Audiolino MIDI SYSEX Specification

Document: RM00001

Revision: a

Document revision history

Revision	Notes
a	First document revision

SYSEX format

Audiolino SYSEX message is made by:

- an header
- an optional payload
- a footer

Header

Header is made of 8 bytes

Byte	Name	Description
1	BOS	Begin of SYSEX (0xF0)
2	MID0	Manufacturer ID0 (0x00)
3	MID1	Manufacturer ID1 (0x21)
4	MID2	Manufacturer ID2 (0x44)
5	DID	Device ID
6	CH	Channel
7	STA	Status
8	CMD	Command

DID (Device ID)

0x01 = Brick

0x7F = All devices

CH (Channel)

0x00 .. 0x0F = MIDI channel (if it applies)

0x7F = All channels

STA (Status)

0x00 = Event

0x10 = Command without ACK

0x11 = Command with ACK

0x20 = Response ACK

0x21 = Response NACK

An event is an asynchronous event that may be triggered by a device at any time. A command is a specific request sent to a device. This can be made with or without ACK. In order to get any response from the device or to be sure that the command is executed, the command must be sent with ACK (*Status = 0x11*).

CMD (Command)

Command value	Mnemonic	Description
0x00	ID	Get device ID
0x01	RESET	Execute device reset
0x02	WHO	Get if appliation or bootloader is running
0x03	SERIAL	Get serial number
0x04	SUPPORTED_CMD	Get bitmask of supported commands
0x05	VER	Get application and bootloader versions
0x06	DISCOVERY	Discover devices
0x07	FACTORY_RESET	Restore factory setting
0x08	STORE_DEV_PARAMS	Store device parameters
0x09	LOAD_PROGRAM	Load internal program
0x0A	STORE_PROGRAM	Store internal program
0x0B	DEV_PARAM_GET	Get device parameter
0x0C	DEV_PARAM_SET	Set device parameter
0x0D	EDIT_PARAM_GET	Get edit device parameter
0x0E	EDIT_PARAM_SET	Set edit device parameter
0x0F	ALG_PARAM_GET	Get algorithm device parameter
0x10	ALG_PARAM_SET	Set algorithm device parameter
0x16	DUMP_INFO	Get dump area informations
0x17	DUMP_ERASE	Erase dump area
0x18	DUMP_WRITE	Write dump area packet
0x19	DUMP_READ	Read dump area packet
0x1A	DUMP_CRC	Dump area CRC
0x1B	FW_UPGRADE_ERASE	Erase firmware area
0x1C	FW_UPGRADE_WRITE	Write firmware area packet
0x70	LOG	Log message (event only)
0x71	CHAIN_INTERNAL	Internal command used in chain mode
0x72	LOOPBACK	Loopback communication test
0x7F	STATUS	Get device status

All unlisted command codes are reserved for future use. Each device or specific firmware may have restrictions. See appendix for details.

Payload

Commands and responses may have a payload depending on command code. A detailed list will be given later in this document.

Footer

Footer is made of 2 bytes:

Byte	Name	Description
1	CKS	Checksum
2	EOS	End of SYSEX (0xF7)

CKS (Checksum)

Checksum is calculated as bitwise XOR of all SYSEX header bytes excluding the leading 0xF0 and all payload bytes. This means that XOR of all SYSEX bytes excluding BOS (0xF0) and EOS (0xF7) must be zero.

SYSEX payloads

Command 0x00 (ID)

TX payload

none

RX payload

Byte	Name	Description
1	PROD_ID	Product ID
2	BOARD_REV	Board revision

Product ID:

0x01 = Audiolino Brick

Command 0x01 (RESET)

TX payload

Byte	Name	Description
1	MODE	Restart mode (0=app, 1=bootloader)

RX payload

none

Command 0x02 (WHO)

TX payload

none

RX payload

Byte	Name	Description
1	WHO	Who (bit mask)

Who bit mask

0x01b (bootloader supported) : 0 = no, 1 = yes

0x02b (bootloader running) : 0 = no, 1 = yes

Command 0x03 (SERIAL)

TX payload

none

RX payload Serial number is made of up to 32 characters, but it is usually smaller in size.

