Common System Exclusive Commands Manual

© iConnectivity 2015

© iConnectivity 2015

1. Overview

Header:

0xF0	- start of system exclusive
0x00, 0x01, 0x73	- iConnectivity's manufacturer ID code
0x7E	- message class

Body:

Device ID	- 7 bytes (2 bytes for product ID, 5 bytes for serial number)
Transaction ID	- 2 bytes (0 - 16383)
Command	- 2 bytes
Data Length	- 2 bytes, number of bytes in Command Data (0 - 16383)
Command Data	- length varies depending on command
Checksum	- 1 byte

Footer:

0xF7	- end of system exclusive

Data Format

All fields are big-endian (most significant byte occurs first, least significant byte occurs last). All fields which accept values greater than 0x7F must have the value split across multiple bytes in order to be compatible with MIDI sysex format. The least significant 7 bits of the value are contained in the 7 least significant bits of the last byte. The next 7 bits are contained in the 7 least significant bits of the last-1 byte, etc. Here are some examples:

Value	First Byte	Second Byte
0x0003	0x00	0x03
0x007F	0x00	0x7F
0x0080	0x01	0x00
0x0081	0x01	0x01
0x1234	0x24	0x34
0x2CA5	0x59	0x25

Device Identifier

The first 2 bytes of Device ID are product ID (PID) a 14 bit value split across 2 bytes. The remaining 5 bytes are serial number (SNUM) a 32 bit value split across 5 bytes. PID and SNUM must be formatted so that each byte is 0x7F or less as explained in the Data Format section. Here are some examples:

PID	SNUM	Device Identifier
0x0001	0x00000001	0x00 0x01 0x00 0x00 0x00 0x00 0x01
0x0001	0x00000080	0x00 0x01 0x00 0x00 0x00 0x01 0x00
0x00A0	0x00000080	0x01 0x02 0x00 0x00 0x00 0x01 0x00
0x0ABC	0x12345678	0x15 0x3C 0x01 0x11 0x51 0x2C 0x78

Some commands allow PID and SNUM to be zero which indicates a wildcard as follows:

PID	SNUM	Meaning
= 0	= 0	All devices.
= 0	> 0	All devices with the specified serial number.
> 0	= 0	All devices with the specified product ID.
> 0	> 0	A device that matches the specified product ID and serial number.

Transaction ID

Transaction ID can be used to uniquely identify messages. Typically, the host sends a query message to one or more devices, each device then sends a response message back to the host. A device will return the same transaction ID in the response as was sent in the query. It is not required that transaction IDs increment for every message sent by a host, they can always be 0x0000 or some other value, it is up to the host whether or not it wants to use transaction ID for managing messages. Transaction ID is a 14 bit value split across 2 bytes. Each byte is 0x7F or less as explained in the Data Format section. Range is 0 - 16383.

Command

A 14 bit field split across 2 bytes. The least significant 10 bits are the command ID. The most significant 4 bits are command flags:

Bit	Description
13	0 = Answer/Read, 1 = Query/Write
12-10	reserved (always zero)
9-0	command ID

© iConnectivity 2015

Data Length

The number of bytes that follow in the command data section. Data length is a 14 bit value split across 2 bytes. Each byte is 0x7F or less as explained in the Data Format section. Range is 0 - 16383.

Command Data

The content and length of the this field depends on the command ID. Some commands may have no command data.

Checksum

This field contains the 2s complement of the sum of all bytes in the body, excluding the checksum byte. In other words, summing all the bytes in the body should result in 0x00. Note that the checksum byte must be 0x7F or less to be compatible with MIDI sysex format (i.e. if the calculated checksum value is 0x83 then the checksum value used in the sysex message should be 0x03).

2. Device Commands

The following commands are defined for protocol version = 1.

2.1. GetDevice (Command ID = 0x01)

This command is used to discover devices. The sender can specify a specific product ID (PID), a specific serial number (SNUM), both PID & SNUM, or neither (using wildcards). The sender should wait several seconds for this message to propagate over a network if using any wildcards and be prepared to receive multiple responses. Every device that matches the PID or SNUM will respond with a RetDevice message. This command should always be the first command sent to a device because the response contains useful information regarding maximum packet size that can be used for other commands.

Command Data

Command flags are Query.
Command ID is 0x01.
Data length is 0.

Example: discover all devices, any product ID, any serial number

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x00                      // product ID (all)
0x00, 0x00, 0x00, 0x00, 0x00    // serial number (all)
0x00, 0x00                      // transaction ID
0x40, 0x01                      // command flags and ID
0x00, 0x00                      // data length
0x3F                            // checksum
0xF7                            // footer
```

Example: discover all devices that are a specific product ID (3), any serial number

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
```

```
0x00, 0x03                  // product ID (3)
0x00, 0x00, 0x00, 0x00, 0x00  // serial number (all)
0x00, 0x00                  // transaction ID
0x40, 0x01                  // command flags and ID
0x00, 0x00                  // data length
0x3C                        // checksum
0xF7                        // footer
```

2.2. RetDevice (Command ID = 0x02)

This command is sent by devices in response to a GetDevice command. It contains basic information that a host will need to use to communicate with this device (e.g. protocol version number). A host should cache this information and use it for all further communication with this device.

Command Data

Command flags are Answer.
Command ID is 0x02.
Data length depends on protocol version. For protocol version = 1, data length is always 4.

Byte #1: protocol version number that this device supports (1 - 127). If the host does not understand the protocol version number then it should not attempt any further communication with the device.

For protocol version number = 1:

Byte #2: device's current operating mode:

Value	Description
1	application mode
2	boot loader mode
3	test mode

Bytes #3-4: maximum length allowed for command data field (0 - 16383). If the host needs to send a message to the device that is longer than the allowed maximum length the host will need to split the command into multiple messages.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x03                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x00, 0x02                    // command flags and ID
0x00, 0x04                    // data length
0x01                          // protocol version number (1)
0x01                          // application mode (1)
0x02, 0x00                    // maximum length of command data field (256)
```

© iConnectivity 2015

```
0xxx                        // checksum
0xF7                        // footer
```

2.3. GetCommandList (Command ID = 0x03)

This command is used to query a device about which command IDs it supports. Devices that support this command will respond with a RetDevice message.

Command Data

Command flags are Query.
Command ID is 0x03.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x03                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

2.4. RetCommandList (Command ID = 0x04)

This command is sent by devices in response to a GetCommandList command. It contains a list of the command IDs that this device supports (command IDs that a host can send to this device to get info or change settings, not command IDs that this device will return to a host). Note that command IDs 0x01 and 0x03 are never returned (it is assumed that all devices support these command IDs).

Command Data

Command flags are Answer.
Command ID is 0x04.
Data length is 2 x the number of command IDs returned.

Bytes #1-2: command ID #1.
Bytes #3-4: command ID #2.
etc.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x04                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x05                      // command ID #1 (GetInfoList)
0xxx                            // checksum
0xF7                            // footer
```

© iConnectivity 2015

2.5. GetInfoList (Command ID = 0x05)

This command is used to query a device about which info IDs it supports. Devices that support this command will respond with a RetInfoList message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x05.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x05                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

2.6. RetInfoList (Command ID = 0x06)

This command is sent by devices in response to a GetInfoList command. It contains a list of the info IDs that this device supports.

Command Data

Command flags are Answer.
Command ID is 0x06.
Data length is 2 x the number of info IDs returned.

Byte #1: info ID #1.
Byte #2: maximum length of info ID #1 (0 if read only).
Byte #3: info ID #2.
Byte #4: maximum length of info ID #2 (0 if read only).
etc.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x06                      // command flags and ID
0x00, 0x06                      // data length
0x01, 0x00                      // info field ID #1 (accessory name, read only)
0x05, 0x00                      // info field ID #2 (firmware version, read only)
0x10, 0x1F                      // info field ID #3 (device name, 31 characters max)
0xxx                            // checksum
0xF7                            // footer
```

© iConnectivity 2015

2.7. GetInfo (Command ID = 0x07)

This command is used to query a device about a single info ID. Devices that support this command will respond with a RetInfo message. If this command or the info ID is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x07.
Data length is 1.

Example: get firmware version

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x07                      // command flags and ID
0x00, 0x01                      // data length
0x05                            // info ID for firmware version
0xxx                            // checksum
0xF7                            // footer
```

2.8. RetInfo / SetInfo (Command ID = 0x08)

RetInfo is sent by devices in response to a GetInfo command. It returns the ASCII string for a specific info ID.

SetInfo is used to set the ASCII string for a specific info ID. Devices that support this command will respond with an ACK message. Not all info IDs can be set, many are read only.

Command Data

Command flags are Answer for RetInfo, Write for SetInfo.
Command ID is 0x08.
Data length is 1 + the length of the info string.

Byte #1: info ID

Info ID	Description
0x01	accessory name (read only)
0x02	manufacturer name (read only)
0x03	model number (read only)
0x04	serial number (read only)
0x05	firmware version (read only)

Info ID	Description
0x06	hardware version (read only)
0x10	device name (read/write)

Bytes #2 - #N: 7-bit ASCII string, not NULL terminated (length varies)

Example: return firmware version

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x08                      // command flags and ID
0x00, 0x06                      // data length
0x05                            // info ID (firmware version)
0x31, 0x2E, 0x30, 0x2E, 0x37    // info value (the ascii string “1.0.7”)
0xxx                            // checksum
0xF7                            // footer
```

Example: set device name to “MIDI1”

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x08                      // command flags and ID
0x00, 0x06                      // data length
0x10                            // info ID (device name)
0x4D, 0x49, 0x44, 0x49, 0x31    // info value (the ascii string “MIDI1”)
0xxx                            // checksum
0xF7                            // footer
```

2.9. GetResetList (Command ID = 0x09)

This command is used to query a device about which reset IDs it supports. Devices that support this command will respond with a RetResetList message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x09.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
```

```
0x00, 0x00          // transaction ID
0x40, 0x09          // command flags and ID
0x00, 0x00          // data length
0xxx                // checksum
0xF7                // footer
```

2.10. RetResetList (Command ID = 0x0A)

This command is sent by devices in response to a GetResetList command. It contains a list of the reset IDs that this device supports.

Command Data

Command flags are Answer.
Command ID is 0x0A.
Data length is the number of reset IDs returned.

Byte #1: reset ID #1.
Byte #2: reset ID #2.
etc.

Reset ID	Description
0x01	restart into application mode
0x02	restart into boot loader mode

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x0A                      // command flags and ID
0x00, 0x02                      // data length
0x01                            // reset ID #1 (restart into application mode)
0x02                            // reset ID #2 (restart into boot loader mode)
0xxx                            // checksum
0xF7                            // footer
```

2.11. GetSaveRestoreList (Command ID = 0x0B)

This command is used to query a device about which save/restore IDs it supports. Devices that support this command will respond with a RetSaveRestoreList message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.

Command ID is 0x0B.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x0B                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

2.12. RetSaveRestoreList (Command ID = 0x0C)

This command is sent by devices in response to a GetSaveRestoreList command. It contains a list of the save/restore IDs that this device supports.

Command Data

Command flags are Answer.
Command ID is 0x0C.
Data length is the number of save/restore IDs returned.

Byte #1: save/restore ID #1.
Byte #2: save/restore ID #2.
etc.

Save/Restore ID	Description
0x01	save current configuration to FLASH
0x02	restore current configuration from FLASH
0x03	restore current configuration to factory default

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x0C                      // command flags and ID
0x00, 0x03                      // data length
0x01                            // reset ID #1 (save current configuration to FLASH)
0x02                            // reset ID #2 (restore current configuration from FLASH)
0x03                            // reset ID #3 (restore current configuration to factory default)
0xxx                            // checksum
0xF7                            // footer
```

2.13. GetEthernetPortInfo (Command ID = 0x0D)

This command is used to query a device about a specific ethernet port/jack. Devices that support this command will respond with a RetEthernetPortInfo message. If this command is not supported the device will respond with an ACK message (with error code). The host should issue a GetMIDIInfo message to the device to discover the number of ethernet ports/jacks before issuing this command.

Command Data

Command flags are Query.
Command ID is 0x0D.
Data length is 2.

Bytes #1-2: ethernet port/jack ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x0D                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // ethernet port/jack ID
0xxx                            // checksum
0xF7                            // footer
```

2.14. RetEthernetPortInfo / SetEthernetPortInfo (Command ID = 0x0E)

RetEthernetPortInfo is sent by devices in response to a GetEthernetPortInfo command.

SetEthernetPortInfo is sent by a host to a device to set the values of some of the port/jack parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. The host only needs to send up to (and including) byte #19, all bytes beyond #19 are read-only and are ignored.

Command Data

Command flags are Answer for RetEthernetPortInfo, Write for SetEthernetPortInfo.
Command ID is 0x0E.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: ethernet port/jack ID (1 - N).

Byte #4 [W]: IPMode, 0 = use static IP (as defined in bytes 5 - 19), 1 = use dynamic IP (DHCP or AutoIP if DHCP server is not available).

Bytes #5 - 19 [W]: Device IP address, subnet mask, and gateway address to use when IPMode = static. All are four-byte big-endian values encoded in five-bytes (similar to device ID) so that the most significant bit of each byte is 0.

Byte	Description
5 - 9 [W]	The IP address to use when IPMode = static
10 - 14 [W]	The subnet mask to use when IPMode = static
15 - 19 [W]	The address of the gateway to use when IPMode = static

Bytes #20 - 34: Current (active) device IP address, subnet mask, and gateway address. All are four-byte big-endian values encoded in five-bytes (similar to device ID) so that the most significant bit of each byte is 0. These values are only valid if the ethernet port is connected to a network, is active, and (if IPMode = dynamic) has obtained values from a DHCP server or AutoIP has finished negotiation.

