NOTE: This is version 1.0, made available here by permission of the MIDI Manufacturers Association. Version 1.1 is the current one and should be obtained directly from the MMA (www.midi.org).

MIDI SHOW CONTROL (MSC) 1.0
MIDI 1.0 Recommended Practice RP-002
1991-07-25

MIDI Manufacturers Association
(info at midi.org)

1. INTRODUCTION

The purpose of MIDI Show Control is to allow MIDI systems to communicate with and to control dedicated intelligent control equipment in theatrical, live performance, multi-media, audio-visual and similar environments.

Applications may range from a simple interface through which a single lighting controller can be instructed to GO, STOP or RESUME, to complex communications with large, timed and synchronized systems utilizing many controllers of all types of performance technology.

The set of commands is modeled on the command structure of currently existing computer memory lighting, sound and show control systems. The intent is that translation between the MIDI Show Control specification and dedicated controller commands will be relatively straightforward, being based on the same operating principles. On the other hand, it has been assumed that translation will involve more than table look-up, and considerable variation will be found in data specifications and other communications details. In essence, MIDI Show Control is intended to communicate easily with devices which are designed to execute the same set or similar sets of operations.

2. GENERAL STRUCTURE

2.1. UNIVERSAL SYSTEM EXCLUSIVE FORMAT

MIDI Show Control uses a single Universal Real Time System Exclusive ID number (sub-ID 1 = 02H) for all Show commands (transmissions from Controller to Controlled Device).

In this version of Show Control, no command responses (from Controlled Device to Controller) are specified or required in order to optimize bandwidth requirements, system response time and system reliability in the event of communication difficulties with one or more Controlled Device. The guiding philosophy behind live performance control is that, as much as possible, failures of individual Controlled Devices should not impair communications with other Controlled Devices. This concept has been a part of MIDI design from the beginning and MIDI Show Control continues to use an "open-loop" design in order that standard MIDI practices may continue to be successfully utilized in applications using all types of standard Channel and System messages. However, a "closed-loop" version of Show Control has been discussed and may be created in the future.

In this document all transmitted characters are represented in hex unless otherwise noted. The initials "msc" will be used to denote the new MIDI Show Control sub-ID 1 (= 02H).

The format of a Show Control message is as follows:

F0 7F <device_ID> 02 <command_format> <command> <data> F7

NOTES:

1. No more than one command can be transmitted in a Sysex.
2. The total number of bytes in a Show Control message should not exceed 128.
3. Sysex's must always be closed with an F7H as soon as all currently prepared information has been transmitted.

2.2. DEVICE IDENTIFICATION

<device_ID> is always a DESTINATION device address.

Commands are most often addressed to one device at a time. For example, to command two lighting consoles to GO, transmit:

F0 7F <device_ID=1> 02 <command_format=lighting> <GO> F7
F0 7F <device_ID=2> 02 <command_format=lighting> <GO> F7

<device_ID> values:

00-6F	Individual ID's
70-7E	Group ID's 1-15 (optional)
7F	"All-call" ID for system wide broadcasts

Every device must be able to respond to both an individual and the "all-call" (7FH) ID. The group addressing mode is optional. A device may respond to one or more individual ID and one or more group ID. Both <device_ID> and <command_format> of a message must match the device_ID and command_format of a controlled device before the message is recognized.

If two separate controlled devices responding to the same command_format are set to respond to the same device_ID then only one message need be sent for both to respond.

The "all-call" device_ID (7FH) is used for system wide "broadcasts" of identical commands to devices of the same command_format (or to all devices when used with <command_format=all-types>; see 4.1, below.)

Before fully interpreting the <device_ID> byte, parsing routines will need to look at <msc> and <command_format>, both of which follow <device_ID>, in order to first determine that the Sysex contains Show Control commands in the appropriate format.