Byte	Name	Description
1	SN[0]	Serial number 1st byte
...		
32	SN[31]	Serial number 32th byte

Command 0x04 (SUPPORTED_CMD)

TX payload

none

RX payload Each byte contains 7 bit of information. If a bit is set, then the corresponding command is supported by the device. For example, if DATA[0] = 0x3F, then commands with number 0 (ID) to 5 (VER) will be supported.

Byte	Name	Description
1	SUPP[0]	Supported commands bit mask 1st byte
...		
19	SUPP[19]	Supported commands bit mask 19th byte

Command 0x05 (VER)

TX payload

none

RX payload This command can be used to get application and bootloader versions.

Byte	Name	Description
1	APP_ID0	App ID0
2	APP_ID1	App ID1
3	APP_MAJ	Application major
4	APP_MIN	Application minor
5	APP_REV	Application revision
6	APP_RC	Application RC (0=production)
7	APP_CFG	Application configuration (usually 'r')
8	BL_MAJ	Bootloader major
9	BL_MIN	Bootloader minor
10	BL_REV	Bootloader revision
11	BL_RC	Bootloader RC (0=production)
12	BL_CFG	Bootloader configuration (usually 'r')

App ID0 and *App ID1* are used to identify a particular type of firmware that is running on the machine.

Command 0x06 (DISCOVERY)

TX payload

none

RX payload

Byte	Name	Description
1	POS	Position in Brick chain (0 = unknown)
2	DID	Device ID
3	CH	Channel

Command 0x07 (FACTORY_RESET)

TX payload

none

RX payload

none

NOTE This command may require a certain amount of time to be executed. To avoid problems always send it with *STATUS* set to 0x11 (command with ACK) and wait the device response, that may be ACK or NACK.

Command 0x08 (STORE_DEV_PARAMS)

TX payload

none

RX payload

none

NOTE This command may require a certain amount of time to be executed. To avoid problems always send it with *STATUS* set to 0x11 (command with ACK) and wait the device response, that may be ACK or NACK.

Command 0x09 (LOAD_PROGRAM)

TX payload

Byte	Name	Description
1	AREA_ID	Memory area ID
2	INDEX	Program index

Memory area ID: For Audiolino Brick this must be set to 0.

Program index: For Audiolino Brick this must be a number from 0 to 31.

RX payload

none

NOTE This command may require a certain amount of time to be executed. To avoid problems always send it with *STATUS* set to 0x11 (command with ACK) and wait the device response, that may be ACK or NACK.

Command 0x0A (STORE_PROGRAM)

TX payload

Byte	Name	Description
1	AREA_ID	Memory area ID
2	INDEX	Program index

Memory area ID: For Audiolino Brick this must be set to 0.

Program index: For Audiolino Brick this must be a number from 0 to 31.

RX payload

none

NOTE This command may require a certain amount of time to be executed. To avoid problems always send it with *STATUS* set to 0x11 (command with ACK) and wait the device response, that may be ACK or NACK.

Command 0x0B (DEV_PARAM_GET)

Each Audiolino device may have a certain number of device specific parameters. With this command you can read the value of one of this parameters. Parameter address and values are 16bit numbers.

TX payload

Byte	Name	Description
1	AD[0]	Parameter address bit0 to bit6
2	AD[1]	Parameter address bit7 to bit13
3	AD[2]	Parameter address bit14 to bit15

RX payload

Byte	Name	Description
1	AD[0]	Parameter address bit0 to bit6
2	AD[1]	Parameter address bit7 to bit13
3	AD[2]	Parameter address bit14 to bit15
4	VAL[0]	Parameter value bit0 to bit6
5	VAL[1]	Parameter value bit7 to bit13
6	VAL[2]	Parameter value bit14 to bit15

Command 0x0C (DEV_PARAM_SET)

Each Audiolino device may have a certain number of device specific parameters. With this command you can write the value of one of this parameters. Parameter address and values are 16bit numbers.

