ALESIS MMT-8 MIDI SYSTEM EXCLUSIVE FORMAT

The following information is provided as a guide for programmers wishing to modify the data received via MIDI from the MMT-8 for the purpose of interchanging parts from separate block dumps, modification of part names, MIDI channel assignments, etc. Great care must be taken to insure that all modified addresses are valid, since one incorrect value (the length of a part, for example) could result in all data being lost in the MMT-8. These errors may not show up immediately, since the incorrect values may not be accessed by the MMT-8 until a particular part or song is selected. Therefore, it is recommended that any data manipulation programs be thoroughly tested after loading into the MMT-8 by selecting and recording on many parts before assuming that the data is valid.

A system exclusive MIDI data dump from the MMT-8 is initiated by holding the TAPE button down, pressing (and releasing) the PAGE DOWN button once, and then pressing the RECORD button. The data sent out MIDI is in the following format:

HEX	COMMENTS
F0H	SYSTEM EXCLUSIVE STATUS BYTE
00H	
00H	
0EH	ALESIS I.D. NUMBER
00H	MMT-8 I.D. NUMBER

Following the above 5 bytes will be a block of data representing the contents of the MMT-8's memory. In order to optimize the data transfer, 8 MIDI bytes are used to transmit each block of 7 MMT-8 data bytes. If the 7 data bytes are looked at as one 56-bit word, the format for transmission is eight 7-bit words beginning with the most significant bit of the first byte, as follows:

	SEVEN MMT-8 BYTES:
0:	A7 A6 A5 A4 A3 A2 A1 A0
1:	B7 B6 B5 B4 B3 B2 B1 B0
2:	C7 C6 C5 C4 C3 C2 C1 C0
3:	D7 D6 D5 D4 D3 D2 D1 D0
4:	E7 E6 E5 E4 E3 E2 E1 E0
5:	F7 F6 F5 F4 F3 F2 F1 F0
6:	G7 G6 G5 G4 G3 G2 G1 G0

TRANSMITTED AS:

0: 0 A7 A6 A5 A4 A3 A2 A1
1: 0 A0 B7 B6 B5 B4 B3 B2
2: 0 B1 B0 C7 C6 C5 C4 C3
3: 0 C2 C1 C0 D7 D6 D5 D4
4: 0 D3 D2 D1 D0 E7 E6 E5
5: 0 E4 E3 E2 E1 E0 F7 F6
6: 0 F5 F4 F3 F2 F1 F0 G7
7: 0 G6 G5 G4 G3 G2 G1 G0

In order to use the data properly, it must be decoded properly into MMT-8 byte format. The following list gives the data locations within the "unpacked" (decoded) block of data, starting with the first byte of the block being 000. NOTE: All absolute addresses must have an offset of 400H added to them (e.g., an absolute pointer to a part that starts at 35AH should have the pointer value 75AH).

000H	MSB of absolute pointer to part 00
001H	LSB of absolute pointer to part 00
002H	MSB of absolute pointer to part 01
003H	LSB of absolute pointer to part 01
004H	MSB of absolute pointer to part 02
005H	LSB of absolute pointer to part 02
"	" " " "
0C6H	MSB of absolute pointer to part 99
0C7H	LSB of absolute pointer to part 99
0C8H-0CEH	DON'T ALTER
0CFH	LSB of absolute pointer to 1st byte past SONG 99 data (start of free mem)
0D0H	MSB of absolute pointer to 1st byte past SONG 99 data (start of free mem)
0D1H-0D2H	DON'T ALTER
0D3H	LSB of FF00H minus data in 0CFH & 0D0H (length of free mem)
0D4H	MSB of FF00H minus data in 0CFH & 0D0H (length of free mem)
0D5H-101H	DON'T ALTER
102H	MSB of absolute pointer to song 00
103H	LSB of absolute pointer to song 00
104H	MSB of absolute pointer to song 01
105H	LSB of absolute pointer to song 01
106H	MSB of absolute pointer to song 02
107H	LSB of absolute pointer to song 02
"	" " " "
1C8H	MSB of absolute pointer to song 99
1C9H	LSB of absolute pointer to song 99
1CAH-1FFH	DON'T ALTER
200H-?	PART 00 DATA

The part and song data must be dealt with in a specific manner:

1) All part and song data must be in consecutive order, i.e., part 05 data cannot be before part 02 data. The order for the data should be part 00 through 99, followed by song 00 through 99.
2) If a part or song does not exist, its MSB pointer will = 0, which is an illegal pointer address. Since there will be no data for this part, it is skipped, i.e., if part 04 is empty, part 05's data follows after part 03's data.
3) There can be no gaps in the data. Part 01's data must follow directly after part 00's data, etc.
4) Locations 0CFH-0D0H (start of free memory) and 0D3H-0D4H (length of free memory) must be kept valid.

