headed by a Header descriptor that contains general information about the AudioControl interface. It contains all the pointers needed to describe the Audio Interface Collection, associated with the described audio function. Only the Header descriptor is present in this device because it does not contain any audio functionality as such.

Table B-4: Class Specific AC Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE.
2	bDescriptorSubtype	1	0x01	HEADER subtype.
3	bcdADC	2	0x0100	Revision of Audio class specification - 1.0
5	wTotalLength	2	0x0009	Total size of class specific descriptors.
7	bInCollection	1	0x01	Number of streaming interfaces.
8	baInterfaceNr(1)	1	0x01	MIDIStreaming interface 1 belongs to this AudioControl interface.

B.3 Example 1 MIDIStreaming Interface Descriptors
(On Alternate Setting 0x00 for USB MIDI 1.0)

B.3.1 Standard MS Interface Descriptor

Table B-5: Standard MS Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor.
2	bInterfaceNumber	1	0x01	Index of this interface.
3	bAlternateSetting	1	0x00	Index of this alternate setting.
4	bNumEndpoints	1	0x02	2 endpoints.
5	bInterfaceClass	1	0x01	AUDIO.
6	bInterfaceSubclass	1	0x03	MIDISTREAMING.
7	bInterfaceProtocol	1	0x00	Unused.
8	iInterface	1	0x00	Unused.

B.3.2 Class-specific MS Interface Descriptor

Table B-6: Class-specific MS Interface Header Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE descriptor.
2	bDescriptorSubtype	1	0x01	MS_HEADER subtype.
3	bcdMSC	2	0x0100	Revision of this class specification.
5	wTotalLength	2	0x0041	Total size of class-specific descriptors.

B.3.3 MIDI IN Jack Descriptor

Table B-7: MIDI IN Jack Descriptor (Embedded)

Offset	Field	Size	Value	Description
0	bLength	1	0x06	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE descriptor.
2	bDescriptorSubtype	1	0x02	MIDI_IN_JACK subtype.
3	bJackType	1	0x01	EMBEDDED.
4	bJackID	1	0x01	ID of this Jack.
5	iJack	1	0x00	Unused.

B.3.4 MIDI IN Jack Descriptor

Table B-8: MIDI IN Jack Descriptor (External)

Offset	Field	Size	Value	Description
0	bLength	1	0x06	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE descriptor.
2	bDescriptorSubtype	1	0x02	MIDI_IN_JACK subtype.
3	bJackType	1	0x02	EXTERNAL.
4	bJackID	1	0x02	ID of this Jack.
5	iJack	1	0x00	Unused.

B.3.5 MIDI OUT Jack Descriptor

Table B-9: MIDI OUT Jack Descriptor (Embedded)

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE descriptor.
2	bDescriptorSubtype	1	0x03	MIDI_OUT_JACK subtype.
3	bJackType	1	0x01	EMBEDDED.
4	bJackID	1	0x03	ID of this Jack.
5	bNrInputPins	1	0x01	Number of Input Pins of this Jack.
6	BaSourceID(1)	1	0x02	ID of the Entity to which this Pin is connected.
7	BaSourcePin(1)	1	0x01	Output Pin number of the Entity to which this Input Pin is connected.
8	iJack	1	0x00	Unused.

B.3.6 MIDI OUT Jack Descriptor (External)

Table B-10: MIDI OUT Jack Descriptor (External)

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE descriptor.
2	bDescriptorSubtype	1	0x03	MIDI_OUT_JACK subtype.
3	bJackType	1	0x02	EXTERNAL.
4	bJackID	1	0x04	ID of this Jack.
5	bNrInputPins	1	0x01	Number of Input Pins of this Jack.
6	BaSourceID(1)	1	0x01	ID of the Entity to which this Pin is connected.
7	BaSourcePin(1)	1	0x01	Output Pin number of the Entity to which this Input Pin is connected.
8	iJack	1	0x00	Unused.

B.4 Example 1 Bulk OUT Endpoint Descriptors
(On Alternate Setting 0x00 for USB MIDI 1.0)

B.4.1 Standard Bulk OUT Endpoint Descriptor

Table B-11: Standard Bulk OUT Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x05	ENDPOINT descriptor.
2	bEndpointAddress	1	0x01	OUT Endpoint 1.
3	bmAttributes	1	0x02	Bulk, not shared.
4	wMaxPacketSize	2	0x0040	64 bytes per packet.
6	bInterval	1	0x00	Ignored for Bulk. Set to zero.
7	bRefresh	1	0x00	Unused.
8	bSynchAddress	1	0x00	Unused.