TX payload

Byte	Name	Description
1	AD[0]	Parameter address bit0 to bit6
2	AD[1]	Parameter address bit7 to bit13
3	AD[2]	Parameter address bit14 to bit15
4	VAL[0]	Parameter value bit0 to bit6
5	VAL[1]	Parameter value bit7 to bit13
6	VAL[2]	Parameter value bit14 to bit15

RX payload

Byte	Name	Description
1	AD[0]	Parameter address bit0 to bit6
2	AD[1]	Parameter address bit7 to bit13
3	AD[2]	Parameter address bit14 to bit15
4	VAL[0]	Parameter value bit0 to bit6

Byte	Name	Description
5	VAL[1]	Parameter value bit7 to bit13
6	VAL[2]	Parameter value bit14 to bit15

Command 0x0D (PROGRAM_PARAM_GET)

Each program may have a certain number of parameters. With this command you can read the value of one of this parameters. Parameter address and values are 16bit numbers.

TX payload

Byte	Name	Description
1	AD[0]	Parameter address bit0 to bit6
2	AD[1]	Parameter address bit7 to bit13
3	AD[2]	Parameter address bit14 to bit15

RX payload

Byte	Name	Description
1	AD[0]	Parameter address bit0 to bit6
2	AD[1]	Parameter address bit7 to bit13
3	AD[2]	Parameter address bit14 to bit15
4	VAL[0]	Parameter value bit0 to bit6
5	VAL[1]	Parameter value bit7 to bit13
6	VAL[2]	Parameter value bit14 to bit15

Command 0x0E (PROGRAM_PARAM_SET)

Each program may have a certain number of parameters. With this command you can write the value of one of this parameters. Parameter address and values are 16bit numbers.

TX payload

Byte	Name	Description
1	AD[0]	Parameter address bit0 to bit6
2	AD[1]	Parameter address bit7 to bit13
3	AD[2]	Parameter address bit14 to bit15
4	VAL[0]	Parameter value bit0 to bit6
5	VAL[1]	Parameter value bit7 to bit13
6	VAL[2]	Parameter value bit14 to bit15

RX payload

Byte	Name	Description
1	AD[0]	Parameter address bit0 to bit6

Byte	Name	Description
2	AD[1]	Parameter address bit7 to bit13
3	AD[2]	Parameter address bit14 to bit15
4	VAL[0]	Parameter value bit0 to bit6
5	VAL[1]	Parameter value bit7 to bit13
6	VAL[2]	Parameter value bit14 to bit15

Command 0x0F (ALG_PARAM_GET)

Each algorithm may have a certain number of parameters. With this command you can read the value of one of this parameters.

TX payload

Byte	Name	Description
1	ALG_IDX	Algorithm index
2	AD[0]	Parameter address bit0 to bit6
3	AD[1]	Parameter address bit7 to bit13
4	AD[2]	Parameter address bit14 to bit15

RX payload

Byte	Name	Description
1	ALG_IDX	Algorithm index
2	AD[0]	Parameter address bit0 to bit6
3	AD[1]	Parameter address bit7 to bit13
4	AD[2]	Parameter address bit14 to bit15
5	VAL[0]	Parameter value bit0 to bit6
6	VAL[1]	Parameter value bit7 to bit13
7	VAL[2]	Parameter value bit14 to bit15

For details on *ALG_IDX*, *AD* and *VAL* fields please see ALG_PARAM_SET command notes.

Command 0x10 (ALG_PARAM_SET)

Each algorithm may have a certain number of parameters. With this command you can write the value of one of this parameters.

TX payload

Byte	Name	Description
1	ALG_IDX	Algorithm index
2	AD[0]	Parameter address bit0 to bit6
3	AD[1]	Parameter address bit7 to bit13
4	AD[2]	Parameter address bit14 to bit15
5	VAL[0]	Parameter value bit0 to bit6

Byte	Name	Description
6	VAL[1]	Parameter value bit7 to bit13
7	VAL[2]	Parameter value bit14 to bit15

RX payload

Byte	Name	Description
1	ALG_IDX	Algorithm index
2	AD[0]	Parameter address bit0 to bit6
3	AD[1]	Parameter address bit7 to bit13
4	AD[2]	Parameter address bit14 to bit15
5	VAL[0]	Parameter value bit0 to bit6
6	VAL[1]	Parameter value bit7 to bit13
7	VAL[2]	Parameter value bit14 to bit15

Parameter addresses and values are specified in device firmware specification document. Each parameter has its own id, unit, step, minimimum, maximum and default values.

ALG_IDX must be currently set to 0.

AD[2..0] must be the same as the matching parameter *Id* indicated in device algorithm parameters table.