A typical system will consist of at least one Controller attached to one or more Controlled Devices. It is possible for the same machine to be both a Controlled Device and a Controller at the same time. In this case, the machine may act as a translator, interpreter or converter of Show Control commands. According to its programmed instructions, the receipt of one type of command may result in the transmission of similar or different commands.

It is also a possibility that multiple Controller outputs could be merged and distributed to one or more Controlled Devices.

Optionally, Controlled Devices may be able to transmit (from a MIDI Out connector) MIDI Show Control commands of the type required by themselves to produce a desired result. In this condition, the Controlled Device will be transmitting a valid MIDI Show Control Command but may not necessarily be doing so as a Controller.

This is useful when the Controller has the ability (through MIDI In) to capture valid MIDI Show Control messages in order to conveniently create and edit the database of messages needed for the performances being controlled. In this case, the Controlled Device will be transmitting to the Controller, but only for the purposes of capturing messages to store and retransmit during performance.

Another application allowed by the transmission of Show Control commands by Controlled Devices is the slaving of multiple Devices of similar type. For example, if a dedicated lighting console transmits a Show Control command to "GO" when its GO button is pressed, then any other dedicated lighting console that obeys MIDI Show Control commands will also GO if it receives MIDI from the first console. In this way, many Controlled Devices may be controlled by another Controlled Device acting as the Controller. Interconnection would follow the same pattern as the normal Controller to Controlled Device arrangement.

2.3. COMMAND_FORMATS

A command_format is a message byte from a Controller to a Controlled Device which identifies the format of the following Command byte. Each command_format has a format code between 01H and 7FH, and must be followed by a valid command byte. (Command_format 00H is reserved for extensions, and not all codes are currently defined.)

2.4. COMMANDS

A command is a message byte from a Controller to a Controlled Device. Each command has a command code between 01H and 7FH, and may be followed by one or more data bytes, up to a total message length of 128 bytes. (Command 00H is reserved for extensions, and not all codes are currently defined.)

2.5. EXTENSION SETS

Command_Format 00H and command 00H are reserved for two extension sets:

```
00 01     1st command_format or command at 1st extension level
00 00 01  1st command_format or command at 2nd extension level
```

At this time, no extended functions have been defined. Nevertheless, to accommodate future extensions to MIDI Show Control, parsing routines must always check for extensions wherever command_format or command fields are encountered in data.

2.6. DATA LENGTH

The only restriction to the number of data bytes sent is that

the total number of message bytes must not be more than 128. The actual data format of the transmitted message will be defined by the manufacturer of the Controlled Device. This means that the Controller (or the programmer of the Controller) must know the exact data format of the Controlled Device. This information will be manufacturer and equipment specific, so it is important that every manufacturer publish a thorough and unambiguous Sysex Implementation document.

Because this specification is intended to accommodate the needs of an extremely wide variety of equipment and industry needs, from very low cost light boards to the most complex audio/video multimedia extravaganzas, the data formats used in simpler systems will be considerably shorter and less complex than in comprehensive equipment. Data are transmitted in the order of most generic information first, with null character delimiters between each group of data bytes in order to signify the sending of progressively less generic data. For instance, simple Controlled Devices may look only at the basic data and discard the rest.

As an example, a complex Controlled Device may be able to process cue numbers with a large number of decimal point delineated subsections i.e. "235.32.7.8.654" If a Controller transmits this cue number to a simple Controlled Device that can only process numbers in the form "xxx.x", then the simple Device can either ignore these data or else respond to them in a predictable manner, such as processing cue number "235.3."

As a further example, cue number data may be transmitted calling up cue 235.3 then followed by a delimiter and data specifying cue list 36.6 and followed by a further delimiter specifying cue path 59. If the Device supports multiple cue lists but not multiple cue paths, it would process cue 235.3 in cue list 36.6 (or 36) and ignore the cue path data, simply using the current or default cue path.

Looking at the situation in the opposite manner, if simple cue number data were transmitted to a Device capable of processing all cue data, it would respond by processing that cue number in the current or default cue list using the current or default cue path.