B.4.2 Class-specific MS Bulk OUT Endpoint Descriptor

Table B-12: Class Specific Bulk OUT Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x05	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x25	CS_ENDPOINT descriptor
2	bDescriptorSubtype	1	0x01	MS_GENERAL subtype.
3	bNumEmbMIDIJack	1	0x01	Number of embedded MIDI IN Jacks.
4	BaAssocJackID(1)	1	0x01	ID of the Embedded MIDI IN Jack.

B.5 Example 1 Bulk IN Endpoint Descriptors
(On Alternate Setting 0x00 for USB MIDI 1.0)

B.5.1 Standard Bulk IN Endpoint Descriptor

Table B-13: Standard Bulk IN Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x05	ENDPOINT descriptor.
2	bEndpointAddress	1	0x81	IN Endpoint 1.
3	bmAttributes	1	0x02	Bulk, not shared.
4	wMaxPacketSize	2	0x0040	64 bytes per packet.
6	bInterval	1	0x00	Ignored for Bulk. Set to zero.

Offset	Field	Size	Value	Description
7	bRefresh	1	0x00	Unused.
8	bSynchAddress	1	0x00	Unused.

B.5.2 Class-specific MS Bulk IN Endpoint Descriptor

Table B-14: Class Specific Bulk IN Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x05	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x25	CS_ENDPOINT descriptor
2	bDescriptorSubtype	1	0x01	MS_GENERAL subtype.
3	bNumEmbMIDIJack	1	0x01	Number of embedded MIDI OUT Jacks.
4	BaAssocJackID(1)	1	0x03	ID of the Embedded MIDI OUT Jack.

B.6 Example 1 MIDIStreaming Interface Descriptors
(On Alternate Setting 0x01 for USB MIDI 2.0)

B.6.1 Standard MS Interface Descriptor

Table B-15: Standard MS Interface Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x09	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x04	INTERFACE descriptor.
2	bInterfaceNumber	1	0x01	Index of this interface.
3	bAlternateSetting	1	0x01	Index of this alternate setting.
4	bNumEndpoints	1	0x02	2 endpoints.
5	bInterfaceClass	1	0x01	AUDIO.
6	bInterfaceSubclass	1	0x03	MIDISTREAMING.
7	bInterfaceProtocol	1	0x00	Unused.
8	iInterface	1	0x00	Unused.

B.6.2 Class-specific MS Interface Header Descriptor

Table B-16: Class-specific MS Interface Header Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x24	CS_INTERFACE descriptor.
2	bDescriptorSubtype	1	0x01	MS_HEADER subtype.

Offset	Field	Size	Value	Description
3	bcdMSC	2	0x0200	MS_MIDI_2_0 - Revision of this class specification.
5	wTotalLength	2	0x0007	Set to 7. This field is not used in this 2.0 version of USB MIDI class but is here for conformity with USB MIDI 1.0 class. In this version, set to match the value of the bLength field of this Class-Specific MS Interface Header.

B.7 Example 1 Bulk OUT Endpoint Descriptors
(On Alternate Setting 0x01 for USB MIDI 2.0)

B.7.1 Standard Bulk OUT Endpoint Descriptor

Table B-17: Standard Bulk OUT Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x05	ENDPOINT descriptor.
2	bEndpointAddress	1	0x01	OUT Endpoint 1.
3	bmAttributes	1	0x02	Bulk, not shared.
4	wMaxPacketSize	2	0x0040	64 bytes per packet.
6	bInterval	1	0x00	Ignored for Bulk. Set to zero.

B.7.2 Class-specific MS Bulk OUT Endpoint Descriptor

Table B-18: Class-specific Bulk OUT Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x05	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x25	CS_ENDPOINT descriptor
2	bDescriptorSubtype	1	0x02	MS_GENERAL_2_0 subtype.
3	bNumGrpTrmBlock	1	0x01	Number of Group Terminal Blocks.
4	baAssoGrpTrmBlkID(1)	1	0x01	ID of the Group Terminal Block that is associated with this endpoint.

B.8 Example 1 Interrupt IN Endpoint Descriptors
(On Alternate Setting 0x01 for USB MIDI 2.0)

B.8.1 Standard Interrupt IN Endpoint Descriptor

Table B-19: Standard Interrupt IN Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x07	Size of this descriptor, in bytes.

