

AdrenaLinn MIDI Implementation

AdrenaLinn receives the following MIDI messages:

Channel Mode messages:

MIDI Note On, Note Off, Pitch Bend, Pressure/Aftertouch, Controller

The velocity and note number information of MIDI Note On messages, as well as Pitch Bend, Aftertouch, and Controller messages, are used as modulation sources for the AdrenaLinn's Filter Frequency. The MIDI Channel parameter must be set to either "ALL" or the same MIDI channel as the received messages, and the Mod Source parameter must be set to receive the type of message being sent (NOT, VEL, BEN, PRE, CON or modulation combinations containing one of these MIDI sources). In the case of MIDI Controller messages, controller numbers 1, 11, 16, 70 and 74 are recognized and all others are ignored. MIDI Note On messages may also be used to trigger the AdrenaLinn's envelope generator if the EG Trigger parameter is set to either "MID" or "MnP" (Midi with no Program Change).

MIDI Program Change

This message will select user presets U 0-U99. Messages for programs above 99 will be ignored. The EG Trigger parameter must be set to either AUD or MID, and the MIDI Channel parameter must be set to either "ALL" or the same MIDI channel as the received messages.

System Common messages:

MIDI Clock, Start, Stop, Continue, and Song Position Pointer

MIDI Start, Stop and Continue will start, stop and continue the AdrenaLinn's drumbeat and filter sequence. MIDI Clock is used to sync the AdrenaLinn to an external MIDI device. MIDI Song Position Pointer is used to remotely reposition the starting position within the AdrenaLinn's sequences. (When syncing the AdrenaLinn to an external MIDI sequencer, these messages are used to start the AdrenaLinn playing at the correct position within its 2-measure loops regardless of the MIDI sequencer's starting location.) All of these commands will only work if the MIDI Clock parameter is set to either "In" or "I-O" (input & output).

MIDI Song Select

This message will select user drumbeats U 0-U99. Message for songs above 99 will be ignored. The MIDI Channel parameter must be set to either "ALL" or the same MIDI channel as the received messages.

System Exclusive messages:

Please see following pages, "System Exclusive messages"

AdrenaLinn sends the following messages:

System Common messages:

MIDI Clock, Start, and Stop

MIDI Clock messages are sent continuously at the current tempo when the MIDI Clock parameter is set to either "Out" or "I-O". When the AdrenaLinn's drumbeat and filter sequence are started (either by pressing START or by receiving a MIDI Start command), then a Start command sent. When the sequence stops, a Stop command is sent.

No other MIDI messages are sent.

System Exclusive Messages

Format of AdrenaLinn System Exclusive messages

Header (note all numbers are hexadecimal)

Byte 1: F0 (MIDI system exclusive identifier)

Byte 2: 00

Byte 3: 01 Manufacturer ID for Roger Linn Design MSB

Byte 4: 37 ID LSB

Byte 5: 01 Adrenalinn Product ID

Byte 6: 01 File version

Byte 7: Command:

01 = receive single parameter

02 = receive single preset

03 = receive single drumbeat

04 = receive all user presets, user drumbeats and Main & MIDI parameters

05 = Transmit request: single Preset

06 = Transmit request: single Drumbeat

07 = Transmit request: all

08 = Drumbeat Change

7A = DSP code download

Bytes 8 – nnnn: message data

Command = 01, single parameter:

First byte:

00 = preset parameter

01 = drumbeat parameter

02 = Main parameter

03 = MIDI parameter

Second byte:

Address of parameter within preset, drumbeat, main or MIDI data structure (see “Data Structures” below)

Third byte:

LS 4 bit nibble of datum within preset, drumbeat, main or MIDI data structure (see “Data Structures” below)

Fourth byte:

MS 4 bit nibble of within preset, drumbeat, main or MIDI data structure (see “Data Structures” below)

Command = 02, single preset:

Data for preset; 56 bytes of data which = 64 bytes of MIDI (see “Data Packing” and “Data Structures” below)

Command = 03, single drumbeat:

Data for drumbeat; 44 bytes of data, 51 bytes of MIDI (see “Data Packing” and “Data Structures” below)

Command = 04, all user presets and drumbeats plus main and MIDI parameters

See below for details.