PART DATA FORMAT

The following is the format of each part, starting with the address pointed to byte the absolute pointer to the part (offset by 400H):

00H	LSB of number of bytes in part, including header.
01H	MSB of number of bytes in part, including header.
02H	LSB of offset from start of part to address of start of track 8 data
03H	MSB of offset from start of part to address of start of track 8 data
04H	LSB of offset from start of part to address of start of track 7 data
05H	MSB of offset from start of part to address of start of track 7 data
06H	LSB of offset from start of part to address of start of track 6 data
07H	MSB of offset from start of part to address of start of track 6 data
08H	LSB of offset from start of part to address of start of track 5 data
09H	MSB of offset from start of part to address of start of track 5 data
0AH	LSB of offset from start of part to address of start of track 4 data
0BH	MSB of offset from start of part to address of start of track 4 data
0CH	LSB of offset from start of part to address of start of track 3 data
0DH	MSB of offset from start of part to address of start of track 3 data
0EH	LSB of offset from start of part to address of start of track 2 data
0FH	MSB of offset from start of part to address of start of track 2 data
10H	LSB of offset from start of part to address of start of track 1 data
11H	MSB of offset from start of part to address of start of track 1 data
12H	LSB of number of beats in part in BCD format (0 beats =
13H	MSB of number of beats in part in BCD format empty part)
14H	MIDI channel for track 8 (0=unchanged, or 1-16)
15H	MIDI channel for track 7 (0=unchanged, or 1-16)
16H	MIDI channel for track 6 (0=unchanged, or 1-16)
17H	MIDI channel for track 5 (0=unchanged, or 1-16)
18H	MIDI channel for track 4 (0=unchanged, or 1-16)
19H	MIDI channel for track 3 (0=unchanged, or 1-16)
1AH	MIDI channel for track 2 (0=unchanged, or 1-16)
1BH	MIDI channel for track 1 (0=unchanged, or 1-16)
1CH-29H	14 digit ASCII name of part
2AH-?	Data for track 8
?-?	Data for track 7, etc...

Part data must follow these rules:

1) Track data must be in the order track 8 through track 1.
2) An empty track must exist as a track with no notes in it. The data for an empty track would be 7 bytes long, as follows: 80H, xLSB, xMSB, 00H, 80H, 00H, 00H, with x = the number of clocks in the part, i.e., number of beats * 96. The number of clocks must be divisible by 96.
3) Adding the number of bytes in a part to the absolute pointer of a part should point to 1 byte past the last byte of the part.

Each data event within a track consists of either 7 or 5 bytes, depending on whether or not other events exists on that same clock step. The 7 byte format is as follows:

Byte #	Format	Comment
1	1nnnnnnn	note or controller number
2	xxxxxxxx	absolute start time lsb
3	yyyyyyyy	absolute start time msb
4	zvvvvvvv	note/controller flag, velocity or controller amount
5	0000cccc	MIDI channel number
6	0aaaaaaa	note duration msb or pitch bend lsb
7	bbbbbbbb	note duration lsb or pitch bend msb

The 5 byte format is similar, but does not include a start time:

Byte #	Format	Comment
1	0nnnnnnn	note or controller number
2	zvvvvvvv	note/controller flag, velocity or controller amount
3	0000cccc	MIDI channel number
4	0aaaaaaa	note duration msb or pitch bend lsb
5	bbbbbbbb	note duration lsb or pitch bend msb

Note that the most significant bit of each packet determines the length of the packet: A 7 byte packet has its 1st byte msb set to 1, while a 5 byte packet has the 1st byte msb set to 0. Each packet contains one of the following seven events: Note event (with duration), controller 0-121 event, program change event, after touch event, pitch bend event, sysex event, or end of track event. The five byte packet versions of these events are shown below. If the event is a seven byte packet, the second and third bytes will contain the clock count at which the event should occur. Each clock count = 1/384th note (4 times MIDI clock resolution), with 0000 = first beat of sequence. 5 byte packets always occur on the clock specified by the nearest 7 byte packet preceeding it.

Note format:

0nnnnnnn	Note number 0-127
0vvvvvvv	Velocity 1-127
0000cccc	MIDI channel number 0 through 15
0aaaaaaa	Note duration msb (number of clocks until note off)
bbbbbbbb	Note duration lsb (number of clocks until note off)

Controller 0-121 format:

0nnnnnnn	Controller number 0-121
1vvvvvvv	Controller amount
0000cccc	MIDI channel number 0-15
00000000	not used
00000000	not used

Program change format:

01111010	Program change flag (122)
1vvvvvvv	Program number 0-127
0000cccc	MIDI channel number 0-15
00000000	not used
00000000	not used

After touch change format:

01111011	After touch flag (123)
1vvvvvvv	After touch amount 0-127
0000cccc	MIDI channel number 0-15
00000000	not used
00000000	not used

Pitch bend format:

01111100	Pitch bend flag (124)
10000000	not used
0000cccc	MIDI channel number 0-15
0aaaaaaa	Pitch bend lsb
0bbbbbbb	Pitch bend msb

Sysex format:

01111101	Sysex flag (125)
1vvvvvvv	sysex byte 0-127
bccccccc	b=1=EOX, b=0 then c=sysex byte
00000000	not used
deeeeeee	d=1=EOX, d=0 then e=sysex byte

A sysex message ends (and an EOX is sent) whenever bits b or d are high, or a new 7 byte packet occurs, or a packet other than sysex occurs. The sysex message can be as long as necessary by having consecutive 5 byte sysex packets.

End of track format:

80H, # clocks lsb, # clocks msb, 0, 80H, 0, 0
The number of clocks should equal the number of beats in the sequence multiplied by 96.

SONG DATA FORMAT

The following is the format of each song, starting with the address pointed to by the absolute pointer to the part (offset by 400H):

00H	LSB of number of bytes in song, including header.
01H	MSB of number of bytes in song, including header.
02H	Tempo of song, in BPM
03H-10H	14 digit ASCII name of song
11H	Step 1 part number (00-99)
12H	Step 1 play tracks (bit 0=track 1, bit 7=track 8; 0=off, 1=on)
13H	Step 2 part number
14H	Step 2 play tracks
15H	etc....
xxH	Part number 0FFH (end of song)

Song data must follow these rules:

1) Part numbers 100-254 are not allowed.
2) There cannot be more than 255 steps in a song.
3) Adding the number of bytes in a song to the absolute pointer of a song should point to 1 byte past the last byte of the song.