3. STANDARD SPECIFICATIONS

Since data often contain some form of Cue Number designation, a "Standard" specification for transmission of Cue Number and related data provides consistency and saves space in the detailed data descriptions (Section 5).

3.1. CUE NUMBERS

When a Cue Number is sent as data, the following additional information fields may or may not be included as part of a complete "Cue Number" description: Q_list and Q_path. Q_list prescribes in which one of all currently Open Cue Lists the Q_number is to be placed or manipulated. Q_path prescribes from which Open Cue Path within all available cue storage media the Q_number is to be retrieved. The data include these information fields in the following order:

<Q_number> 00 <Q_list> 00 <Q_path> F7

Between each separate field a delimiter byte of the value 00H is placed as shown to indicate the end of the previous field and beginning of the next. It is acceptable to send only:

<Q_number> F7
or
<Q_number> 00 <Q_list> F7.

Controlled Devices should be able to accept more than one set of delimiter bytes, including directly before F7H, and even if no Q_number, Q_list or Q_path data are sent. Data are always terminated by F7H.

Q_number, Q_list and Q_path are expressed as ASCII numbers 0-9 (encoded as 30H-39H) with the ASCII decimal point character (2EH) used to delineate subsections. In the example above, cue 235.6 list 36.6 path 59 would be represented by the hex data:

32 33 35 2E 36 00 33 36 2E 36 00 35 39 F7

Decimal points should be separated by at least one digit, but Controlled Devices should accommodate the error of sending two or more decimal points together. Any number of decimal point delineated subsections may be used and any number of digits may be used in each subsection except that the length of the data must not cause the total length of the MIDI Show Control message to exceed 128 bytes.

Controlled Devices which do not support Q_list (or Q_path) data must detect the 00H byte immediately after the Q_number (or Q_list) data and then discard all data until F7H is detected. Likewise, Controlled Devices which do not support the received number of decimal point delineated subsections, the received number of digits in a subsection or the total number of received characters in any field must handle the data received in a predictable and logical manner.

Controlled Devices which support Q_list and/or Q_path will normally default to the current or base Q_list and Q_path if these fields are not sent with Q_number.

For lighting applications, Q_list optionally defines the Playback or Submaster Controls (0 to 127) with which the cue corresponds.

It is highly recommended that every manufacturer publish a clear and concise description of their equipment's response to the above conditions.

3.2. TIME CODE NUMBERS

Since data often contain some form of time reference, a "Standard" specification for transmission of time provides consistency and saves space in the data descriptions.

MIDI Show Control time code and user bit specifications are entirely consistent with the formats used by MIDI Time Code and MIDI Cueing and are identical to the Standard Time Code format proposed in MIDI Machine Control 0.05. Some extra flags have been added, but are defined such that if used in the MIDI Time Code/Cueing environment they would always be reset to zero, and so are completely transparent.

3.2.1. STANDARD TIME CODE (types {ff} and {st}):

This is the "full" form of the Time Code specification, and always contains exactly 5 bytes of data.

Two forms of Time Code subframe data are defined:

The first (labelled {ff}), contains subframe data exactly as described in the MIDI Cueing specification i.e. fractional frames measured in 1/100 frame units.