Byte	Description
20 - 24	Current device IP address
25 - 29	Current subnet mask
30 - 34	Current gateway address

Bytes #35 - 46: Ethernet MAC address for this port, 48-bit value encoded as a 12 character, 7-bit ASCII string, not NULL terminated.
Byte #47: Length of device name (Bonjour name) field that follows (0 - N).
Bytes #48-N: Device name (Bonjour name), 7-bit ASCII string, not NULL terminated. Device name is only valid if the ethernet port is connected to a network, is active, and has finished Bonjour name resolution.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x0E                      // command flags and ID
0x00, 0x31                      // data length (51)
0x01                            // command version number (1)
0x00, 0x01                      // ethernet port/jack ID
0x01                            // IPMode: dynamic
0x0C, 0x05, 0x20, 0x02, 0x64    // static IP address (192.168.1.100)
0x0F, 0x7F, 0x7F, 0x7E, 0x00    // static subnet mask (255.255.255.0)
0x0C, 0x05, 0x20, 0x02, 0x01    // static gateway address (192.168.1.1)
0x0A, 0x4F, 0x78, 0x00, 0x08    // current IP address (169.254.0.8)
0x0F, 0x7F, 0x7C, 0x00, 0x00    // current subnet mask (255.255.0.0)
0x0A, 0x4F, 0x78, 0x00, 0x01    // current gateway address (169.254.0.1)
```

```
0x41, 0x43, 0x37, 0x41,     // ethernet MAC address as ASCII string “AC7A42010203”
0x34, 0x32, 0x30, 0x31,
0x30, 0x32, 0x30, 0x32
0x04                        // length of device name (Bonjour name) field that follows
0x69, 0x43, 0x4D, 0x34      // the ASCII string “iCM4”
0xxx                        // checksum
0xF7                        // footer
```

2.15. ACK (Command ID = 0x0F)

This command is sent by devices in response to various commands (acknowledgement that a command succeeded or failed).

Command Data

Command flags are Answer.
Command ID is 0x0F.
Data length is 3.

Bytes #1-2: command that generated this ACK

Byte #3: error code

Error Code	Description
0x00	no error
0x01	unknown command
0x02	malformed message
0x03	command failed

Example: no error response to SetInfo command

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x0F                      // command flags and ID
0x00, 0x03                      // data length
0x40, 0x08                      // command that generated this ACK (SetInfo)
0x00                            // error code (no error)
0xxx                            // checksum
0xF7                            // footer
```

2.16. Reset (Command ID = 0x10)

This command is used to reset a device. Devices that support this command will respond with an ACK message and then reset itself after a short delay (usually around 1 second). If this command or the reset ID is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Write.
Command ID is 0x10.
Data length is 1.

Example: reset device so that it restarts into boot loader mode

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x10                      // command flags and ID
0x00, 0x01                      // data length
0x02                            // reset ID for restart into boot loader mode
0xxx                            // checksum
0xF7                            // footer
```

2.17. SaveRestore (Command ID = 0x11)

This command is used to save or restore the current device configuration. Devices that support this command will respond with an ACK message. If this command or the save/restore ID is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Write.
Command ID is 0x11.
Data length is 1.

Example: restore current configuration to factory default

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x11                      // command flags and ID
0x00, 0x01                      // data length
0x03                            // save/restore ID for restore current configuration to factory default
0xxx                            // checksum
0xF7                            // footer
```

2.18. GetGizmoCount (Command ID = 0x12)

This command is used to query a device about the number of other devices (gizmos) that are connected to it. Devices should previously have been sent a GetDevice command before issuing this command. Devices that support this command will respond with a RetGizmoCount message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x12.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x12                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

2.19. RetGizmoCount (Command ID = 0x13)

This command is sent by devices in response to a GetGizmoCount command. It contains the number of devices (gizmos) that are connected to this device.

Command Data

Command flags are Answer.
Command ID is 0x13.
Data length is always 2.

Bytes #1-2: gizmo count

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x13                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x04                      // gizmo count (4)
0xxx                            // checksum
0xF7                            // footer
```

2.20. GetGizmoInfo (Command ID = 0x14)

This command is used to query a device about a specific gizmo. Devices should previously have been sent a GetDevice and GetGizmoCount command before issuing this command. Devices that support this command will respond with a RetGizmoInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x14.

Data length is 2.

Bytes #1-2: gizmo ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x14                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // gizmo ID
0xxx                            // checksum
0xF7                            // footer
```

2.21. RetGizmoInfo (Command ID = 0x15)

This command is sent by devices in response to a GetGizmoInfo command. It contains information for a specific gizmo.

Command Data

Command flags are Answer.
Command ID is 0x15.
Data length depends on command version number. For version = 1, data length is always 13.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte	Description
2 - 3	gizmo ID (1 - N)
4	gizmo type code: 1 = Query / Write (source), 2 = Answer / Read (destination)
5 - 6	port ID as used in MIDI commands (1 - N)
7 - 13	iConnectivity device identifier, all zeros for non-iConnectivity gizmos

The gizmo type code can be used to determine if a gizmo is upstream (source) or downstream (destination) of the device that is sent the GetGizmoInfo command. Sources generate queries, destinations provide answers.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x03                      // product ID
```

```
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x15                      // command flags and ID
0x00, 0x0D                      // data length
0x01                            // command version number (1)
0x00, 0x01                      // gizmo ID (1)
0x01                            // gizmo type (destination)
0x00, 0x02                      // port ID (2)
0x00, 0x05                      // product ID of gizmo
0x05, 0x04, 0x03, 0x02, 0x01    // serial number of gizmo
0xxx                            // checksum
0xF7                            // footer
```

3. MIDI Commands

The following commands are defined for protocol version = 1.

3.1. GetMIDIInfo (Command ID = 0x20)

This command is used to query a device about MIDI parameters. Devices that support this command will respond with a RetMIDIInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x20.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x20                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

3.2. RetMIDIInfo / SetMIDIInfo (Command ID = 0x21)

RetMIDIInfo is sent by devices in response to a GetMIDIInfo command. It contains basic information that a host will need to use to communicate with this device for all other MIDI related messages. A host should cache this information and use it for all further communication with this device.

SetMIDIInfo is sent by a host to a device to set the values of some of the parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMIDIInfo, Write for SetMIDIInfo.
Command ID is 0x21.
Data length depends on command version number. For version = 1, data length is always 15.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: number of MIDI ports supported by this device (0 - N).
Bytes #4-5: MIDI port number that the host is using to communicate to this device (0 - N).
Byte# 6: number of DIN jack pairs (in/out) supported by this device (0 - N).
Byte# 7: number of USB device jacks supported by this device (0 - N).
Byte# 8: number of USB host jacks supported by this device (0 - N).
Byte# 9: number of ethernet jacks supported by this device (0 - N).
Byte# 10: number of USB-MIDI ports supported by each USB device jack (0 - 16).
Byte# 11: number of USB-MIDI ports supported by each USB host jack (0 - N).
Byte# 12: number of RTP-MIDI sessions supported by each ethernet jack (0 - N).
Byte# 13: number of RTP-MIDI connections supported by each RTP-MIDI session (0 - N).

Byte #14 [W]: global MIDI flags:

Bit	Description
7 - 2	reserved (always zero)
1	set if routing should be enabled between ports on multi-port USB devices (only applicable if device has USB host jacks)
0	set if running status should be enabled on DIN outputs (only applicable if device has DIN jacks)

Byte #15 [W]: maximum number of ports to use on multi-port USB devices connected to a USB host jack (only applicable if device has USB host jacks). Minimum value is 1, maximum value is the number of USB-MIDI ports supported by each USB host jack (the value in byte #11).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x21                      // command flags and ID
0x00, 0x0F                      // data length (15)
0x01                            // command version number (1)
0x00, 0x14                      // number of MIDI ports (20)
0x00, 0x01                      // host is communicating on MIDI port #1
0x02                            // device has 2 sets of DIN jacks
0x02                            // device has 2 USB device jacks
```

```
0x01                    // device has 1 USB host jack
0x01                    // device has 1 ethernet jack
0x04                    // device supports 4 MIDI ports on each USB device jack
0x08                    // device supports 8 MIDI ports on each USB host jack
0x04                    // device supports 4 RTP-MIDI sessions on each ethernet jack
0x01                    // device supports 1 RTP-MIDI connection on each RTP-MIDI session
0x01                    // routing between multi-port is disabled, running status is enabled
0x04                    // use 4 ports maximum on multi-port USB devices
0xxx                    // checksum
0xF7                    // footer
```

3.3. GetMIDIPortInfo (Command ID = 0x22)

This command is used to query a device about a specific MIDI port. Devices that support this command will respond with a RetMIDIPortInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x22.
Data length is 2.

Bytes #1-2: MIDI port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x22                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // port ID
0xxx                            // checksum
0xF7                            // footer
```

3.4. RetMIDIPortInfo / SetMIDIPortInfo (Command ID = 0x23)

RetMIDIPortInfo is sent by devices in response to a GetMIDIPortInfo command.

SetMIDIPortInfo is sent by a host to a device to set the values of some of the port parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMIDIPortInfo, Write for SetMIDIPortInfo.
Command ID is 0x23.
Data length depends on command version number. For version = 1, minimum length is 10, actual length depends on the port name (a 7-bit ASCII string, not NULL terminated). Use the data length field to determine the string length.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: port ID (1 - N).

Byte #4: port type:

Value	Description
1	DIN
2	USB device
3	USB host
4	ethernet

Bytes #5-8: port info, depends on port type (byte #4):

Type	Description
DIN	Byte #5: DIN jack # (1 - N) Bytes #6-8: reserved (always 0)
USB device	Byte #5: USB device jack # (1 - N) Byte #6: device port # (1 - 16) Bytes #7-8: reserved (always 0)
USB host	Byte #5: USB host jack # (1 - N) Byte #6: USB host jack port # (1 - N) Byte #7-8: reserved (always 0)
ethernet	Byte #5: ethernet jack # (1 - N) Byte #6: session # (0 - N) Byte #7-8: reserved (always 0)

Byte #9: maximum length allowed for port name, 0 if read-only (0 - 127).

Byte #10 [W]: port MIDI flags:

© iConnectivity 2015

Bit	Description
7 - 2	reserved (always zero)
1	set if port output is enabled
0	set if port input is enabled

Bytes #11-N [W]: port name, 7-bit ASCII string, not NULL terminated.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x23                      // command flags and ID
0x00, 0x0E                      // data length (14)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x01                            // port type = DIN
0x01                            // port is first set of DIN jacks
0x00, 0x00, 0x00                // reserved for DIN port
0x0F                            // maximum length allowed for port name (15 ASCII characters)
0x03                            // input and output are both enabled
0x44, 0x49, 0x4E, 0x31          // port name, the ASCII string “DIN1”
0xxx                            // checksum
0xF7                            // footer
```

3.5. GetMIDIPortFilter (Command ID = 0x24)

This command is used to query a device about a specific MIDI port’s filters. Devices that support this command will respond with a RetMIDIPortFilter message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x24.
Data length is 3.

Bytes #1-2: MIDI port ID (1 - N).
Byte #3: filter ID:

Filter ID	Description
1	input filter

Filter ID	Description
2	output filter

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x24                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x01                      // port ID
0x01                            // filter ID (input)
0xxx                            // checksum
0xF7                            // footer
```

3.6. RetMIDIPortFilter / SetMIDIPortFilter (Command ID = 0x25)

RetMIDIPortFilter is sent by devices in response to a GetMIDIPortFilter command.

SetMIDIPortFilter is sent by a host to a device to set the values of some of the port's filter parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMIDIPortFilter, Write for SetMIDIPortFilter.
Command ID is 0x25.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: port ID (1 - N).
Byte #4: filter ID (1 - N).
Byte #5: maximum number of controller filters supported by this filter (0 - 127).

Bytes #6-7 [W]: bitmap indicating filter status for the following system messages:

Bit	Description
15 - 9	always zero
8	set if port should filter MIDI reset events (0xFF)

Bit	Description
7	always zero
6	set if port should filter MIDI active sensing events (0xFE)
5	set if port should filter MIDI realtime events (0xF8-0xFD)
4	set if port should filter MIDI tune request events (0xF6)
3	set if port should filter MIDI song select events (0xF3)
2	set if port should filter MIDI song position pointer events (0xF2)
1	set if port should filter MIDI time code (MTC) events (0xF1)
0	set if port should filter MIDI system exclusive events (0xF0, 0xF7)

Bytes #8-23 [W]: bitmap indicating filter status for channel messages. One byte for each of the 16 MIDI channels (MIDI channel #1 is the first byte, MIDI channel #16 is the last byte). Note that setting the filter for controller changes will filter out all controllers on that MIDI channel; to filter out specific controllers use the controller filters section (bytes 24 and up). Each bitmap for bytes #8-23 is as follows:

Bit	Description
7 - 6	always zero
5	set if port should filter MIDI pitch bend events (0xEn)
4	set if port should filter MIDI channel pressure events [mono aftertouch] (0xDn)
3	set if port should filter MIDI program change events (0xCn)
2	set if port should filter MIDI control change events (0xBn)
1	set if port should filter MIDI poly key pressure events [poly aftertouch] (0xAn)
0	set if port should filter MIDI note on/off events (0x8n, 0x9n)

Bytes #24-N [W]: controller filters. Bytes 24 and up are used only if byte #5 is not zero. Five bytes are used for each controller filter as follows:

Byte	Description
1 - 4	MIDI channel bitmap, specific to this controller filter
5	controller ID (0 - 127)

The MIDI channel bitmap has the following format:

Bit	Description
31 - 28	always zero
27	set if port should filter MIDI messages on channel 16
26	set if port should filter MIDI messages on channel 15
25	set if port should filter MIDI messages on channel 14
24	set if port should filter MIDI messages on channel 13
23 - 20	
19	set if port should filter MIDI messages on channel 12
18	set if port should filter MIDI messages on channel 11
17	set if port should filter MIDI messages on channel 10
16	set if port should filter MIDI messages on channel 9
15 - 12	always zero
11	set if port should filter MIDI messages on channel 8
10	set if port should filter MIDI messages on channel 7
9	set if port should filter MIDI messages on channel 6
8	set if port should filter MIDI messages on channel 5
7 - 4	always zero
3	set if port should filter MIDI messages on channel 4
2	set if port should filter MIDI messages on channel 3
1	set if port should filter MIDI messages on channel 2

Bit	Description
0	set if port should filter MIDI messages on channel 1

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x25                      // command flags and ID
0x00, 0x21                      // data length (33)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x01                            // filter ID (input)
0x02                            // number of controller filters (2)
0x00, 0x40                      // filter active sensing
0x20                            // filter pitch bend on MIDI channel 1
0x10                            // filter mono aftertouch on MIDI channel 2
0x04                            // filter all controllers on MIDI channel 3
0x3F                            // filter all channel messages on MIDI channel 4
0x20                            // filter pitch bend on MIDI channel 5
0x10                            // filter mono aftertouch on MIDI channel 6
0x04                            // filter all controllers on MIDI channel 7
0x3F                            // filter all channel messages on MIDI channel 8
0x20                            // filter pitch bend on MIDI channel 9
0x10                            // filter mono aftertouch on MIDI channel 10
0x04                            // filter all controllers on MIDI channel 11
0x3F                            // filter all channel messages on MIDI channel 12
0x20                            // filter pitch bend on MIDI channel 13
0x10                            // filter mono aftertouch on MIDI channel 14
0x04                            // filter all controllers on MIDI channel 15
0x3F                            // filter all channel messages on MIDI channel 16
0x00, 0x00, 0x00, 0x01, 0x07    // filter controller #7 (volume on channel 1)
0x00, 0x00, 0x00, 0x00, 0x40    // filter controller #64 (sustain pedal, filter disabled)
0xxx                            // checksum
0xF7                            // footer
```

3.7. GetMIDIPortRemap (Command ID = 0x26)

This command is used to query a device about a specific MIDI port's remap. Devices that support this command will respond with a RetMIDIPortRemap message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x26.
Data length is 3.