Command = 05, transmit Preset Request

Preset Number to transmit (0-99, user presets only)

Command = 06, Transmit Drumbeat Request
Drumbeat Number to transmit (0-99, user drumbeats only)

Command = 07, Transmit All Request
No message data – Transmit All starts when command received

Command = 08, Drumbeat Change
First Byte – New drumbeat number, 0 – 99 (decimal) which selects user drumbeats only

Command = 7A (DSP code image):
Data for DSP code image (for field updates of DSP code-- internal use only!)

Last byte: F7 EOX (end of exclusive)

Data Structures

Preset Parameters

<u>Byte</u>	<u>Parameter</u>	<u>Range/Description</u>
0	Volume	00-99
1	Pan	0-100 (50L – C – 50R), 101-111 (DLY, SEQ, EG, LFO, AUD, HOL, NOT, VEL, CON, BEN, PRE)
2	Filter Mode	0-6 (2PO, 4PO, FL1, FL2, PIT, VOL, OFF)
3	Seq Timebase	0-5 (8n, 8t, 16n, 16h, 32s, DB)
4	Filter Frequency	0-99
5	Mod Source	0-21 (SEQ, EG, LFO, AUD, HOL, NOT, VEL, BEN, CON, PRE, E-L, E-H, E-N, E-V, E-B, E-C, E-P, L-S, L-A, L-C, L-P, S-N)
6	Mod Amount	0-199 (-99 to 0 to 99)
7	Filter Resonance	0-99
8	Envel Attack	0-99
9	Envel Decay	0-199 (0-99 followed by r0 – r99)
10	LFO Wave	0-4 (SIN, TRI, PUL, SAT, RAN)
11	LFO Rate	0-99 followed by 100-111 (8b, 4b, 2b, 1b, 2n, 4n, 8n, 8t, 16n, 16t, 32n, 32t)
12	Amp Drive	0-99, following by 100-199 (b0 – b99)
13	Bass	0-99
14	Mid	0-99
15	Tre	0-99
16	Delay Vol	0-99
17	Delay Time	0-99 followed by 100-108 (2n, 4n, 8n, 8t, 16n, 16t, 32n, 32t)
18	Delay Feedback	0-99
19	Amp Model	0-12 (OFF and 12 amp models)
20-23	Unused	For future expansion
24-55	32 sequence steps	LS 7 bits of each byte holds sequence level of 0-99; MS bit is envel on/off

Drumbeat Parameters

<u>Byte</u>	<u>Parameter</u>	<u>Range/Description</u>
0	Volume	00-99
1	Pan	0-100 (50L – C – 50R) 101-104 (ST1, ST2, ST3, ST4)
2	To Delay/Filter	0-99 (to delay) followed by 100-199 (0-99, to filter)
3	Timebase	0-4 (8n, 8t, 16n, 16h, 16s)
4	Bass sound-vol	10-99 (1 st decimal digit is sound select 1-9, 2 nd decimal digit is mix volume 0-9)
5	Snare sound-vol	10-99 (1 st decimal digit is sound select 1-9, 2 nd decimal digit is mix volume 0-9)
6	Hihat sound-vol	10-99 (1 st decimal digit is sound select 1-9, 2 nd decimal digit is mix volume 0-9)
7	Perc sound-vol	10-99 (1 st decimal digit is sound select 1-5, 2 nd decimal digit is mix volume 0-9)
8	Tempo	30-250 (drumbeat's assigned tempo)
9-11	Unused	For future expansion
12-44	32 sequence steps	Bits 0-1 hold OFF plus 3 volumes of bass Bits 2-3 hold OFF plus 3 volumes of snare Bits 4-5 hold OFF plus 3 volumes of hihat Bits 6-7 hold OFF plus selection of 1 of 3 percussion sounds