Offset	Field	Size	Value	Description
1	bDescriptorType	1	0x05	ENDPOINT descriptor.
2	bEndpointAddress	1	0x81	IN Endpoint 1.
3	bmAttributes	1	0x03	Interrupt.
4	wMaxPacketSize	2	0x0040	64 bytes per packet.
6	bInterval	1	0x01	Polling Interval: 1ms.*

*The polling interval 0x01 is equal to 1ms in this case because this sample device is a USB Fullspeed Device. The value and value range might have a different meaning in other device implementations which use other USB transfer speeds such as High Speed or Superspeed. See USB Specifications.

B.8.2 Class-specific MS Interrupt IN Endpoint Descriptor

Table B-20: Class Specific Interrupt IN Endpoint Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x05	Size of this descriptor, in bytes.
1	bDescriptorType	1	0x25	CS_ENDPOINT descriptor
2	bDescriptorSubtype	1	0x02	MS_GENERAL_2_0 subtype.
3	bNumGrpTrmBlock	1	0x01	Number of Group Terminal Blocks.
4	baAssoGrpTrmBlkID (1)	1	0x01	ID of the Group Terminal Block that is associated with this endpoint.

B.9 Example 1 Group Terminal Block Descriptors (On Alternate Setting 0x01 for USB MIDI 2.0)

When the Host sends a Get Group Terminal Blocks Descriptors Request, the device returns the number of requested bytes from the following set of descriptors:

B.9.1 Class Specific Group Terminal Block Header Descriptor

Table B-21: Class-Specific Group Terminal Block Header Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x05	Size of this descriptor, in bytes: 5
1	bDescriptorType	1	0x26	CS_GR_TRM_BLOCK descriptor type.
2	bDescriptorSubtype	1	0x01	GR_TRM_BLOCK_HEADER descriptor subtype.
3	wTotalLength	2	0x0012	Total number of bytes returned for the class-specific Group Terminal Block descriptors. Includes the combined length of this header descriptor and all Group Terminal Block descriptors.

B.9.2 Group Terminal Block Descriptor

Table B-22: Group Terminal Block Descriptor

Offset	Field	Size	Value	Description
0	bLength	1	0x0D	Size of this descriptor, in bytes: 13
1	bDescriptorType	1	0x26	CS_GR_TRM_BLOCK descriptor type.
2	bDescriptorSubtype	1	0x02	GR_TRM_BLOCK subtype.
3	bGrpTrmBlkID	1	0x01	ID of this Group Terminal Block
4	bGrpTrmBlkType	1	0x00	bi-directional
5	nGroupTrm	1	0x00	Group Terminal Number 1 (all data on the first terminal uses Group value 0x00 = 1)
6	nNumGroupTrm	1	0x01	Number of member Group Terminals spanned. = 1
7	iBlockItem	1	0x04	ID of STRING descriptor for UI representation of Block item in Unicode.
8	bMIDIProtocol	1	0x00	Unknown default protocol. Use MIDI-CI Protocol Negotiation.
9	wMaxInputBandwidth	2	0x0001	Maximum Input Bandwidth Capability in 4KB/second. 0x0001 = 32kb/s (Rounded version of 31.25kb/s)
11	wMaxOutputBandwidth	2	0x0000	Maximum Output Bandwidth Capability in 4KB/second. 0x0000 = Unknown or Not Fixed.

B.10 Example 1 Get Descriptor Requests

Following are the requests from the Host used to retrieve the descriptors in B.9.1 and B.9.2 above.

Table B-23: First Get Group Terminal Blocks Descriptors Request

Field	Size	Value	Description
bmRequestType	1	0x81	Request type. Direction=Device to Host, Type=Standard, Recipient=Interface
bRequest	1	0x06	GET _DESCRIPTOR request.
wValue	2	0x2601	High byte (0x26) = Descriptor Type: CS_GR_TRM_BLOCK Low byte (0x01) = Number of alternate setting.
wIndex	2	0x0001	Number of interface.
wLength	2	0x0005	Requested descriptor length. (0x05 = get header only)*

The first request discovers that the total size of all Group Terminal Block descriptors in this example device = 0x0012 (see Section B.9.1). This value is used in a second request below to retrieve 18 bytes of all Group Terminal Block descriptors.

Table B-24: Second Get Group Terminal Blocks Descriptors Request

Field	Size	Value	Description
bmRequestType	1	0x81	Request type. Direction=Device to Host, Type=Standard, Recipient=Interface
bRequest	1	0x06	GET_DESCRIPTOR request.
wValue	2	0x2601	High byte (0x26) = Descriptor Type: CS_GR_TRM_BLOCK Low byte (0x01) = Number of alternate setting.
wIndex	2	0x0001	Number of interface.
wLength	2	0x0012	Requested descriptor length.