Bytes #1-2: MIDI port ID (1 - N).
Byte #3: remap ID:

Remap ID	Description
1	input remap
2	output remap

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x26                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x01                      // port ID
0x01                            // remap ID (input)
0xxx                            // checksum
0xF7                            // footer
```

3.8. RetMIDIPortRemap / SetMIDIPortRemap (Command ID = 0x27)

RetMIDIPortRemap is sent by devices in response to a GetMIDIPortRemap command.

SetMIDIPortRemap is sent by a host to a device to set the values of some of the port's remap parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMIDIPortRemap, Write for SetMIDIPortRemap.
Command ID is 0x27.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: port ID (1 - N).
Byte #4: remap ID (input = 1, output = 2).
Byte #5: maximum number of controller remaps supported by this remap (0 - 127).

Bytes #6-37 [W]: bitmap indicating remap status and remap channel for channel messages. Two bytes for each of the 16 MIDI channels (MIDI channel #1 is the first two bytes, MIDI channel #16 is the last two bytes). Each bitmap for bytes #6-37 is as follows:

Bit	Description
15 - 14	always zero
13	set if port should remap MIDI pitch bend events (0xEn)
12	set if port should remap MIDI channel pressure events [mono aftertouch] (0xDn)
11	set if port should remap MIDI program change events (0xCn)
10	set if port should remap MIDI control change events (0xBn)
9	set if port should remap MIDI poly key pressure events [poly aftertouch] (0xAn)
8	set if port should remap MIDI note on/off events (0x8n, 0x9n)
7 - 4	reserved (always zero)
3 - 0	MIDI channel number for remap. MIDI channels are defined as 1 through 16 but the value used is 0 through 15 (i.e. MIDI channel 1 has value 0, MIDI channel 2 has value 1, etc.)

Bytes #38-N [W]: controller remap flags and value pairs. Bytes 38 and up are only used if byte #5 in the header is not zero. 6 bytes are used for each controller remap as follows:

Byte	Description
1 - 4	MIDI channel bitmap, specific to this controller remap
5	controller source ID (0 - 127)
6	controller destination ID (0 - 127)

The MIDI channel bitmap has the following format:

Bit	Description
31 - 28	always zero
27	set if port should remap MIDI messages on channel 16
26	set if port should remap MIDI messages on channel 15
25	set if port should remap MIDI messages on channel 14

Bit	Description
24	set if port should remap MIDI messages on channel 13
23 - 20	
19	set if port should remap MIDI messages on channel 12
18	set if port should remap MIDI messages on channel 11
17	set if port should remap MIDI messages on channel 10
16	set if port should remap MIDI messages on channel 9
15 - 12	always zero
11	set if port should remap MIDI messages on channel 8
10	set if port should remap MIDI messages on channel 7
9	set if port should remap MIDI messages on channel 6
8	set if port should remap MIDI messages on channel 5
7 - 4	always zero
3	set if port should remap MIDI messages on channel 4
2	set if port should remap MIDI messages on channel 3
1	set if port should remap MIDI messages on channel 2
0	set if port should remap MIDI messages on channel 1

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x27                      // command flags and ID
0x00, 0x31                      // data length (49)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x01                            // remap ID (input)
0x02                            // number of controller remaps (2)
0x3F, 0x09                      // channel 1: remap all messages to channel 10
0x00, 0x01                      // channel 2: don't remap any messages
0x01, 0x0F                      // channel 3: remap note events to channel 16
0x20, 0x00                      // channel 4: remap pitch bend to channel 1
```

```
0x00, 0x04                  // channel 5: don't remap any messages
0x3F, 0x01                  // channel 6: remap all messages to channel 2
0x01, 0x02                  // channel 7: remap mono aftertouch to channel 3
0x00, 0x07                  // channel 8: don't remap any messages
0x3F, 0x09                  // channel 9: remap all messages to channel 10
0x00, 0x09                  // channel 10: don't remap any messages
0x01, 0x0F                  // channel 11: remap note events to channel 16
0x20, 0x00                  // channel 12: remap pitch bend to channel 1
0x00, 0x0C                  // channel 13: don't remap any messages
0x3F, 0x01                  // channel 14: remap all messages to channel 2
0x01, 0x02                  // channel 15: remap mono aftertouch to channel 3
0x00, 0x0F                  // channel 16: don't remap any messages
0x0F, 0x0F, 0x0F, 0x0F,             // remap controller #1 to controller #2 (all channels)
0x01, 0x02
0x00, 0x00, 0x00, 0x03,     // remap controller #3 to controller #4 (channels 1 & 2)
0x03, 0x04
0xxx                        // checksum
0xF7                        // footer
```

3.9. GetMIDIPortRoute (Command ID = 0x28)

This command is used to query a device about a specific MIDI port's routing. Devices that support this command will respond with a RetMIDIPortRoute message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x28.
Data length is 2.

Bytes #1-2: MIDI port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x28                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // port ID
0xxx                            // checksum
0xF7                            // footer
```

3.10. RetMIDIPortRoute / SetMIDIPortRoute (Command ID = 0x29)

RetMIDIPortRoute is sent by devices in response to a GetMIDIPortRoute command.

SetMIDIPortRoute is sent by a host to a device to set the port's routing. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMIDIPortRoute, Write for SetMIDIPortRoute.
Command ID is 0x29.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: port ID (1 - N).
Bytes #4-N [W]: bitmap indicating port routing of this port to all other ports. Bit 0 is port #1, bit 1 is port #2, etc. Least significant byte is first, most significant byte is last. The most significant 4 bits of each byte must be 0 which allows 4 ports to be specified per byte. Unused bits should be set to 0. The number of ports is given in the RetMIDIInfo command. The number of bytes (in the bitmap) is an even value so that the bitmap (when unpacked) is a multiple of 8 bits:

number of bytes = (((number of ports - 1) / 8) + 1) x 2

Example: device has 20 ports, route port #1 to ports 2, 3, 7, 11-14, 19, 20

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x29                      // command flags and ID
0x00, 0x09                      // data length (9)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x06                            // routing for ports 4 through 1
0x04                            // routing for ports 8 through 5
0x0C                            // routing for ports 12 through 9
0x03                            // routing for ports 16 through 13
0x08                            // routing for ports 20 through 17
0x00                            // routing for ports 24 through 21 (padding)
0xxx                            // checksum
0xF7                            // footer
```

3.11. GetMIDIPortDetail (Command ID = 0x2A)

This command is used to query a device about details related to a specific MIDI port. Devices that support this command will respond with a RetMIDIPortDetail message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x2A.
Data length is 2.

Bytes #1-2: MIDI port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x2A                     // command flags and ID
0x00, 0x02                     // data length
0x00, 0x01                     // port ID
0xxx                           // checksum
0xF7                           // footer
```

3.12. RetMIDIPortDetail / SetMIDIPortDetail (Command ID = 0x2B)

RetMIDIPortDetail is sent by devices in response to a GetMIDIPortDetail command.

SetMIDIPortDetail is sent by a host to a device to set the MIDI port details of USB host ports. DIN ports, USB device ports, and ethernet ports cannot be modified using this command. Writeable parameters for USB host ports are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. The host only needs to send up to (and including) byte #7, all bytes beyond #7 are read-only and are ignored.

Command Data

Command flags are Answer for RetMIDIPortDetail, Write for SetMIDIPortDetail.
Command ID is 0x2B.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: MIDI port ID (1 - N).

Byte #4: port type:

Value	Description
1	DIN
2	USB device
3	USB host
4	ethernet

Bytes #5-N: port detail, depends on port type (byte #4):

Type	Description
DIN	no details are available for DIN ports
USB device	Byte #5: host type: 0 = no host, 1 = Mac/PC, 2 = iOS device Byte #6: length of host name field that follows (0 - N) Bytes #7-N: host name, 7-bit ASCII string, not NULL terminated Host name is only applicable for host type = iOS device. For Mac/PC hosts (or no host at all), length field is always 0x00 and host name is empty.
USB host	Byte #5 [W]: Flags: bits 7-1 = always zero bit 0 = set if port is reserved for a specific vendor ID and product ID Byte #6 [W]: USB host ID (1 - N), is 0 if no device is being hosted on this port or if a port is inactive because a reserved device could not be connected. Byte #7 [W: Hosted device's MIDI port number (1 - 16), is 0 if no device is hosted or reserved on this port. Byte #8-10: Hosted or reserved device's USB vendor ID (16 bit value encoded in 3 bytes), is 0 if no device is hosted or reserved on this port. Byte #11-13: Hosted or reserved device's USB product ID (16 bit value encoded in 3 bytes, is 0 if no device is hosted or reserved on this port). Byte #14: Length of vendor name field that follows (0 - N). Bytes #X: Vendor name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device). Byte #X: Length of product name field that follows (0 - N). Bytes #X: Product name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device). The maximum value for USB host ID (byte #6) is the same as the number of USB-MIDI ports supported by each USB host jack (byte #11 in RetMIDIInfo command). The maximum value for Device MIDI port number (byte #7) depends on the number of MIDI ports supported by the hosted device (bytes #4-5 in RetUSBHostMIDIDeviceDetail command).
ethernet	Bytes #5-7: RTP-MIDI port number (16 bit value encoded in 3 bytes) Byte #8: number of active RTP-MIDI connections on this MIDI port (RTP-MIDI session) Byte #9: Length of session name (Bonjour name) field that follows (0 - N). Bytes #10-N: Session name (Bonjour name), 7-bit ASCII string, not NULL terminated. Session name is only valid if the ethernet port is connected to a network, is active, and has finished Bonjour name resolution.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
```

```
0x00, 0x2B                  // command flags and ID
0x00, 0x0A                  // data length (10)
0x01                        // command version number (1)
0x00, 0x03                  // port ID
0x02                        // port type = USB device
0x02                        // host type = iOS device
0x04                        // length of ASCII string that follows (4)
0x69, 0x50, 0x61, 0x64      // host name, the ASCII string “iPad”
0xxx                        // checksum
0xF7                        // footer
```

3.13. GetRTPMIDIConnectionDetail (Command ID = 0x2C)

This command is used to query a device about details related to a specific RTP-MIDI connection. Devices that support this command will respond with a RetRTPMIDIConnectionDetail message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x2C.
Data length is 3.

Bytes #1-2: MIDI port ID (1 - N).
Byte #3: RTP-MIDI connection number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x2C                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x3D                      // port ID (61)
0x01                            // connection number (1)
0xxx                            // checksum
0xF7                            // footer
```

3.14. RetRTPMIDIConnectionDetail (Command ID = 0x2D)

RetRTPMIDIConnectionDetail is sent by devices in response to a GetRTPMIDIConnectionDetail command.

Command Data

Command flags are Answer.
Command ID is 0x2D.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: MIDI port ID (1 - N).
Byte #4: RTP-MIDI connection number (1 - N).
Bytes #5-9: IP address of the device on the other side of this connection. A four-byte big-endian value encoded in five-bytes (similar to device ID) so that the most significant bit of each byte is 0.
Bytes #10-12: RTP-MIDI port number of the device on the other side of this connection (16 bit value encoded in 3 bytes).
Byte #13: Length of session name (Bonjour name) field that follows (0 - N).
Bytes #14-N: Session name (Bonjour name) of the device on the other side of this connection, 7-bit ASCII string, not NULL terminated.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x2D                      // command flags and ID
0x00, 0x11                      // data length (17)
0x01                            // command version number (1)
0x00, 0x3D                      // port ID (61)
0x01                            // connection number (1)
0x0C, 0x05, 0x20, 0x02, 0x65    // IP address (192.168.1.101)
0x00, 0x27, 0x0C                // port number (5004)
0x04                            // length of session name string that follows (4)
0x4D, 0x49, 0x44, 0x49          // the ASCII string “MIDI”
0xxx                            // checksum
0xF7                            // footer
```

3.15. GetUSBHostMIDIDeviceDetail (Command ID = 0x2E)

This command is used to query a device about details related to USB MIDI devices that are being hosted by USB host jacks. Devices that support this command will respond with a RetUSBHostMIDIDeviceDetail message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x2E.
Data length is 2.