Main Parameters

<u>Byte</u>	<u>Parameter</u>	<u>Range/Description</u>
0	Preset	0-99 (F0-F99) followed by 100-199 (U0-F99)
1	Drumbeat	0-99 (F0-F99) followed by 100-199 (U0-F99)
2	Tempo	30-250
3	Volume	0-99
4	Bypass Preset	0 (BYP), 1 (RES), 2-201 (F0-F99-U0-U99)
5	Dir-Amp/Gate	0-3 (DIR, AMP, DNG, ANG)
6	Use Drmbt Tempo	0 (OFF), 1 (ON)
7	Balance/SEP	0-100 (P50 – EQU – D50) followed by 101 (SEP)

MIDI Parameters

<u>Byte</u>	<u>Parameter</u>	<u>Range/Description</u>
0	Channel	0-15 (MIDI channels 1-16) or 16 (ALL)
1	Clock	0-3 (OFF, IN, OUT, I-O)
2	EG Trigger	0-3 (AUD, MID, AnP, MnP)
3	Dump	0-2 (PST, DBT, ALL)

Data Packing

The general data packing scheme for Preset, Drumbeat, and All data dumps groups 7 bytes of data, stripping off the MS bit of each, and packing these MS bits into an additional byte. So, 7 bytes of internal memory yields 8 bytes of MIDI data.

Assuming 7 bytes of memory data are:

AAAAaaaa	Memory byte 0
BBBBbbbb	Memory byte 1
CCCCcccc	Memory byte 2
DDDDdddd	Memory byte 3
EEEEeeee	Memory byte 4
FFFFffff	Memory byte 5
GGGGgggg	Memory byte 6

Then it is sent over MIDI with the MS bits first as follows:

```
0GFEDCBA    Packed MS bits
0AAAAaaa    MIDI Data Bytes
0BBBBbbb
0CCCCccc
0DDDDddd
0EEEEeee
0FFFFfff
0GGGgggg
```

Note that fewer than 7 bytes can be sent, and the unused MS bits will be set to zero. For example, if two bytes are sent:

Assuming 2 bytes of memory data are:

```
AAAAaaaa    Memory byte 0
BBBBbbbb    Memory byte 1
```

Then it is sent over MIDI as a three byte sequence, with the MS bits first as follows:

```
000000BA    Packed MS bits
0AAAAaaa    MIDI Data Bytes
0BBBBbbb
```

Send All

Send All has a slightly different format, since data is sent as one large block. The data (after header) is as follows:

1. Main parameters (8 bytes data = 10 bytes MIDI; standard packing format)
2. MIDI parameters (4 bytes data = 5 bytes MIDI; for completeness, all 4 MIDI bytes are transmitted, even though Channel and Dump are ignored when received.)
3. All 100 user presets and 100 user drumbeats. To avoid overloading the AdrenaLinn input buffers, data is sent as three 4k [4096 byte] memory image blocks. The first block contains Presets 0 – 59. The second has Presets 60-99 and Drumbeats 0-24. The final block has Drumbeats 25-99. The same basic packing is used, only it goes through memory in 16 byte groups, which yields 19 MIDI bytes as such:

```
0GEDCBA    MS bits packed
0AAAAAAA    data byte 1
0BBBBBB
0CCCCCC
0DDDDDD
0EEEEEE
0FFFFFFF
0GGGGGG    data byte 7

0NMLKJIH    MS bits packed
0HHHHHHH    data byte 8
0IIIIIII
```

```
0JJJJJJJ
0KKKKKKK
0LLLLLLL
0MMMMMMM
0NNNNNNN    data byte 14
000000QP    MS bits packed
0PPPPPPP    data byte 15 ("0" is not used here for clarity)
0QQQQQQQ    data byte 16
```

The data structures for Presets and Drumbeats are as before, with one difference. Presets have an additional 8 bytes of zeros after the 56 bytes shown above in Data Structures, and Drumbeats have an additional 7 bytes of zeros. These are also packed in the continuous memory dump. So, each Preset is 64 bytes of memory, and each Drumbeat is 51 bytes of memory. Also, to avoid overloading the AdrenaLinn input buffers, the three 4096 byte blocks are separated by 4,500 zero bytes, followed by a 7F. The exact number of zeros is not critical – just around 1.5 seconds of delay; the 7F signals that the next byte is the first in the next block of data.