VAL[2..0] must be calculated using this formula:

```
VAL = Target / Step
```

Step defines the resolution available on each parameter. VAL is always a 16 bit integer. Using this 16 bit representation we avoid the use of complicated hard-coded parameter tables without sacrificing resolution.

For example let's consider this parameter table:

Id	Name	Unit	Step	Min	Max	Default
...						
5	LRPhase	degree	1	0	180	90
...						

Minimum value is 0 degrees, maximum value is 180 degrees and step is 1. Example MIDI value are:

Target value	Formula	VAL
0 degrees	0 / 1	**0**
90 degrees	90 / 1	**90**
180 degrees	180 / 1	**180**

As another example:

Id	Name	Unit	Step	Min	Max	Default
...						
2	DryWet	%	0.1	0	100	100
...						

Minimum value is 0%, maximum value is 100% and step is 0.1. Example MIDI value are:

Target value	Formula	VAL
0 %	0 / 0.1	**0**
50 %	50 / 0.1	**500**
100 %	1000 / 0.1	**1000**

Last example:

Id	Name	Unit	Step	Min	Max	Default
...						
6	Predelay	ms	0.01	0	20	0
...						

Minimum value is 0ms, maximum value is 20ms and step is 0.1. Example MIDI value are:

Target value	Formula	VAL
0 ms	0 / 0.01	**0**
10 ms	10 / 0.01	**1000**
20 ms	20 / 0.01	**2000**

Command 0x16 (DUMP_INFO)

With this command you can get some informations about the selected dump area, like its name and size. Use this before starting a dump read request.

TX payload

Byte	Name	Description
1	AREA_ID	Area ID
2	INDEX	Element index

RX payload

Byte	Name	Description

Byte	Name	Description
1	AREA_ID	Area ID
2	INDEX	Element index
3	NAME[0]	Area name 1st character
...		
18	NAME[15]	Area name 16th character
19	SIZE[0]	Area size bit0 to bit6
20	SIZE[1]	Area size bit7 to bit13
21	SIZE[2]	Area size bit14 to bit20
22	SIZE[3]	Area size bit21 to bit27
23	SIZE[4]	Area size bit28 to bit31)

For Audiolino Brick both *Area ID* and *Element index* are 0.

Command 0x17 (DUMP_ERASE)

With this command you will erase the entire dump area. Use this command before a *DUMP_WRITE* sequence.

TX payload

Byte	Name	Description
1	AREA_ID	Area ID
2	INDEX	Element index

For Audiolino Brick both *Area ID* and *Element index* are 0.

RX payload

none

NOTE This command may require a certain amount of time to be executed. To avoid problems always send it with *STATUS* set to 0x11 (command with ACK) and wait the device response, that may be ACK or NACK.

Command 0x18 (DUMP_WRITE)

With this command you can write a single packet of dump area contents. Data is encoded with MIDI 7bit data encoding and the net payload of each packet is always 64 bytes. This means that sender must use data padding on the last write only, where it may have less than 64 bytes to be written to area. Preferred padding is with 0xFF data.

TX payload

Byte	Name	Description
1	AREA_ID	Area ID

Byte	Name	Description
2	INDEX	Element index
3	AD[0]	Area address bit0 to bit6
4	AD[1]	Area address bit7 to bit13
5	AD[2]	Area address bit14 to bit20
6	AD[3]	Area address bit21 to bit27
7	AD[4]	Area address bit28 to bit31)
8	DATA[0]	1st byte of 7-bit encoded data
...		
81	DATA[73]	74th byte of 7-bit encoded data

For Audiolino Brick both *Area ID* and *Element index* are 0.

Note that 7bit encoding of 64 bytes always give a 74 bytes sequence.

RX payload

none

NOTE This command may require a certain amount of time to be executed. To avoid problems always send it with *STATUS* set to 0x11 (command with ACK) and wait the device response, that may be ACK or NACK.

Command 0x19 (DUMP_READ)

With this command you can read a single packet of dump area contents. Data is encoded with MIDI 7bit data encoding and the net payload of each packet is always 64 bytes. This means that sender must use data padding on the last write only, where it may have less than 64 bytes to be written to area. Preferred padding is with 0xFF data.