Byte #1: USB host jack # (1 - N). Maximum value is the number of USB host jacks supported by this device (byte #8 in RetMIDIInfo command).
Byte #2: USB host ID (1 - N). Maximum value is the number of USB-MIDI ports supported by each USB host jack (byte #11 in RetMIDIInfo command).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x2E                      // command flags and ID
```

```
0x00, 0x02          // data length
0x01                // USB host jack # (1)
0x01                // USB host ID (1)
0xxx                // checksum
0xF7                // footer
```

3.16. RetUSBHostMIDIDeviceDetail (Command ID = 0x2F)

RetUSBHostMIDIDeviceDetail is sent by devices in response to a GetUSBHostMIDIDeviceDetail command.

Command Data

Command flags are Answer.
Command ID is 0x2F.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: USB host jack # (1 - N). Maximum value is the number of USB host jacks supported by this device (byte #8 in RetMIDIInfo command).
Byte #3: USB host ID (1 - N). Maximum value is the number of USB-MIDI ports supported by each USB host jack (byte #11 in RetMIDIInfo command).
Byte #4: Number of MIDI IN ports supported by the hosted device (1 - 16), is 0 if no device is connected.
Byte #5: Number of MIDI OUT ports supported by the hosted device (1 - 16), is 0 if no device is connected.
Byte #6-8: Hosted device's USB vendor ID (16 bit value encoded in 3 bytes), is 0 if no device is connected.
Byte #9-11: Hosted device's USB product ID (16 bit value encoded in 3 bytes), is 0 if no device is connected.
Byte #12: Length of vendor name field that follows (0 - N).
Bytes #X: Vendor name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device).
Byte #X: Length of product name field that follows (0 - N).
Bytes #X: Product name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x2F                      // command flags and ID
0x00, 0x15                      // data length (21)
0x01                            // command version number (1)
0x01                            // USB host jack # (1)
0x02                            // USB host ID (2)
0x04                            // number of MIDI IN ports (4)
0x04                            // number of MIDI OUT ports (4)
0x00, 0x46, 0x21                // vendor ID (0x2321)
0x00, 0x00, 0x0F                // product ID (0x000F)
0x04                            // length of vendor name string that follows (4)
0x49, 0x43, 0x4F, 0x4E          // the ASCII string "ICON"
```

```
0x04                    // length of product name string that follows (4)
0x49, 0x43, 0x4D, 0x32  // the ASCII string “ICM2”
0xxx                    // checksum
0xF7                    // footer
```

4. Audio Commands V2

The following commands are defined for protocol version = 1.

4.1. GetAudioGlobalParm (Command ID = 0x40)

This command is used to query a device about global audio parameters. Devices that support this command will respond with a RetAudioGlobalParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x40.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x40, 0x40                    // command flags and ID
0x00, 0x00                    // data length
0xxx                          // checksum
0xF7                          // footer
```

4.2. RetAudioGlobalParm / SetAudioGlobalParm (Command ID = 0x41)

RetAudioGlobalParm is sent by devices in response to a GetAudioGlobalParm command. It contains information that a host will need to use to communicate with this device for other audio related messages. A host should cache this information and use it for further communication with this device.

SetAudioGlobalParm is sent by a host to a device to set the current audio configuration. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioGlobalParm, Write for SetAudioGlobalParm.
Command ID is 0x41.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) byte #10, all bytes beyond #10 are read-only and are ignored.

Byte #2-3: total number of audio ports supported by this device (0 - N).
Byte #4: minimum number of audio frames that can be buffered (1 - N).
Byte #5: maximum number of audio frames that can be buffered (1 - N).
Byte #6 [W]: current number of audio frames to buffer (1 - N).
Byte #7: minimum allowed value for sync factor (1 - N).
Byte #8: maximum allowed value for sync factor (1 - N).
Byte #9 [W]: current sync factor value (1 - N).
Byte #10 [W]: number of the currently active audio configuration (1 - N).
Byte #11: number of configuration blocks that follow (0 - N).

Bytes #12-N: audio configurations blocks. Bytes 12 and up are used only if byte #11 is not zero. Three bytes are used for each audio configuration block as follows:

Byte	Description
1	audio configuration number (1 - N)
2	bit depth code: 1 = 4 bit, 2 = 8 bit, 3 = 12 bit, 4 = 16 bit bit depth code: 5 = 20 bit, 6 = 24 bit, 7 = 28 bit, 8 = 32 bit
3	sample rate code: 1 = 11025, 3 = 22050, 5 = 44100, 7 = 88200 sample rate code: 2 = 12000, 4 = 24000, 6 = 48000, 8 = 96000

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x00, 0x41                     // command flags and ID
0x00, 0x1D                     // data length (29)
0x01                           // command version number (1)
0x00, 0x06                     // device has 6 audio ports
0x01, 0x04, 0x02               // minimum, maximum, and current # of audio frames to buffer
0x01, 0x04, 0x02               // minimum, maximum, and current sync factor
0x02                           // currently active audio configuration is #2
0x06                           // number of audio configuration blocks that follow (6)
0x01, 0x04, 0x05               // configuration block #1: 16 bit, 44100
0x02, 0x04, 0x06               // configuration block #2: 16 bit, 48000
0x03, 0x04, 0x07               // configuration block #3: 16 bit, 88200
0x04, 0x04, 0x08               // configuration block #4: 16 bit, 96000
0x05, 0x06, 0x05               // configuration block #5: 24 bit, 44100
0x06, 0x06, 0x06               // configuration block #6: 24 bit, 48000
0xxx                           // checksum
0xF7                           // footer
```

4.3. GetAudioPortParm (Command ID = 0x42)

This command is used to query a device about a specific audio port. Devices that support this command will respond with a RetAudioPortParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x42.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x42                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID
0xxx                            // checksum
0xF7                            // footer
```

4.4. RetAudioPortParm / SetAudioPortParm (Command ID = 0x43)

RetAudioPortParm is sent by devices in response to a GetAudioPortParm command.

SetAudioPortParm is sent by a host to a device to set the values of some of the port parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioPortParm, Write for SetAudioPortParm.
Command ID is 0x43.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The port configuration blocks are read-only, thus the host can skip sending these by setting byte #7 to 0x00.

Bytes #2-3: audio port ID (1 - N).
Byte #4: port type:

Value	Description
2	USB device
3	USB host
4	ethernet
5	analogue

Byte #5 [W]: number of input channels to use for this port (0 - N), must be a legal value for the current audio configuration.
Byte #6 [W]: number of output channels to use for this port (0 - N), must be a legal value for the current audio configuration.
Byte #7: number of port configuration blocks that follow (0 - N).

Bytes #8-N: port configurations blocks, only included if byte #7 is not zero. There is one port configuration block for each audio configuration (from RetAudioGlobalParm command). Six bytes are used for each port configuration block as follows:

Byte	Description
1	audio configuration number (1 - N), from RetAudioGlobalParm command
2	maximum number of audio channels (0 - N) allowed for this port (for this audio configuration number), the number of input and output channels combined must not exceed this value
3	minimum number of input channels allowed for this port (0 - N)
4	maximum number of input channels allowed for this port (0 - N)
5	minimum number of output channels allowed for this port (0 - N)
6	maximum number of output channels allowed for this port (0 - N)

Byte #N: maximum length allowed for port name, 0 if read-only (0 - 127).
Byte #N [W]: length of port name field that follows (0 - N).
Bytes #N-N [W]: port name, 7-bit ASCII string, not NULL terminated.

Bytes #N-N: port info, depends on port type (byte #4):

Type	Description
USB device	Byte #1: USB device jack # (1 - N) Byte #2: Flags: bits 7 - 4: reserved (always 0) bit 3 [W]: set if port is currently enabled for audio with iOS device bit 2 [W]: set if port is currently enabled for audio with PC/Mac bit 1: set if port supports audio with iOS devices bit 0: set if port supports audio with PC/Mac
USB host	Byte #1: USB host jack # (1 - N) Byte #2: device number on this USB host jack (1 - N), USB host jacks may support several audio devices
ethernet	Byte #1: ethernet jack # (1 - N) Byte #2: device number on this ethernet jack (1 - N), ethernet jacks may support several audio devices
analogue	Byte #1: analogue port # (1 - N)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E          // header
0x00, 0x05                            // product ID
0x01, 0x02, 0x03, 0x04, 0x05          // serial number
0x00, 0x00                            // transaction ID
0x00, 0x43                            // command flags and ID
0x00, 0x2D                            // data length (45)
0x01                                  // command version number (1)
0x00, 0x01                            // audio port ID
0x02                                  // port type = USB device
0x04                                  // number of input channels to use for this port (4)
0x04                                  // number of output channels to use for this port (4)
0x06                                  // number of port configuration blocks that follow (6)
0x01, 0x08, 0x01, 0x07, 0x01, 0x07    // port configuration block #1 (16/44): 8 max, 1/7 in, 1/7 out
0x02, 0x08, 0x01, 0x07, 0x01, 0x07    // port configuration block #2 (16/48): 8 max, 1/7 in, 1/7 out
0x03, 0x04, 0x01, 0x03, 0x01, 0x03    // port configuration block #3 (16/88): 4 max, 1/3 in, 1/3 out
0x04, 0x04, 0x01, 0x03, 0x01, 0x03    // port configuration block #4 (16/96): 4 max, 1/3 in, 1/3 out
0x05, 0x04, 0x01, 0x03, 0x01, 0x03    // port configuration block #5 (24/44): 4 max, 1/3 in, 1/3 out
0x06, 0x04, 0x01, 0x03, 0x01, 0x03    // port configuration block #6 (24/48): 4 max, 1/3 in, 1/3 out
0x0F                                  // maximum length allowed for port name (15 ASCII characters)
0x04                                  // length of port name field that follows (4 ASCII characters)
0x55, 0x53, 0x42, 0x31                // port name, the ASCII string “USB1”
0x01                                  // port is first USB device jack
0x07                                  // port supports PC/Mac and iOS audio, iOS audio is disabled
0xxx                                  // checksum
0xF7                                  // footer
```

4.5. GetAudioDeviceParm (Command ID = 0x44)

This command is used to query a device about details related to a specific audio device. Devices that support this command will respond with a RetAudioDeviceParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x44.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x44                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID
0xxx                            // checksum
0xF7                            // footer
```

4.6. RetAudioDeviceParm / SetAudioDeviceParm (Command ID = 0x45)

RetAudioDeviceParm is sent by devices in response to a GetAudioDeviceParm command.

SetAudioDeviceParm is sent by a host to a device to set the audio device details for USB host ports. Other types of audio ports cannot be modified using this command. Writeable parameters for USB host ports are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioDeviceParm, Write for SetAudioDeviceParm.
Command ID is 0x45.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) the last writeable byte. All other bytes are read-only and are ignored.

Bytes #2-3: audio port ID (1 - N).
Byte #4: port type:

Value	Description
2	USB device
3	USB host
4	ethernet
5	analogue

Byte #5: maximum number of controllers supported by this device (0 - N).
Bytes #6-N: device detail, depends on port type (byte #4):

Type	Description
USB device	Byte #6: host type: 0 = no host, 1 = Mac/PC, 2 = iOS device Byte #7: length of host name field that follows (0 - N) Bytes #8-N: host name, 7-bit ASCII string, not NULL terminated Host name is only applicable for host type = iOS device. For Mac/PC hosts (or no host at all), length field is always 0x00 and host name is empty.

Type	Description
USB host	Byte #6: Flags: bit 7 = always zero bit 6 = set if a device is connected bits 5-1 = always zero bit 0 [W] = set if port is reserved for a specific vendor ID and product ID Byte #7: Maximum number of input channels supported by the hosted or reserved device, is 0 if no device is hosted or reserved on this port. Byte #8: Maximum number of output channels supported by the hosted or reserved device, is 0 if no device is hosted or reserved on this port. Byte #9: Number of input channel maps that follow (value should be the save as byte #5 from RetAudioPortParm. Bytes #N-N [W]: port input channel to device input channel map. Two bytes for each port input channel. First byte is port input channel number (1 - N), second byte is device input channel number (1 - N), or 0 if nothing is connected. There can only be one connection per device input channel. Byte #N: Number of output channel maps that follow (value should be the save as byte #6 from RetAudioPortParm. Bytes #N-N [W]: port output channel to device output channel map. Two bytes for each port output channel. First byte is port output channel number (1 - N), second byte is device output channel number (1 - N), or 0 if nothing is connected. There can only be one connection per device output channel. Bytes #N-N: Hosted or reserved device's USB vendor ID (16 bit value encoded in 3 bytes), is 0 if no device is hosted or reserved on this port. Bytes #N-N: Hosted or reserved device's USB product ID (16 bit value encoded in 3 bytes, is 0 if no device is hosted or reserved on this port). Byte #N: Length of vendor name field that follows (0 - N). Bytes #N-N: Vendor name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device). Byte #N: Length of product name field that follows (0 - N). Bytes #N-N: Product name, 7-bit ASCII string, not NULL terminated (from USB descriptor of hosted device).
ethernet	no details are available for ethernet ports at this time
analogue	no details are available for analogue ports at this time

Example 1 (USB device):

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x45                      // command flags and ID
0x00, 0x0B                      // data length (11)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x02                            // port type = USB device
0x00                            // device has no controllers
0x02                            // host type = iOS device
```

```
0x04                          // length of ASCII string that follows (4)
0x69, 0x50, 0x61, 0x64        // host name, the ASCII string “iPad”
0xxx                          // checksum
0xF7                          // footer
```

Example 2 (USB host):

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                    // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                    // transaction ID
0x00, 0x45                    // command flags and ID
0x00, 0x2A                    // data length (42)
0x01                          // command version number (1)
0x00, 0x04                    // port ID
0x03                          // port type = USB host
0x02                          // device has 2 controllers
0x01                          // device is reserved but not connected
0x03                          // maximum # of input channels device supports (3)
0x02                          // maximum # of output channels device supports (2)
0x04                          // port has 4 inputs:
0x01, 0x02                    // port input #1 connected to device input #2
0x02, 0x01                    // port input #2 connected to device input #1
0x03, 0x03                    // port input #3 connected to device input #3
0x04, 0x00                    // port input #4 not connected to anything
0x04                          // port has 4 outputs:
0x01, 0x00                    // port output #1 not connected to anything
0x02, 0x00                    // port output #2 not connected to anything
0x03, 0x01                    // port output #3 connected to device output #1
0x04, 0x02                    // port output #4 connected to device output #2
0x00, 0x46, 0x21              // vendor ID (0x2321)
0x00, 0x00, 0x0F              // product ID (0x000F)
0x04                          // length of vendor name string that follows (4)
0x49, 0x43, 0x4F, 0x4E        // the ASCII string “ICON”
0x04                          // length of product name string that follows (4)
0x49, 0x43, 0x41, 0x34        // the ASCII string “ICA4”
0xxx                          // checksum
0xF7                          // footer
```

4.7. GetAudioControlParm (Command ID = 0x46)

This command is used to query a device about a controller for a specific audio device. Devices that support this command will respond with a RetAudioControlParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x46.
Data length is 3.

Bytes #1-2: audio port ID (1 - N).
Byte #3: controller number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x46                     // command flags and ID
0x00, 0x03                     // data length
0x00, 0x01                     // audio port ID
0x01                           // controller number
0xxx                           // checksum
0xF7                           // footer
```

4.8. RetAudioControlParm / SetAudioControlParm (Command ID = 0x47)

RetAudioControlParm is sent by devices in response to a GetAudioControlParm command.