The second form (labelled {st}) substitutes time code "status" data in place of subframes. For example, when reading data from tape, it is useful to know whether these are real time code data, or simply time data updated by tachometer pulses during a high speed wind. In this case, as in other cases of "moving" time code, subframe data are practically useless, being difficult both to obtain and to transmit in a timely fashion.

```
hr mn sc fr (ff|st)
     hr = Hours and type:  0 tt hhhhh
          tt = time type (bit format):
               00 = 24 frame
               01 = 25 frame
               10 = 30 drop frame
               11 = 30 frame
          hhhhh = hours (0-23, encoded as 00-17hex)
     mn = Minutes: 0 c mmmmmm
          c = colour frame bit (copied from bit in time code
              stream):
               0 = non colour frame
               1 = colour framed code
          mmmmmm = minutes (0-59, encoded as 00-3Bhex)
     sc = Seconds: 0 k ssssss
          k = reserved - must be set to zero
          ssssss = seconds (0-59, encoded as 00-3Bhex)
     fr = Frames, byte 5 ident and sign: 0 g i fffff
          g = sign bit:
               0 = positive
               1 = negative (where signed time code is
                   permitted)
          i = final byte identification bit:
               0 = subframes
               1 = status
          fffff = frames (0-29, encoded as 00-1Dhex)
If final byte bit = subframes (i = 0):
     ff = fractional frames:  0 bbbbbbb (0-99, encoded as
          00-63hex)
If final byte bit = status (i = 1):
     st = code status: 0 e v d xxxx
          e = estimated code flag bit:
               0 = normal time code
               1 = tach or control track updated code
          v = invalid code bit (ignore if e = 1):
               0 = valid
               1 = invalid (error or not current)
          d = video field identification bit:
               0 = no field information in this frame
               1 = first frame in 4 or 8 field video
                   sequence
          xxxx = reserved bits - must be set to 0000
```

DROP FRAME NOTES

1. When writing time code data, the drop-frame or non-drop-frame status of the data being written may be overridden by the status of the Controlled Device (i.e. the time code from the device itself).

   For example, if the SET_CLOCK data are loaded with a non-drop-frame number and if the time code on the Controlled Device is drop-frame, then the SET_CLOCK data will simply be interpreted as a drop-frame number, with no attempt being made to perform any mathematical transformations.

2. Furthermore, if the above SET_CLOCK number had in fact been loaded with a non-existent drop-frame number (e.g. 00:22:00:00), then the next higher valid number would have been used (in this case, 00:22:00:02).

3. Calculation of offsets, or simply the mathematical difference between two time codes, can cause confusion when one or both of the numbers is drop-frame.

   For the purposes of this specification, DROP-FRAME NUMBERS SHOULD FIRST BE CONVERTED TO NON-DROP-FRAME BEFORE OFFSET CALCULATIONS ARE PERFORMED. Results of an offset calculation will then be expressed as non-drop-frame quantities.

   To convert from drop-frame to non-drop-frame, subtract the number of frames that have been "dropped" since the reference point 00:00:00:00. For example, to convert the drop-frame number 00:22:00:02 to non-drop-frame, subtract 40 frames, giving 00:21:58:22. The number 40 is produced by the fact that 2 frames were "dropped" at each of the minute marks 01 through 09, 11 through 19, 21 and 22.

   (Some manufacturers will prefer to store all internal time codes as a simple quantity of frames from reference point 00:00:00:00. This reduces calculation complexity, but does require that conversions are performed at all input or output stages.)

4. INDEX LIST

4.1. COMMAND_FORMATS

Command_formats fall into the categories of General, Specific and All-types. General command_formats have a least significant nibble equal to 0, except for lighting which is 01H. Specific command_formats are related to the General command_format with the most significant nibble of the same value, but represent a more restricted range of functions within the format.

Command_format "All-types" (7FH) is used for system wide "broadcasts" of identical commands to devices of the same device_ID (or to all devices when used with <device_ID=All-call>; see 2.2, above.)

For example, use of the All-types command_format along with the All-call device_ID allows a complete system to be RESET with a single message.

Controlled Devices will normally respond to only one

command_format besides All-types. Occasionally, more complex control systems will respond to more than one command_format since they will be in control of more than one technical performance element. Controllers, of course, should normally be able to create and send commands in all command_formats, otherwise their usefulness will be limited.