TX payload

Byte	Name	Description
1	AREA_ID	Area ID
2	INDEX	Element index
3	AD[0]	Area address bit0 to bit6
4	AD[1]	Area address bit7 to bit13
5	AD[2]	Area address bit14 to bit20
6	AD[3]	Area address bit21 to bit27
7	AD[4]	Area address bit28 to bit31)

RX payload

Byte	Name	Description
1	AREA_ID	Area ID
2	INDEX	Element index
3	AD[0]	Area address bit0 to bit6
4	AD[1]	Area address bit7 to bit13
5	AD[2]	Area address bit14 to bit20

Byte	Name	Description
6	AD[3]	Area address bit21 to bit27
7	AD[4]	Area address bit28 to bit31)
8	DATA[0]	1st byte of 7-bit encoded data
...		
81	DATA[73]	74th byte of 7-bit encoded data

For Audiolino Brick both *Area ID* and *Element index* are 0.

Note that 7bit encoding of 64 bytes always give a 74 bytes sequence.

Command 0x1A (DUMP_CRC)

With this command you can ask device to calculate 16-bit CRC of selected area. This may be used for data validation after a dump erase/write sequence.

TX payload

Byte	Name	Description
1	AREA_ID	Area ID
2	INDEX	Element index

RX payload

Byte	Name	Description
1	AREA_ID	Area ID
2	INDEX	Element index
3	CRC[0]	CRC bit0 to bit6
4	CRC[1]	CRC bit7 to bit13
5	CRC[2]	CRC bit14 to bit15

For Audiolino Brick both *Area ID* and *Element index* are 0.

Command 0x1B (FW_UPGRADE_ERASE)

With this command you will erase the entire firmware area. Use this command before a *FW_UPGRADE_WRITE* sequence.

TX payload

none

RX payload

none

NOTE This command may require a certain amount of time to be executed. To avoid problems always send it with *STATUS* set to 0x11 (command with ACK) and wait the device response, that may be ACK or NACK.

Command 0x1C (FW_UPGRADE_WRITE)

With this command you can write a single packet of firmware area contents. Data is encoded with MIDI 7bit data encoding and the net payload of each packet is always 64 bytes. This means that sender must use data padding on the last write only, where it may have less than 64 bytes to be written to area. Preferred padding is with 0xFF data.

TX payload

Byte	Name	Description
1	AD[0]	Area address bit0 to bit6
2	AD[1]	Area address bit7 to bit13
3	AD[2]	Area address bit14 to bit20
4	AD[3]	Area address bit21 to bit27
5	AD[4]	Area address bit28 to bit31)
6	DATA[0]	1st byte of 7-bit encoded data
...		
79	DATA[73]	74th byte of 7-bit encoded data

Note that 7bit encoding of 64 bytes always give a 74 bytes sequence.

RX payload

none

NOTE This command may require a certain amount of time to be executed. To avoid problems always send it with *STATUS* set to 0x11 (command with ACK) and wait the device response, that may be ACK or NACK.

Command 0x70 (LOG)

This message is sent by a device as an event only to log some messages. It is usually used internally for debug purposes but may be used in some production firmware.

TX payload

Byte	Name	Description
1	DATA[0]	1st character of message
...		
N	DATA[N-1]	Nth character of message

RX payload

none

Command 0x71 (CHAIN_INTERNAL)

This command can be used to test communication with the device. It simply does loopback of received test data.

TX payload

Byte	Name	Description
1	TYPE	Message type (device specific)
2	DATA[0]	Data 1st byte
3	DATA[1]	Data 2nd byte
4	DATA[2]	Data 3rd byte
5	DATA[3]	Data 4th byte

RX payload

none

Command 0x72 (LOOPBACK)

This command can be used to test communication with the device. It simply does loopback of received test data.

TX payload

Byte	Name	Description
1	DATA[0]	Test data 1st byte
...		
N	DATA[N-1]	Test data Nth byte

RX payload

Byte	Name	Description
1	DATA[0]	Test data 1st byte
...		
N	DATA[N-1]	Test data Nth byte

Command 0x7F (STATUS)

This command is used to get device status. Device status contents is firmware specific. Please refer to device-specific MIDI implementation for details.

TX payload

none

RX payload

Data is device specific, and may also depend on firmware version.