SetAudioControlParm is sent by a host to a device to set the values for some of the controller parameters. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioControlParm, Write for SetAudioControlParm.
Command ID is 0x47.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) the last writeable byte. All other bytes are read-only and are ignored.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: controller number (1 - N).
Byte #5: controller type:

Value	Description
5	Selector
6	Feature
10	Clock Source

Bytes #6-N: controller values, depends on controller type (byte #5):

Type	Description
Selector	Byte #6 [W]: current selector input Byte #7: number of selector inputs (controller details) Byte #8: length of controller name field that follows (0 - N) Bytes #9-N: controller name, 7-bit ASCII string, not NULL terminated
Feature	Byte #6: number of feature channels (controller details) Byte #7: length of controller name field that follows (0 - N) Bytes #8-N: controller name, 7-bit ASCII string, not NULL terminated
Clock Source	Byte #6 [W]: current clock source input Byte #7: number of clock source inputs (controller details) Byte #8: length of controller name field that follows (0 - N) Bytes #9-N: controller name, 7-bit ASCII string, not NULL terminated

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x47                      // command flags and ID
0x00, 0x0D                      // data length (13)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x02                            // controller number (2)
0x05                            // controller type (selector)
0x02                            // current selector input (2)
0x03                            // number of selector inputs (3)
0x05                            // length of ASCII string that follows (5)
0x49, 0x6E, 0x70, 0x75, 0x74    // controller name, the ASCII string “Input”
0xxx                            // checksum
0xF7                            // footer
```

4.9. GetAudioControlDetail (Command ID = 0x48)

This command is used to query a device about the item details for a controller for a specific audio device. Devices that support this command will respond with a RetAudioControlDetail message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x48.
Data length is 4.

Bytes #1-2: audio port ID (1 - N).
Byte #3: controller number (1 - N).
Byte #4: detail number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x48                      // command flags and ID
0x00, 0x04                      // data length
0x00, 0x01                      // audio port ID
0x01                            // controller number
0x02                            // detail number
0xxx                            // checksum
0xF7                            // footer
```

4.10. RetAudioControlDetail / SetAudioControlDetail (Command ID = 0x49)

RetAudioControlDetail is sent by devices in response to a GetAudioControlDetail command.

SetAudioControlDetail is sent by a host to a device to set the values for some of the controller details. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioControlDetail, Write for SetAudioControlDetail.
Command ID is 0x49.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) the last writeable byte. All other bytes are read-only and are ignored.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: controller number (1 - N).
Byte #5: detail number (1 - N).
Byte #6: controller type:

Value	Description
5	Selector
6	Feature
10	Clock Source

Bytes #7-N: controller detail, depends on controller type (byte #6):

Type	Description
Selector	Byte #7: length of selector input name field that follows (0 - N) Bytes #8-N: selector input name, 7-bit ASCII string, not NULL terminated
Feature	Byte #7: associated audio channel type: 0 = no association 1 = input channel 2 = output channel Byte #8: associated audio channel number (1 - N), 0 if not associated. Byte #9: exist flags: bits 7-5 = always zero bit 4 = set if stereo link control exists bit 3 = set if high impedance control exists bit 2 = set if phantom power control exists bit 1 = set if mute control exists bit 0 = set if volume control exists Byte #10: edit flags: bits 7-5 = always zero bit 4 = set if stereo link control is editable bit 3 = set if high impedance control is editable bit 2 = set if phantom power control is editable bit 1 = set if mute control is editable bit 0 = set if volume control is editable The following 6 values are only present if volume control exists: Bytes #11-13: minimum value of volume control (16 bit value encoded in 3 bytes) Bytes #14-16: maximum value of volume control (16 bit value encoded in 3 bytes) Bytes #17-19: resolution of volume control (16 bit value encoded in 3 bytes) Bytes #20-22: pad value for volume control (16 bit value encoded in 3 bytes), optional offset applied to volume control values when using line inputs (as opposed to mic inputs), 0 means “no pad” Bytes #23-25: minimum value of trim control (16 bit value encoded in 3 bytes) Bytes #26-28: maximum value of trim control (16 bit value encoded in 3 bytes) Byte #29: length of channel name field that follows (0 - N) Bytes #30-N: channel name, 7-bit ASCII string, not NULL terminated The volume, trim, and pad values are 16 bit 2’s complement numbers that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). Resolution can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF). In addition, code 0x8000, representing silence (i.e., -∞ dB), must always be implemented. However, it must never be reported as the minimum value.

Type	Description
Clock Source	Byte #7: length of clock source name field that follows (0 - N) Bytes #8-N: clock source name, 7-bit ASCII string, not NULL terminated

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x49                      // command flags and ID
0x00, 0x21                      // data length (33)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x01                            // controller number (1)
0x02                            // detail number (2)
0x06                            // controller type (feature)
0x02                            // detail is associated to an output channel (1)
0x01                            // detail is associated to input channel 1
0x03, 0x03                      // both mute and volume controls exist and are editable
0x00, 0x00, 0x00                // minimum value of volume control (0 dB = 0x0000)
0x02, 0x00, 0x00                // maximum value of volume control (16 dB = 0x1000)
0x00, 0x02, 0x00                // resolution of volume control (1 dB = 0x0100)
0x00, 0x00, 0x00                // pad value (no pad)
0x00, 0x00, 0x00                // minimum value of trim control (0 dB = 0x0000)
0x02, 0x00, 0x00                // maximum value of trim control (16 dB = 0x1000)
0x04                            // length of ASCII string that follows (4)
0x4C, 0x65, 0x66, 0x74          // channel name, the ASCII string “Left”
0xxx                            // checksum
0xF7                            // footer
```

4.11. GetAudioControlDetailValue (Command ID = 0x4A)

This command is used to query a device about the values for a controller for a specific audio device. Devices that support this command will respond with a RetAudioControlDetailValue message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x4A.
Data length is 4.

Bytes #1-2: audio port ID (1 - N).
Byte #3: controller number (1 - N).
Byte #4: detail number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
```

```
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x4A                      // command flags and ID
0x00, 0x04                      // data length
0x00, 0x01                      // audio port ID
0x01                            // controller number
0x02                            // detail number
0xxx                            // checksum
0xF7                            // footer
```

4.12. RetAudioControlDetailValue / SetAudioControlDetailValue (Command ID = 0x4B)

RetAudioControlDetailValue is sent by devices in response to a GetAudioControlDetailValue command.

SetAudioControlDetailValue is sent by a host to a device to set the values for some of the controller details. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioControlDetailValue, Write for SetAudioControlDetailValue.
Command ID is 0x4B.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) the last writeable byte. All other bytes are read-only and are ignored.

Bytes #2-3: audio port ID (1 - N).
Byte #4: controller number (1 - N).
Byte #5: detail number (1 - N).
Byte #6: controller type:

Value	Description
5	Selector
6	Feature
10	Clock Source

Bytes #7-N: controller detail, depends on controller type (byte #6):

Type	Description
Selector	not supported at this time
Feature	Byte #7: value flags: bits 7-5 = always zero bit 4 = set if stereo link control value is included bit 3 = set if high impedance control value is included bit 2 = set if phantom power control value included bit 1 = set if mute control value is included bit 0 = set if volume & trim control values are included Values follow for each of the flags in byte #7 (volume/trim first, mute second, etc): Bytes #8-10 [W]: value of volume control (16 bit value encoded in 3 bytes) Bytes #11-13 [W]: value of trim control (16 bit value encoded in 3 bytes) Byte #14 [W]: value of mute control (0 = mute off, 1 = mute on) Byte #15 [W]: value of phantom power control (0 = phantom power off, 1 = phantom power on) Byte #16 [W]: value of high impedance control (0 = high impedance off, 1 = high impedance on) Byte #17 [W]: value of stereo link control (0 = link off, 1 = link on) The volume and trim values are 16 bit 2's complement numbers that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001).
Clock Source	Byte #7: length of clock source name field that follows (0 - N) Bytes #8-N: clock source name, 7-bit ASCII string, not NULL terminated

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x4B                      // command flags and ID
0x00, 0x0E                      // data length (14)
0x01                            // command version number (1)
0x00, 0x01                      // port ID
0x01                            // controller number (1)
0x02                            // detail number (2)
0x06                            // controller type (feature)
0x03                            // mute and volume/trim value are included
0x00, 0x08, 0x00                // volume control value (4 dB = 0x0400)
0x00, 0x00, 0x00                // trim control value (0 dB = 0x0000)
0x00                            // mute is off (0)
0xxx                            // checksum
0xF7                            // footer
```

4.13. GetAudioClockParm (Command ID = 0x4C)

This command is used to query a device about audio clock sources. Devices that support this command will respond with a RetAudioClockParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x4C.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x4C                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

4.14. RetAudioClockParm / SetAudioClockParm (Command ID = 0x4D)

RetAudioClockParm is sent by devices in response to a GetAudioClockParm command. It contains information regarding audio clock sources for the device.

SetAudioClockParm is sent by a host to a device to set the audio clock source. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. The host only needs to send up to (and including) byte #2, all bytes beyond #2 are read-only and are ignored.

Command Data

Command flags are Answer for RetAudioClockParm, Write for SetAudioClockParm.
Command ID is 0x4D.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2 [W]: number of the active audio clock source block (1 - N).
Byte #3: number of audio clock source blocks that follow (0 - N).

Bytes #4-N: audio clock source blocks. Bytes 4 and up are used only if byte #3 is not zero. Four bytes are used for each audio clock source block as follows:

Byte	Description
1	audio clock source number (1 - N)
2	clock source type: 1 = internal clock 2 = audio port clock
3 - 4	For clock source type = internal clock: Byte 3: internal clock ID (1 - N) Byte 4: reserved (always 0) For clock source type = audio port clock: Byte 3: audio port ID (1 - N) Byte 4: device number (1 - N)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x4D                      // command flags and ID
0x00, 0x0F                      // data length (15)
0x01                            // command version number (1)
0x02                            // active audio clock source block is #2
0x03                            // number of audio clock blocks that follow (3)
0x01, 0x01, 0x01, 0x00          // audio clock source block #1: internal clock #1
0x02, 0x02, 0x01, 0x01          // audio clock source block #2: USB audio port #1, device #1
0x03, 0x02, 0x02, 0x01          // audio clock source block #3: USB audio port #2, device #1
0xxx                            // checksum
0xF7                            // footer
```

4.15. GetAudioPatchbayParm (Command ID = 0x4E)

This command is used to query a device about audio port patchbay setup. Devices that support this command will respond with a RetAudioPatchbayParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x4E.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
```

```
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x4E                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID
0xxx                            // checksum
0xF7                            // footer
```

4.16. RetAudioPatchbayParm / SetAudioPatchbayParm (Command ID = 0x4F)

RetAudioPatchbayParm is sent by devices in response to a GetAudioPatchbayParm command. It contains the patchbay setup for a specific audio port.

SetAudioPatchbayParm is sent by a host to a device to configure the patchbay for a specific audio port.

Command Data

Command flags are Answer for RetAudioPatchbayParm, Write for SetAudioPatchbayParm.
Command ID is 0x4F.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).

Byte #4: number of patchbay configuration blocks that follow (0 - N).

Bytes #5-N: patchbay configurations blocks. Bytes 5 and up are used only if byte #4 is not zero. If byte #4 is not zero, there should be one patchbay configuration block for each input channel for this port. Four bytes are used for each port configuration block as follows:

Byte	Description
1	input channel number (1 - N)
2	output channel number of the port driving this input (1 - N)
3 - 4	port ID of output port driving this input

All values in each patch configuration block must be valid for the output port ID, output channel number, and input channel number. Use 0x00 for bytes #2-4 to indicate “no connection” (i.e. no output is driving the input).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
```

```
0x00, 0x05                  // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                  // transaction ID
0x00, 0x4F                  // command flags and ID
0x00, 0x14                  // data length (20)
0x01                        // command version number (1)
0x00, 0x01                  // audio port ID
0x04                        // number of patchbay configuration blocks that follow (4)
0x01, 0x03, 0x00, 0x02      // patchbay block #1: input #1 is connected to output channel 3 of port #2
0x02, 0x01, 0x00, 0x03      // patchbay block #2: input #2 is connected to output channel 1 of port #3
0x03, 0x02, 0x00, 0x03      // patchbay block #3: input #3 is connected to output channel 2 of port #3
0x04, 0x00, 0x00, 0x00      // patchbay block #4: input #4 is not connected to anything
0xxx                        // checksum
0xF7                        // footer
```

4.17. GetAudioChannelName (Command ID = 0x3C)

This command is used to query a device about the names of channels on a specific audio port. Devices that support this command will respond with a RetAudioChannelName message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x3C.
Data length is 4.