Hex	command_format	
00	reserved for extensions	
01	Lighting	(General Category)
02	Moving Lights	
03	Colour Changers	
04	Strobes	
05	Lasers	
06	Chasers	
10	Sound	(General Category)
11	Music	
12	CD Players	
13	EPROM Playback	
14	Audio Tape Machines	
15	Intercoms	
16	Amplifiers	
17	Audio Effects Devices	
18	Equalisers	
20	Machinery	(General Category)
21	Rigging	
22	Flys	
23	Lifts	
24	Turntables	
25	Trusses	
26	Robots	
27	Animation	
28	Floats	
29	Breakaways	
2A	Barges	
30	Video	(General Category)
31	Video Tape Machines	
32	Video Cassette Machines	
33	Video Disc Players	
34	Video Switchers	
35	Video Effects	
36	Video Character Generators	
37	Video Still Stores	
38	Video Monitors	
40	Projection	(General Category)
41	Film Projectors	
42	Slide Projectors	
43	Video Projectors	
44	Dissolvers	
45	Shutter Controls	
50	Process Control	(General Category)
51	Hydraulic Oil	
52	H20	
53	CO2	
54	Compressed Air	
55	Natural Gas	

56	Fog	
57	Smoke	
58	Cracked Haze	
60	Pyro	(General Category)
61	Fireworks	
62	Explosions	
63	Flame	
64	Smoke pots	
7F	All-types	

Although it can be seen that a wide variety of potentially dangerous and life-threatening performance processes may be under MIDI Show Control, the intent of this specification is to allow the user considerably more exacting and precise control over the type of command_format and command which will result in the desired result than normally may be provided in a non-electronic cueing situation. The major advantages to the use of MIDI Show Control in these conditions are:

1. Less likelihood of errors in cueing. Digital communications can be demonstrated to be extremely reliable in repetitive duty conditions; much more so than tired or inexperienced stagehands.

2. More precise timing. Likewise, digital communications and computer control can be consistently accurate in automatic timing sequences and exactly as accurate as their human operators when under manual control.

IN NO WAY IS THIS SPECIFICATION INTENDED TO REPLACE ANY ASPECT OF NORMAL PERFORMANCE SAFETY WHICH IS EITHER REQUIRED OR MAKES GOOD SENSE WHEN DANGEROUS EQUIPMENT IS IN USE. MANUAL CONTROLS SUCH AS EMERGENCY STOPS, DEADMAN SWITCHES, CONFIRMATION ENABLE CONTROLS OR LIKE SAFETY DEVICES SHALL BE USED FOR MAXIMUM SAFETY.

AUTOMATIC SAFETY DEVICES SUCH AS LIMIT SWITCHES, PROXIMITY SENSORS, GAS DETECTORS, INFRARED CAMERAS AND PRESSURE AND MOTION DETECTORS SHALL BE USED FOR MAXIMUM SAFETY. MIDI SHOW CONTROL IS NOT INTENDED TO TELL DANGEROUS EQUIPMENT WHEN IT IS SAFE TO GO: IT IS ONLY INTENDED TO SIGNAL WHAT IS DESIRED IF ALL CONDITIONS ARE ACCEPTABLE AND IDEAL FOR SAFE PERFORMANCE. ONLY PROPERLY DESIGNED SAFETY SYSTEMS AND TRAINED SAFETY PERSONNEL CAN ESTABLISH IF CONDITIONS ARE ACCEPTABLE AND IDEAL AT ANY TIME.

4.2. RECOMMENDED MINIMUM SETS

MIDI Show Control does not specify an absolute minimum set of commands and data which must be implemented in each device responding to a given command_format.

However, in order to ease the burden of interfacing between Controllers and Controlled Devices from different manufacturers, four RECOMMENDED MINIMUM SETS of commands and data have been created. Once a Controlled Device is specified to conform to a particular Recommended Minimum Set, then the task of designing a Controller which will successfully operate that device is considerably simplified.