B.11 Example 1 String Descriptors

When the Host requests a String descriptor type with an ID for a String descriptor, the example device returns the descriptor for that ID from among the following String descriptors:

Table B-25: String Descriptor ID 0

Offset	Field	Size	Value	Description
0	bLength	1	0x04	Size of this descriptor, in bytes: 4
1	bDescriptorType	1	0x03	String Descriptor
2	wLANGID	2	0x0409	Supported Language Code (0x0409 = English - United States)

Table B-26: String Descriptor ID 1

Offset	Field	Size	Value	Description
0	bLength	1	0x24	Size of this descriptor, in bytes: 36
1	bDescriptorType	1	0x03	String Descriptor
2	bString	34	"Manufacturer Name"	in Unicode

Table B-27: String Descriptor ID 2

Offset	Field	Size	Value	Description
0	bLength	1	0x1A	Size of this descriptor, in bytes: 26
1	bDescriptorType	1	0x26	String Descriptor.
2	bString	24	"Product Name"	in Unicode

Table B-28: String Descriptor ID 3

Offset	Field	Size	Value	Description
0	bLength	1	0x1C	Size of this descriptor, in bytes: 28
1	bDescriptorType	1	0x03	String Descriptor
2	bString	26	“Serial Number”	in Unicode

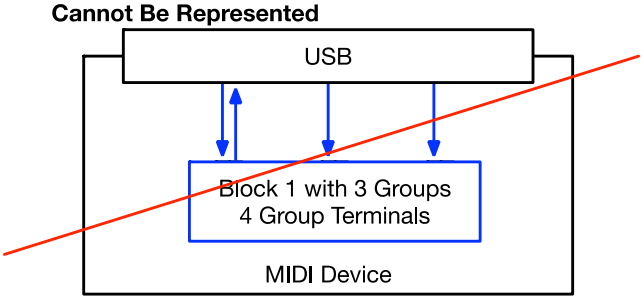
Table B-29: String Descriptor ID 4

Offset	Field	Size	Value	Description
0	bLength	1	0x18	Size of this descriptor, in bytes: 24
1	bDescriptorType	1	0x03	String Descriptor
2	bString	22	"Synthesizer"	Describes the Function associated with this Group Terminal Block, in Unicode

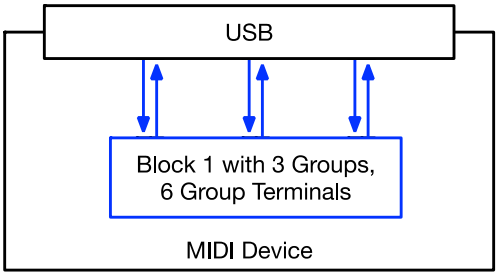
Appendix C. MIDI-CI Topology Limitation (Informative)

All Group Terminals in a Block must have either **NO** MIDI-CI In/Out pair or **ALL** MIDI-CI In/Out pairs for all member UMP Groups.

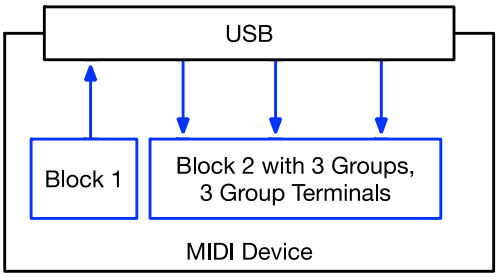
Cannot Be Represented



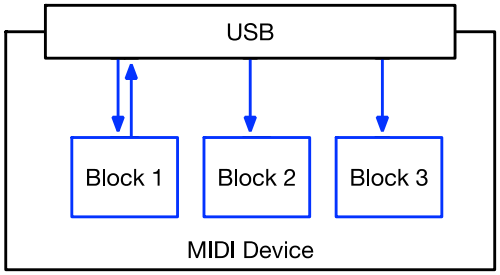
Option A: Preferred Design



Option B: Acceptable design. Does Not Show MIDI-CI Pair But MIDI-CI Discovery can find the MIDI-CI Pair



Option C: Possible but Does Not Show Relationship of 3 Group Inputs. MIDI does not provide a fix.



Prohibited: Group Terminal cannot be a member of more than 1 Block