Bytes #1-2: audio port ID (1 - N)
Byte #3: audio channel number (1 - N)
Byte #4: input/output selector (1 = input, 2 = output)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E  // header
0x00, 0x05                  // product ID
0x01, 0x02, 0x03, 0x04, 0x05  // serial number
0x00, 0x00                  // transaction ID
0x40, 0x3C                  // command flags and ID
0x00, 0x04                  // data length
0x00, 0x01                  // audio port ID (1)
0x03                        // audio channel (3)
0x02                        // input/output selector (2 = output)
0xxx                        // checksum
0xF7                        // footer
```

4.18. RetAudioChannelName / SetAudioChannelName (Command ID = 0x3D)

RetAudioChannelName is sent by devices in response to a GetAudioChannelName command.

SetAudioChannelName is sent by a host to a device to set the name of an audio channel on a specific audio port. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioChannelName, Write for SetAudioChannelName.
Command ID is 0x3D.
Data length depends on length of channel name.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: audio channel number (1 - N).
Byte #5: input/output flag (1 = input, 2 = output)
Byte #6: maximum length allowed for channel name, 0 if read-only (0 - 127).
Byte #7 [W]: length of channel name field that follows (0 - N).
Bytes #8-N [W]: channel name, 7-bit ASCII string, not NULL terminated.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x3D                      // command flags and ID
0x00, 0x0B                      // data length (11)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x03                            // audio channel number (3)
0x02                            // input/output flag (2 = output)
0x0F                            // maximum length allowed for channel name (15 ASCII characters)
0x04                            // length of ASCII string that follows (4)
0x4C, 0x65, 0x66, 0x74          // channel name, the ASCII string “Left”
0xxx                            // checksum
0xF7                            // footer
```

4.19. GetAudioPortMeterValue (Command ID = 0x3E)

This command is used to query a device for the most recent meter values for a specific audio port. Devices that support this command will respond with a RetAudioPortMeterValue message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x3E.
Data length is 3.

Bytes #1-2: audio port ID (1 - N)
Byte #3: bitmap indicating which meter values should be returned from the device:

Bit	Description
7 - 2	always zero
1	channel outputs
0	channel inputs

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x3E                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x01                      // audio port ID (1)
0x03                            // return meter values for channel inputs and channel outputs
0xxx                            // checksum
0xF7                            // footer
```

4.20. RetAudioPortMeterValue (Command ID = 0x3F)

RetAudioPortMeterValue is sent by devices in response to a GetAudioPortMeterValue command.

Command Data

Command flags are Answer.
Command ID is 0x3F.
Data length depends on the number of audio channels.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: number of audio meter blocks that follow.
Bytes #5-N: audio meter blocks:

Byte	Description
1	Bitmap indicating which meter values are in this block: bits 7 - 2: always zero bit 1: channel outputs bit 0: channel inputs

Byte	Description
2	Number of meter values that follow.
3 - 4 (channel 1), 5 - 6 (channel 2), 7 - 8 (channel 3), etc.	Meter values. Each meter value is a 14 bit big-endian number split across 2 bytes. 0 dB is 8192 (0x2000) and is the maximum value that can be returned (representing digital full scale). The first value is channel 1, the second value is channel 2, etc. To convert to decibels (dB) use the formula: dB = 20 log (value/8192) Note that the minimum meter value (1) corresponds to -78.27 dB but resolution drops to 1 dB when meter value is 8 (-60 dB).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x3F                      // command flags and ID
0x00, 0x14                      // data length (20)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x02                            // number of audio meter blocks that follow (2)
0x01                            // block #1: channel inputs (0x01)
0x02                            // meter readings that follow (2)
0x00, 0x00                      // input channel 1: meter value = 0x0000
0x20, 0x00                      // input channel 2: meter value = 0x1000
0x02                            // block #2: channel outputs (0x02)
0x04                            // meter readings that follow (4)
0x00, 0x00                      // output channel 1: meter value = 0x0000
0x40, 0x00                      // output channel 2: meter value = 0x2000
0x1F, 0x7F                      // output channel 3: meter value = 0x0FFF
0x01, 0x7F                      // output channel 4: meter value = 0x00FF
0xxx                            // checksum
0xF7                            // footer
```

5. Audio Mixer Commands

The following commands are defined for protocol version = 1.

5.1. GetMixerParm (Command ID = 0x50)

This command is used to query a device about audio mixer configurations. Devices that support this command will respond with a RetMixerParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x50.
Data length is 1.

Byte #1: audio configuration number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x50                      // command flags and ID
0x00, 0x01                      // data length
0x01                            // audio configuration number (1) [16 bit, 44100]
0xxx                            // checksum
0xF7                            // footer
```

5.2. RetMixerParm / SetMixerParm (Command ID = 0x51)

RetMixerParm is sent by devices in response to a GetMixerParm command. It contains information regarding mixer configurations. A host should cache this information and use it for further communication with this device.

SetMixerParm is sent by a host to a device to set the current mixer configuration. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMixerParm, Write for SetMixerParm.
Command ID is 0x51.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) byte #2, all bytes beyond #2 are read-only and are ignored.

Byte #2 [W]: number of the active mixer configuration block (1 - N). Must be valid for the active audio configuration.
Byte #3: audio configuration number (1 - N).
Byte #4: number of audio configuration mixer blocks that follow (1 - N) for this audio configuration.
Bytes #5-N: audio configuration mixer blocks. Three bytes are used for each block as follows:

Byte	Description
1	mixer configuration number (1 - N)
2	maximum number of mixer inputs (1 - N)
3	maximum number of mixer outputs (1 - N)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x51                      // command flags and ID
0x00, 0x0D                      // data length (13)
0x01                            // command version number (1)
0x03                            // active mixer configuration block (3)
0x01                            // audio configuration number (1) [16 bit, 44100]
0x03                            // number of mixer configuration blocks that follow (3)
0x01, 0x0C, 0x08                // block #1: 12 inputs, 8 outputs
0x02, 0x0A, 0x0A                // block #2: 10 inputs, 10 outputs
0x03, 0x08, 0x0C                // block #3: 8 inputs, 12 outputs
0xxx                            // checksum
0xF7                            // footer
```

5.3. GetMixerPortParm (Command ID = 0x52)

This command is used to query a device about audio port mixer configurations. Devices that support this command will respond with a RetMixerPortParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x52.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x52                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

5.4. RetMixerPortParm / SetMixerPortParm (Command ID = 0x53)

RetMixerPortParm is sent by devices in response to a GetMixerPortParm command. It contains information regarding audio port mixer configurations. A host should cache this information and use it for further communication with this device.

SetMixerPortParm is sent by a host to a device to set the current audio port mixer configuration. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMixerPortParm, Write for SetMixerPortParm.
Command ID is 0x53.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2: number of audio port mixer blocks that follow (1 - N).
Bytes #3-N: audio port mixer blocks. One block for each audio port.

Byte	Description
1 - 2	audio port ID (1 - N)
3 [W]	number of mixer inputs to use for this audio port (0 - N) The number of inputs must be less than or equal to the number of inputs for the active mixer configuration.
4 [W]	number of mixer outputs to use for this audio port (0 - N) The sum of all outputs for all ports must be less than or equal to the number of outputs for the active mixer configuration.

Example (active mixer configuration has 8 inputs and 12 outputs):

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x53                      // command flags and ID
0x00, 0x0E                      // data length (14)
0x01                            // command version number (1)
0x03                            // number of audio port mixer blocks that follow (3)
0x00, 0x01, 0x08, 0x02          // audio port #1: 8 inputs, 2 outputs
0x00, 0x02, 0x08, 0x06          // audio port #2: 8 inputs, 6 outputs
0x00, 0x03, 0x08, 0x04          // audio port #3: 8 inputs, 4 outputs
```

0xxx // checksum
0xF7 // footer

5.5. GetMixerInputParm (Command ID = 0x54)

This command is used to query a device about mixer input assignments for a specific audio port. Devices that support this command will respond with a RetMixerInputParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x54.
Data length is 3.

Bytes #1-2: audio port ID (1 - N).
Byte #3: mixer input number (1 - N).

Example:

0xF0, 0x00, 0x01, 0x73, 0x7E // header
0x00, 0x05 // product ID
0x01, 0x02, 0x03, 0x04, 0x05 // serial number
0x00, 0x00 // transaction ID
0x40, 0x54 // command flags and ID
0x00, 0x03 // data length
0x00, 0x01 // audio port ID (1)
0x01 // mixer input number (1)
0xxx // checksum
0xF7 // footer

5.6. RetMixerInputParm / SetMixerInputParm (Command ID = 0x55)

RetMixerInputParm is sent by devices in response to a GetMixerInputParm command.

SetMixerInputParm is sent by a host to a device to set the mixer input channel parameters for a specific audio port. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMixerInputParm, Write for SetMixerInputParm.
Command ID is 0x55.
Data length is 7.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: mixer input number (1 - N).

Bytes #5-6 [W]: audio port ID of audio source (0 - N). 0 if mixer input is not assigned.
Byte #7 [W]: channel number of audio source (0 - N). 0 if mixer input is not assigned.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x00, 0x55                     // command flags and ID
0x00, 0x07                     // data length (7)
0x01                           // command version number (1)
0x00, 0x01                     // audio port ID (1)
0x01                           // mixer input number (1)
0x00, 0x03                     // mixer input is audio port 3
0x04                           // mixer input is channel 4 (of audio port 3)
0xxx                           // checksum
0xF7                           // footer
```

5.7. GetMixerOutputParm (Command ID = 0x56)

This command is used to query a device about mixer output assignments for a specific audio port. Devices that support this command will respond with a RetMixerOutputParm message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x56.
Data length is 3.

Bytes #1-2: audio port ID (1 - N).
Byte #3: mixer output number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x56                     // command flags and ID
0x00, 0x03                     // data length
0x00, 0x01                     // audio port ID (1)
0x01                           // mixer output number (1)
0xxx                           // checksum
0xF7                           // footer
```

5.8. RetMixerOutputParm / SetMixerOutputParm (Command ID = 0x57)

RetMixerOutputParm is sent by devices in response to a GetMixerOutputParm command.

SetMixerOutputParm is sent by a host to a device to set the mixer output channel parameters for a specific audio port. All parameters must be sent in the message but only the writeable parameters are

changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMixerOutputParm, Write for SetMixerOutputParm.
Command ID is 0x57.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: mixer output number (1 - N)
Byte #5 [W]: number of mixer output assignments that follow (0 - N).
Bytes #6-N [W]: channel number of mixer output assignment on this audio port (1 - N).
Byte #N: maximum length allowed for mix name, 0 if read-only (0 - 127).
Byte #N [W]: length of mix name field that follows (0 - N).
Bytes #N-N [W]: mix name, 7-bit ASCII string, not NULL terminated.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x57                      // command flags and ID
0x00, 0x0D                      // data length (13)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x01                            // mixer output number (1)
0x02                            // number of mixer assignments that follow (2)
0x04                            // mixer output 1 is sent to output channel 4 (of audio port 1)
0x06                            // mixer output 1 is sent to output channel 6 (of audio port 1)
0x0F                            // maximum length allowed for mix name (15 ASCII characters)
0x04                            // length of ASCII string that follows (4)
0x4C, 0x65, 0x66, 0x74          // mix name, the ASCII string “Left”
0xxx                            // checksum
0xF7                            // footer
```

5.9. GetMixerInputControl (Command ID = 0x58)

This command is used to query a device about mixer input controls for a specific audio port. Devices that support this command will respond with a RetMixerInputControl message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x58.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x58                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID (1)
0xxx                            // checksum
0xF7                            // footer
```

5.10. RetMixerInputControl (Command ID = 0x59)

RetMixerInputControl is sent by devices in response to a GetMixerInputControl command.

Command Data

Command flags are Answer.
Command ID is 0x59.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).

Byte #4: exist flags:
- bit 7 = always zero
- bit 6 = set if pan control exists
- bit 5 = set if invert control exists
- bit 4 = set if stereo link control exists
- bit 3 = set if solo PFL control exists
- bit 2 = set if solo control exists
- bit 1 = set if mute control exists
- bit 0 = set if volume control exists

Byte #5: edit flags:
- bit 7 = always zero
- bit 6 = set if pan control is editable
- bit 5 = set if invert control is editable
- bit 4 = set if stereo link control is editable
- bit 3 = set if solo PFL control is editable
- bit 2 = set if solo control is editable
- bit 1 = set if mute control is editable
- bit 0 = set if volume control is editable

The following 3 values are only present if pan control exists:
Bytes #6-8: maximum value of pan control representing a signal panned fully right (a positive 16 bit 2's complement value encoded in 3 bytes), the negative of this value represents a signal panned fully left
Byte #9: number of pan curve laws that follow

Bytes #10-N: pan curve law (only included if byte #9 is not zero):

Value	Description
1	0 dB at center
2	-1.5 dB at center
3	-3 dB at center
4	-4.5 dB at center
5	-6 dB at center

The pan values are 16 bit 2's complement numbers that can range from +32767 (0x7FFF) down to -32767 (0x8001). Positive numbers are panning to the right. Negative numbers are panning to the left. 0 means fully centered. The maximum pan value representing a signal panned fully right is specified in bytes #6-8, the minimum pan value representing a signal panned fully left is the negative of this value.

The following 3 values are only present if volume control exists:
Bytes #11-13: minimum value of volume control (16 bit value encoded in 3 bytes)
Bytes #14-16: maximum value of volume control (16 bit value encoded in 3 bytes)
Bytes #17-19: resolution of volume control (16 bit value encoded in 3 bytes)

The volume values are 16 bit 2's complement numbers that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). Resolution can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF). In addition, code 0x8000, representing silence (i.e., -∞ dB), must always be implemented. However, it must never be reported as the minimum value.

Note that for mixer inputs the solo control is a pushbutton (similar to mute), whereas for mixer outputs the solo control is a fader (similar to volume).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x59                      // command flags and ID
0x00, 0x15                      // data length (21)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x7F                            // all controls exist
0x7F                            // all controls are editable
0x00, 0x00, 0x7F                // maximum value of pan control (127)
0x03                            // number of pan curve laws that follow (3)
0x01                            // pan curve law #1: 0 dB at center
0x03                            // pan curve law #2: -3 dB at center
0x05                            // pan curve law #3: -6 dB at center
0x02, 0x40, 0x00                // minimum value of volume control (-96 dB)
```

```
0x00, 0x14, 0x00          // maximum value of volume control (+6 dB)
0x00, 0x00, 0x40          // resolution value of volume control (0.25 dB)
0xxx                      // checksum
0xF7                      // footer
```