The currently defined Recommended Minimum Sets are:

1. Simple Controlled Device; no time code; basic data only

2. No time code; full data capability
3. Full time code; full data capability

Assignment of any particular command or data to a Recommended Minimum Set may be found in the far right hand column of the Index List.

Recommended Minimum Sets are in no way intended to restrict the scope of operations supported by any device. They are offered only in the spirit of a "lowest common denominator".

4.3. GENERAL COMMANDS

The following commands are basic to the current implementation of Memory Lighting systems and probably apply to all dedicated theatrical show control systems in a general sense. Although it is not required that Controlled Devices incorporate all of these commands, it is highly recommended:

Hex	command	Number of data bytes	Recomm'd Min Sets
00	reserved for extensions		
01	GO	variable	123
02	STOP	variable	123
03	RESUME	variable	123
04	TIMED_GO	variable	-23
05	LOAD	variable	-23
06	SET	4 or 9	-23
07	FIRE	1	-23
08	ALL_OFF	0	-23
09	RESTORE	0	-23
0A	RESET	0	-23
0B	GO_OFF	variable	-23

4.4. SOUND COMMANDS

The following commands, in addition to the above, are basic to the current implementation of Computer Controlled Sound Memory Programming Systems and are widely used by Show Control Systems in more comprehensive applications. It is recommended that Controllers support the transmission of these commands:

Hex	command	Number of data bytes	Recomm'd Min Sets
10	GO/JAM_CLOCK	variable	--3
11	STANDBY_+	variable	-23
12	STANDBY_-	variable	-23
13	SEQUENCE_+	variable	-23
14	SEQUENCE_-	variable	-23
15	START_CLOCK	variable	--3
16	STOP_CLOCK	variable	--3
17	ZERO_CLOCK	variable	--3
18	SET_CLOCK	variable	--3
19	MTC_CHASE_ON	variable	--3
1A	MTC_CHASE_OFF	variable	--3
1B	OPEN_CUE_LIST	variable	-23
1C	CLOSE_CUE_LIST	variable	-23
1D	OPEN_CUE_PATH	variable	-23

5. DETAILED COMMAND AND DATA DESCRIPTIONS

00 Reserved for extensions

01 GO

Starts a transition or fade to a cue. Transition time is determined by the cue in the Controlled Device. If no Cue Number is specified, the next cue in numerical sequence GOes. If a Cue Number is specified, that cue GOes.
Transitions "run" until complete.
If the Controller wishes to define the transition time, TIMED_GO should be sent.

In Controlled Devices with multiple Cue Lists, if no Cue Number is Specified, the next cues in numerical order and numbered identically and which are in Open Cue Lists GO. If Q_number is sent without Q_list, all cues with a number identical to Q_number and which are in Open Cue Lists GO.

01	GO
<Q_number>	optional; required if Q_list is sent
00	delimiter
<Q_list>	optional; required if Q_path is sent
00	delimiter
<Q_path>	optional

02 STOP

Halts currently running transition(s). If no Cue Number is specified, all running transitions STOP. If a Cue Number is specified, only that single, specific transition STOPs, leaving all others unchanged.

02	STOP
<Q_number>	optional; required if Q_list is sent
00	delimiter
<Q_list>	optional; required if Q_path is sent
00	delimiter
<Q_path>	optional

03 RESUME

Causes STOPped transition(s) to continue running. If no Cue Number is specified, all STOPped transitions RESUME. If a Cue Number is specified, only that transition RESUMEs, leaving all others unchanged.

03	RESUME
<Q_number>	optional; required if Q_list is sent
00	delimiter
<Q_list>	optional; required if Q_path is sent
00	delimiter
<Q_path>	optional

04 TIMED_GO

Starts a timed transition or fade to a cue. If no Cue Number is specified, the next cue in numerical sequence GOes. If a Cue Number is specified, that cue GOes. Transitions "run" until complete.