5.11. GetMixerOutputControl (Command ID = 0x5A)

This command is used to query a device about mixer output controls for a specific audio port. Devices that support this command will respond with a RetMixerOutputControl message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x5A.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x5A                     // command flags and ID
0x00, 0x02                     // data length
0x00, 0x01                     // audio port ID (1)
0xxx                           // checksum
0xF7                           // footer
```

5.12. RetMixerOutputControl (Command ID = 0x5B)

RetMixerOutputControl is sent by devices in response to a GetMixerOutputControl command.

Command Data

Command flags are Answer.
Command ID is 0x5B.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).

Byte #4: exist flags:
  bit 7 = always zero
  bit 6 = set if pan control exists
  bit 5 = set if invert control exists

bit 4 = set if stereo link control exists
bit 3 = set if solo PFL control exists
bit 2 = set if solo control exists
bit 1 = set if mute control exists
bit 0 = set if volume control exists

Byte #5: edit flags:
bit 7 = always zero
bit 6 = set if pan control is editable
bit 5 = set if invert control is editable
bit 4 = set if stereo link control is editable
bit 3 = set if solo PFL control is editable
bit 2 = set if solo control is editable
bit 1 = set if mute control is editable
bit 0 = set if volume control is editable

The following 3 values are only present if pan control exists:
Bytes #6-8: maximum value of pan control representing a signal panned fully right (a positive 16 bit 2's complement value encoded in 3 bytes), the negative of this value represents a signal panned fully left
Byte #9: number of pan curve laws that follow
Bytes #10-N: pan curve law (only included if byte #9 is not zero):

Value	Description
1	0 dB at center
2	-1.5 dB at center
3	-3 dB at center
4	-4.5 dB at center
5	-6 dB at center

The pan values are 16 bit 2's complement numbers that can range from +32767 (0x7FFF) down to -32767 (0x8001). Positive numbers are panning to the right. Negative numbers are panning to the left. 0 means fully centered. The maximum pan value representing a signal panned fully right is specified in bytes #6-8, the minimum pan value representing a signal panned fully left is the negative of this value.

The following 3 values are only present if volume or solo control exists:
Bytes #11-13: minimum value of volume or solo control (16 bit value encoded in 3 bytes)
Bytes #14-16: maximum value of volume or solo control (16 bit value encoded in 3 bytes)
Bytes #17-19: resolution of volume or solo control (16 bit value encoded in 3 bytes)

The volume and solo values are 16 bit 2's complement numbers that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001). Resolution can only have positive values and range from 1/256 dB (0x0001) to +127.9961 dB (0x7FFF). In addition, code 0x8000, representing silence (i.e., -∞ dB), must always be implemented. However, it must never be reported as the minimum value.

Note that for mixer inputs the solo control is a pushbutton (similar to mute), whereas for mixer outputs the solo control is a fader (similar to volume).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x00, 0x5B                     // command flags and ID
0x00, 0x15                     // data length (21)
0x01                           // command version number (1)
0x00, 0x01                     // audio port ID (1)
0x7F                           // all controls exist
0x7F                           // all controls are editable
0x00, 0x00, 0x7F               // maximum value of pan control (127)
0x03                           // number of pan curve laws that follow (3)
0x01                           // pan curve law #1: 0 dB at center
0x03                           // pan curve law #2: -3 dB at center
0x05                           // pan curve law #3: -6 dB at center
0x02, 0x40, 0x00               // minimum value of volume/solo control (-96 dB)
0x00, 0x14, 0x00               // maximum value of volume/solo control (+6 dB)
0x00, 0x00, 0x40               // resolution of volume/solo control (0.25 dB)
0xxx                           // checksum
0xF7                           // footer
```

5.13. GetMixerInputControlValue (Command ID = 0x5C)

This command is used to query a device about mixer input control values for a specific audio port. Devices that support this command will respond with a RetMixerInputControlValue message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x5C.
Data length is 4.

Bytes #1-2: audio port ID (1 - N).
Byte #3: mixer output number (1 - N).
Byte #4: mixer input number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x5C                     // command flags and ID
0x00, 0x04                     // data length
0x00, 0x01                     // audio port ID (1)
0x02                           // mixer output number (2)
0x01                           // mixer input number (1)
0xxx                           // checksum
0xF7                           // footer
```

5.14. RetMixerInputControlValue / SetMixerInputControlValue (Command ID = 0x5D)

RetMixerInputControlValue is sent by devices in response to a GetMixerInputControlValue command.

SetMixerInputControlValue is sent by a host to a device to set the mixer input control values for a specific audio port. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMixerInputControlValue, Write for SetMixerInputControlValue.
Command ID is 0x5D.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: mixer output number (1 - N)
Byte #5: mixer input number (1 - N)

Byte #6: value flags:
bit 7 = always zero
bit 6 = set if pan control and pan curve law values are included
bit 5 = set if invert control value is included
bit 4 = set if stereo link control value is included
bit 3 = set if solo PFL control value is included
bit 2 = set if solo control value is included
bit 1 = set if mute control value is included
bit 0 = set if volume control value is included

Values follow for each of the flags in byte #6 (volume first, mute second, etc):

Bytes #7-9 [W]: value of volume control (16 bit value encoded in 3 bytes)
Byte #10 [W]: value of mute control (0 = mute off, 1 = mute on)
Byte #11 [W]: value of solo control (0 = solo off, 1 = solo on)
Byte #12 [W]: value of solo PFL control (0 = solo PFL off, 1 = solo PFL on)
Byte #13 [W]: value of stereo link control (0 = link off, 1 = link on)
Byte #14 [W]: value of invert control (0 = invert off, 1 = invert on)
Bytes #15-17 [W]: value of pan control (16 bit value encoded in 3 bytes)
Byte #18 [W]: pan curve law

The volume values are 16 bit 2's complement numbers that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001).

The pan values are 16 bit 2's complement numbers that can range from +32767 (0x7FFF) down to -32767 (0x8001). Positive numbers are panning to the right. Negative numbers are panning to the left. 0 means fully centered.

Note that for mixer inputs the solo control is a pushbutton (similar to mute), whereas for mixer outputs the solo control is a fader (similar to volume).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x5D                      // command flags and ID
0x00, 0x0F                      // data length (15)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x02                            // mixer output number (2)
0x01                            // mixer input number (1)
0x47                            // pan, solo, mute and volume are included
0x00, 0x08, 0x00                // volume control value (4 dB = 0x0400)
0x00                            // mute is off (0)
0x01                            // solo is on (1)
0x00, 0x00, 0x00                // pan control is centered (0x0000)
0x03                            // pan curve law (-3 dB at center)
0xxx                            // checksum
0xF7                            // footer
```

5.15. GetMixerOutputControlValue (Command ID = 0x5E)

This command is used to query a device about mixer output control values for a specific audio port. Devices that support this command will respond with a RetMixerOutputControlValue message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x5E.
Data length is 3.

Bytes #1-2: audio port ID (1 - N).
Byte #3: mixer output number (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x5E                      // command flags and ID
0x00, 0x03                      // data length
0x00, 0x01                      // audio port ID (1)
0x01                            // mixer output number (1)
0xxx                            // checksum
0xF7                            // footer
```

5.16. RetMixerOutputControlValue / SetMixerOutputControlValue (Command ID = 0x5F)

RetMixerOutputControlValue is sent by devices in response to a GetMixerOutputControlValue command.

SetMixerOutputControlValue is sent by a host to a device to set the mixer output control values for a specific audio port. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetMixerOutputControlValue, Write for SetMixerOutputControlValue.
Command ID is 0x5F.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: mixer output number (1 - N)

Byte #5: value flags:
bit 7 = always zero
bit 6 = set if pan control and pan curve law values are included
bit 5 = set if invert control value is included
bit 4 = set if stereo link control value is included
bit 3 = set if solo PFL control value is included
bit 2 = set if solo control value is included
bit 1 = set if mute control value is included
bit 0 = set if volume control value is included

Values follow for each of the flags in byte #5 (volume first, mute second, etc):

Bytes #6-8 [W]: value of volume control (16 bit value encoded in 3 bytes)
Byte #9 [W]: value of mute control (0 = mute off, 1 = mute on)
Bytes #10-12 [W]: value of solo control (16 bit value encoded in 3 bytes)
Byte #13 [W]: value of solo PFL control (0 = solo PFL off, 1 = solo PFL on)
Byte #14 [W]: value of stereo link control (0 = link off, 1 = link on)
Byte #15 [W]: value of invert control (0 = invert off, 1 = invert on)
Bytes #16-18 [W]: value of pan control (16 bit value encoded in 3 bytes)
Byte #19 [W]: pan curve law

The volume and solo values are 16 bit 2's complement numbers that can range from +127.9961 dB (0x7FFF) down to -127.9961 dB (0x8001) in steps of 1/256 dB or 0.00390625 dB (0x0001).

The pan values are 16 bit 2's complement numbers that can range from +32767 (0x7FFF) down to -32767 (0x8001). Positive numbers are panning to the right. Negative numbers are panning to the left. 0 means fully centered.

Note that for mixer inputs the solo control is a pushbutton (similar to mute), whereas for mixer outputs the solo control is a fader (similar to volume).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
```

```
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x5F                      // command flags and ID
0x00, 0x0E                      // data length (14)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x01                            // mixer output number (1)
0x37                            // invert, link, solo, mute and volume are included
0x00, 0x08, 0x00                // volume control value (4 dB = 0x0400)
0x00                            // mute is off (0)
0x03, 0x78, 0x00                // solo control value (-4 dB = 0xFC00)
0x00                            // link is off (0)
0x01                            // invert is on (1)
0xxx                            // checksum
0xF7                            // footer
```

5.17. GetMixerMeterValue (Command ID = 0x60)

This command is used to query a device for the most recent meter values for a specific mixer. Devices that support this command will respond with a RetMixerMeterValue message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x60.
Data length is 4.

Bytes #1-2: audio port ID (1 - N).
Byte #3: mixer output number (1 - N).
Byte #4: bitmap indicating which meter values should be returned from the device:

Bit	Description
7 - 2	always zero
1	mixer output channel
0	mixer input channels

A mixer always has one output but the number of inputs may be 0, 1, or some other value. For mixers configured as stereo pairs it is necessary to send two of these commands: one for the left channel (N, where N is an odd number) and one for the right channel (N+1).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
```

```
0x40, 0x60          // command flags and ID
0x00, 0x04          // data length
0x00, 0x01          // audio port ID (1)
0x02                // mixer output number (2)
0x03                // return meter values for mixer inputs and mixer output
0xxx                // checksum
0xF7                // footer
```

5.18. RetMixerMeterValue (Command ID = 0x61)

RetMixerMeterValue is sent by devices in response to a GetMixerMeterValue command.

Command Data

Command flags are Answer.
Command ID is 0x61.
Data length depends on the number of mixer channels.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).
Byte #4: mixer output number (1 - N).
Byte #5: number of audio meter blocks that follow.
Bytes #6-N: audio meter blocks:

Byte	Description
1	Bitmap indicating which meter values are in this block: bits 7 - 2: always zero bit 1: mixer output channel bit 0: mixer input channels
2	Number of meter values that follow.
3 - 4 (channel 1), 5 - 6 (channel 2), 7 - 8 (channel 3), etc.	Meter values. Each meter value is a 14 bit big-endian number split across 2 bytes. 0 dB is 8192 (0x2000). The maximum value is 16383 (0x3FFF) or +6 dB. For inputs, the first value is mixer input 1, the second value is mixer input 2, etc. For outputs there is only ever one value. To convert to decibels (dB) use the formula: dB = 20 log (value/8192) Note that the minimum meter value (1) corresponds to -78.27 dB but resolution drops to 1 dB when meter value is 8 (-60 dB).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
```

```
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x61                      // command flags and ID
0x00, 0x0F                      // data length (15)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID (1)
0x02                            // mixer output number (2)
0x02                            // number of audio meter blocks that follow (2)
0x01                            // block #1: mixer inputs (0x01)
0x02                            // meter readings that follow (2)
0x00, 0x00                      // mixer input 1: meter value = 0x0000
0x20, 0x00                      // mixer input 2: meter value = 0x1000
0x02                            // block #2: mixer output (0x02)
0x04                            // meter readings that follow (1)
0x02, 0x00                      // mixer output: meter value = 0x0100
0xxx                            // checksum
0xF7                            // footer
```

6. Audio Commands V1 [deprecated]

The following commands are defined for protocol version = 1.

6.1. GetAudioInfo (Command ID = 0x30)

This command is used to query a device about audio parameters. Devices that support this command will respond with a RetAudioInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x30.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x30                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

6.2. RetAudioInfo (Command ID = 0x31)

RetAudioInfo is sent by devices in response to a GetAudioInfo command. It contains basic information that a host will need to use to communicate with this device for all other audio related messages. A host should cache this information and use it for all further communication with this device.

Command Data

Command flags are Answer.
Command ID is 0x31.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: number of audio ports supported by this device (0 - N).
Byte# 4: number of audio capable USB device jacks supported by this device (0 - N).
Byte# 5: number of audio capable USB host jacks supported by this device (0 - N).
Byte# 6: number of audio capable ethernet jacks supported by this device (0 - N).
Byte# 7: number of audio ports supported by each USB host jack (0 - N).
Byte# 8: number of audio ports supported by each ethernet jack (0 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x31                      // command flags and ID
0x00, 0x08                      // data length (8)
0x01                            // command version number (1)
0x00, 0x09                      // device has 9 audio capable ports
0x03                            // device has 3 audio capable USB device jacks
0x01                            // device has 1 audio capable USB host jack
0x01                            // device has 1 audio capable ethernet jack
0x02                            // device supports 2 audio ports on each USB host jack
0x04                            // device supports 4 audio ports on each ethernet jack
0xxx                            // checksum
0xF7                            // footer
```

6.3. GetAudioCfgInfo (Command ID = 0x32)

This command is used to query a device about audio configurations. Devices that support this command will respond with a RetAudioCfgInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x32.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E   // header
0x00, 0x05                     // product ID
0x01, 0x02, 0x03, 0x04, 0x05   // serial number
0x00, 0x00                     // transaction ID
0x40, 0x32                     // command flags and ID
0x00, 0x00                     // data length
0xxx                           // checksum
0xF7                           // footer
```

6.4. RetAudioCfgInfo / SetAudioCfgInfo (Command ID = 0x33)

RetAudioCfgInfo is sent by devices in response to a GetAudioCfgInfo command. It contains information that a host will need to use to communicate with this device for other audio related messages. A host should cache this information and use it for further communication with this device.

SetAudioCfgInfo is sent by a host to a device to set the current configuration. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioCfgInfo, Write for SetAudioCfgInfo.
Command ID is 0x33.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) byte #8, all bytes beyond #8 are read-only and are ignored.

Byte #2: minimum number of audio frames that can be buffered (1 - N).
Byte #3: maximum number of audio frames that can be buffered (1 - N).
Byte #4 [W]: current number of audio frames to buffer (1 - N).
Byte #5: minimum allowed value for sync factor (1 - N).
Byte #6: maximum allowed value for sync factor (1 - N).
Byte #7 [W]: current sync factor value (1 - N).
Byte #8 [W]: number of the currently active configuration (1 - N).
Byte #9: number of configuration blocks that follow (0 - N).

Bytes #10-N: configurations blocks. Bytes 10 and up are used only if byte #9 is not zero. Four bytes are used for each configuration block as follows:

Byte	Description
1	configuration number (1 - N)
2	bit depth code: 1 = 4 bit, 2 = 8 bit, 3 = 12 bit, 4 = 16 bit bit depth code: 5 = 20 bit, 6 = 24 bit, 7 = 28 bit, 8 = 32 bit

Byte	Description
3	sample rate code: 1 = 11025, 3 = 22050, 5 = 44100, 7 = 88200 sample rate code: 2 = 12000, 4 = 24000, 6 = 48000, 8 = 96000
4	number of audio channels (0 - N)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x33                      // command flags and ID
0x00, 0x29                      // data length (41)
0x01                            // command version number (1)
0x01, 0x04, 0x02                // minimum, maximum, and current # of audio frames to buffer
0x01, 0x04, 0x02                // minimum, maximum, and current sync factor
0x02                            // currently active configuration is #2
0x08                            // number of configuration blocks that follow (8)
0x01, 0x04, 0x05, 0x04          // configuration block #1: 16 bit, 44100, 4 channels
0x02, 0x04, 0x05, 0x08          // configuration block #2: 16 bit, 44100, 8 channels
0x03, 0x04, 0x06, 0x04          // configuration block #3: 16 bit, 48000, 4 channels
0x04, 0x04, 0x06, 0x08          // configuration block #4: 16 bit, 48000, 8 channels
0x05, 0x04, 0x06, 0x04          // configuration block #5: 16 bit, 88200, 4 channels
0x06, 0x04, 0x08, 0x04          // configuration block #6: 16 bit, 96000, 4 channels
0x07, 0x06, 0x05, 0x04          // configuration block #7: 24 bit, 44100, 4 channels
0x08, 0x06, 0x06, 0x04          // configuration block #8: 24 bit, 48000, 4 channels
0xxx                            // checksum
0xF7                            // footer
```

6.5. GetAudioPortInfo (Command ID = 0x34)

This command is used to query a device about a specific audio port. Devices that support this command will respond with a RetAudioPortInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x34.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
```

```
0x40, 0x34          // command flags and ID
0x00, 0x02          // data length
0x00, 0x01          // audio port ID
0xxx                // checksum
0xF7                // footer
```

6.6. RetAudioPortInfo / SetAudioPortInfo (Command ID = 0x35)

RetAudioPortInfo is sent by devices in response to a GetAudioPortInfo command.

SetAudioPortInfo is sent by a host to a device to set the values of some of the port parameters. All parameters must be sent in the message but only the writeable parameters are changed in the device. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioPortInfo, Write for SetAudioPortInfo.
Command ID is 0x35.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

For version = 1, minimum length is 13, actual length depends on the port name (a 7-bit ASCII string, not NULL terminated). Use the data length field to determine the string length.

Bytes #2-3: audio port ID (1 - N).

Byte #4: port type:

Value	Description
2	USB device
3	USB host
4	ethernet

Bytes #5-8: port info, depends on port type (byte #4):

Type	Description
USB device	Byte #5: USB device jack # (1 - N) Byte #6-8: reserved (always 0)

Type	Description
USB host	Byte #5: USB host jack # (1 - N) Byte #6: host port # (1 - N) Byte #7-8: reserved (always 0)
ethernet	Byte #5: ethernet jack # (1 - N) Byte #6: port # (1 - N) Byte #7-8: reserved (always 0)

Byte #9: maximum length allowed for port name, 0 if read-only (0 - 127).

Bytes #10-13: port specific options, format of these bytes depends on port type (byte #4). For USB device ports the four bytes represent a bitmap as follows

Bit	Description
31 - 4	reserved (always zero)
3 [W]	set if port is currently enabled for audio with iOS devices
2 [W]	set if port is currently enabled for audio with PC/Mac
1	set if port supports audio with iOS devices
0	set if port supports audio with PC/Mac

Bytes #14-N [W]: port name, 7-bit ASCII string, not NULL terminated.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x35                      // command flags and ID
0x00, 0x11                      // data length (17)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID
0x02                            // port type = USB device
0x01                            // port is first USB device jack
0x00, 0x00, 0x00                // reserved for USB device port
0x04                            // maximum length allowed for port name (4 ASCII characters)
0x00, 0x00, 0x00, 0x07          // port supports PC and iOS audio but iOS audio is currently disabled
0x55, 0x53, 0x42, 0x31          // port name, the ASCII string “USB1”
0xxx                            // checksum
0xF7                            // footer
```

6.7. GetAudioPortCfgInfo (Command ID = 0x36)

This command is used to query a device about audio port configurations. Devices that support this command will respond with a RetAudioPortCfgInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x36.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x36                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID
0xxx                            // checksum
0xF7                            // footer
```

6.8. RetAudioPortCfgInfo / SetAudioPortCfgInfo (Command ID = 0x37)

RetAudioPortCfgInfo is sent by devices in response to a GetAudioPortCfgInfo command. It contains information related to the number of audio channels that this port supports for each of the different audio configurations (from RetAudioCfgInfo command).

SetAudioPortCfgInfo is sent by a host to a device to set the current port configuration. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored.

Command Data

Command flags are Answer for RetAudioPortCfgInfo, Write for SetAudioPortCfgInfo.
Command ID is 0x37.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

The host only needs to send up to (and including) byte #5, all bytes beyond #5 are read-only and are ignored.

Bytes #2-3: audio port ID (1 - N).

Byte #4 [W]: number of input channels to use for this port (0 - N). Must be a legal value for the current audio configuration.

Byte #5 [W]: number of output channels to use for this port (0 - N). Must be a legal value for the current audio configuration.
Byte #6: number of port configuration blocks that follow (0 - N).

Bytes #7-N: port configurations blocks. Bytes 7 and up are used only if byte #6 is not zero. There is one port configuration block for each audio configuration (from RetAudioCfgInfo command). Five bytes are used for each port configuration block as follows:

Byte	Description
1	audio configuration number (1 - N, from RetAudioCfgInfo command)
2	minimum number of input channels (0 - N)
3	maximum number of input channels (0 - N)
4	minimum number of output channels (0 - N)
5	maximum number of output channels (0 - N)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x37                      // command flags and ID
0x00, 0x2E                      // data length (46)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID
0x04                            // number of input channels to use for this port (4)
0x04                            // number of output channels to use for this port (4)
0x08                            // number of port configuration blocks that follow (8)
0x01, 0x01, 0x04, 0x00, 0x04    // port configuration block #1 (16/44/4): 1/4 in, 0/4 out
0x02, 0x01, 0x08, 0x00, 0x08    // port configuration block #2 (16/44/8): 1/8 in, 0/8 out
0x03, 0x01, 0x04, 0x00, 0x04    // port configuration block #3 (16/48/4): 1/4 in, 0/4 out
0x04, 0x01, 0x08, 0x00, 0x08    // port configuration block #4 (16/48/8): 1/8 in, 0/8 out
0x05, 0x01, 0x02, 0x00, 0x02    // port configuration block #5 (16/88/4): 1/2 in, 0/2 out
0x06, 0x01, 0x02, 0x00, 0x02    // port configuration block #6 (16/96/4): 1/2 in, 0/2 out
0x07, 0x01, 0x04, 0x00, 0x04    // port configuration block #7 (24/44/4): 1/4 in, 0/4 out
0x08, 0x01, 0x04, 0x00, 0x04    // port configuration block #8 (24/48/2): 1/4 in, 0/4 out
0xxx                            // checksum
0xF7                            // footer
```

6.9. GetAudioPortPatchbay (Command ID = 0x38)

This command is used to query a device about audio port patchbay setup. Devices that support this command will respond with a RetAudioPortPatchbay message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x38.
Data length is 2.

Bytes #1-2: audio port ID (1 - N).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x38                      // command flags and ID
0x00, 0x02                      // data length
0x00, 0x01                      // audio port ID
0xxx                            // checksum
0xF7                            // footer
```

6.10. RetAudioPortPatchbay / SetAudioPortPatchbay (Command ID = 0x39)

RetAudioPortPatchbay is sent by devices in response to a GetAudioPortPatchbay command. It contains the patchbay setup for a specific audio port.

SetAudioPortCfgInfo is sent by a host to a device to configure the patchbay for a specific audio port.

Command Data

Command flags are Answer for RetAudioPortPatchbay, Write for SetAudioPortPatchbay.
Command ID is 0x39.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Bytes #2-3: audio port ID (1 - N).

Byte #4: number of patchbay configuration blocks that follow (0 - N).

Bytes #5-N: patchbay configurations blocks. Bytes 5 and up are used only if byte #4 is not zero. If byte #4 is not zero, there should be one patchbay configuration block for each input channel for this port. Four bytes are used for each port configuration block as follows:

Byte	Description
1	input channel number (1 - N)
2	output channel number of the port driving this input (1 - N)

Byte	Description
3 - 4	port ID of output port driving this input

All values in each patch configuration block must be valid for the output port ID, output channel number, and input channel number. Use 0x00 for bytes #2-4 to indicate “no connection” (i.e. no output is driving the input).

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x39                      // command flags and ID
0x00, 0x14                      // data length (20)
0x01                            // command version number (1)
0x00, 0x01                      // audio port ID
0x04                            // number of patchbay configuration blocks that follow (4)
0x01, 0x03, 0x00, 0x02          // patchbay block #1: input #1 is connected to output channel 3 of port #2
0x02, 0x01, 0x00, 0x03          // patchbay block #2: input #2 is connected to output channel 1 of port #3
0x03, 0x02, 0x00, 0x03          // patchbay block #3: input #3 is connected to output channel 2 of port #3
0x04, 0x00, 0x00, 0x00          // patchbay block #4: input #4 is not connected to anything
0xxx                            // checksum
0xF7                            // footer
```

6.11. GetAudioClockInfo (Command ID = 0x3A)

This command is used to query a device about audio clock sources. Devices that support this command will respond with a RetAudioClockInfo message. If this command is not supported the device will respond with an ACK message (with error code).

Command Data

Command flags are Query.
Command ID is 0x3A.
Data length is 0.

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x40, 0x3A                      // command flags and ID
0x00, 0x00                      // data length
0xxx                            // checksum
0xF7                            // footer
```

6.12. RetAudioClockInfo / SetAudioClockInfo (Command ID = 0x3B)

RetAudioClockInfo is sent by devices in response to a GetAudioClockInfo command. It contains information regarding audio clock sources for the device.

SetAudioClockInfo is sent by a host to a device to set the audio clock source. Writeable parameters are indicated below with a [W]. Read-only parameters can be set to any value, they will be ignored. The host only needs to send up to (and including) byte #2, all bytes beyond #2 are read-only and are ignored.

Command Data

Command flags are Answer for RetAudioClockInfo, Write for SetAudioClockInfo.
Command ID is 0x3B.
Data length depends on command version number.

Byte #1: command version number that this device supports (1 - 127). If the host does not understand the command version number then it should not attempt any further communication with the device.

For command version number = 1:

Byte #2 [W]: number of the active audio clock source block (1 - N).
Byte #3: number of audio clock source blocks that follow (0 - N).
Bytes #4-N: audio clock source blocks. Bytes 4 and up are used only if byte #3 is not zero. Four bytes are used for each audio clock source block as follows:

Byte	Description
1	audio clock source number (1 - N)
2	clock source type: 1 = internal clock 2 = audio port clock
3 - 4	For clock source type = internal clock: Byte 3: internal clock ID (1 - N) Byte 4: reserved (always 0) For clock source type = audio port clock: Byte 3: audio port ID (1 - N) Byte 4: reserved (always 0)

Example:

```
0xF0, 0x00, 0x01, 0x73, 0x7E    // header
0x00, 0x05                      // product ID
0x01, 0x02, 0x03, 0x04, 0x05    // serial number
0x00, 0x00                      // transaction ID
0x00, 0x3B                      // command flags and ID
0x00, 0x0F                      // data length (15)
0x01                            // command version number (1)
0x02                            // active audio clock source block is #2
```

```
0x03                    // number of audio clock blocks that follow (3)
0x01, 0x01, 0x01, 0x00  // audio clock source block #1: internal clock #1
0x02, 0x02, 0x01, 0x00  // audio clock source block #2: USB audio port #1
0x03, 0x02, 0x02, 0x00  // audio clock source block #3: USB audio port #2
0xxx                    // checksum
0xF7                    // footer
```

7. History

#	Date	Author	Description
14	15-05-27	J. Cain	Reformatting from original document.