Time is Standard Time Specification with subframes (type {ff}), providing anything from "instant" to 24 hour transitions. If a Controlled Device does not support TIMED_GO it should GO instead, ignoring the time data but processing Cue Number data normally. If the transition time desired is the preexisting default cue time, GO should be sent instead of TIMED_GO.

Rules for Controlled Devices with multiple Cue Lists are the same as for GO, above.

04	TIMED_GO
hr mn sc fr ff	Standard Time Specification
<Q_number>	optional; required if Q_list is sent
00	delimiter
<Q_list>	optional; required if Q_path is sent
00	delimiter
<Q_path>	optional

05 LOAD

Places a cue into a standby position. Cue Number must be specified. LOAD is useful when the cue desired takes a finite time to access. LOAD is sent in advance so that the cue will GO instantly.

In Controlled Devices with multiple Cue Lists, if Q_number is sent without Q_list, all cues with a number identical to Q_number and which are in Open Cue Lists LOAD to standby.

05	LOAD
<Q_number>	required
00	delimiter
<Q_list>	optional; required if Q_path is sent
00	delimiter
<Q_path>	optional

06 SET

Defines the value of a Generic Control. The Generic Control and its value are each specified by a 14 bit number. A Controlled Device may treat virtually any of its variables, attributes, rates, levels, modes, functions, effects, subs, channels, switches, etc. as Generic Controls which may be sent values via SET. Optionally, the time it takes the Generic Control to achieve its value may be sent.

Time is Standard Time Specification with subframes (type {ff}), providing anything from "instant" to 24 hour transitions. If a Controlled Device does not support times in SET, it should ignore time data.

Standard Generic Control Numbers for Lighting:

0-127	Sub masters
128-129	Masters of the first playback
130-131	Masters of the second playback
...	
etc.	
...	
190-191	Masters of the 32nd playback
192-223	Speed controllers for the 32 playbacks
224-255	Chase sequence masters
256-287	Chase sequence speed masters
510	Grand Master for all channels
511	General speed controller for all fades
512-1023	Individual channel levels

06	SET
cc cc	Generic Control Number, LSB first
vv vv	Generic Control Value, LSB first
hr mn sc fr ff	Standard Time Specification, optional

07 FIRE

Triggers a preprogrammed keyboard Macro. The Macro is defined by a 7 bit number. The Macros themselves are either programmed at the Controlled Device, or loaded via MIDI file dump facilities using the ASCII Cue Data format or any method applicable to the Controlled Device.

07	FIRE
mm	Macro Number

08 ALL_OFF

Independently turns all functions and outputs off without changing the control settings. Operating status prior to ALL_OFF may be reestablished by RESTORE.

08	ALL_OFF

09 RESTORE

Reestablishes operating status to exactly as it was prior to ALL_OFF.

09	RESTORE

0A RESET

Terminates all running cues, setting all timed functions to an initialized state equivalent to a newly powered-up condition and loads the first cue of each applicable cue list into the appropriate standby positions. In other words, RESET stops the show without arbitrarily changing any control values and loads the top of the show to standby.

It should be decided by the manufacturer of the Controlled Device whether or not RESET should automatically open all CLOSEd_CUE_LISTs and

CLOSEd_CUE_PATHs and this decision should be stated clearly in the device's MIDI Implementation documentation.

0A	RESET

0B GO_OFF

Starts a transition or fade of a cue to the off state. Transition time is determined by the cue in the Controlled Device.

If no Cue Number is specified, the current cue GOes OFF. If a Cue Number is specified, that cue GOes OFF.

In Controlled Devices with multiple Cue Lists, if no Cue Number is Specified, all currently active cues in Open Cue Lists GO OFF. If Q_number is sent without Q_list, all cues with a number identical to Q_number and which are in Open Cue Lists GO OFF.

For compatibility with Controlled Devices which do not automatically replace an existing cue with a new cue upon receipt of the GO command, Controllers should optionally prompt the programmer to simultaneously create a GO_OFF command.

0B	GO_OFF
<Q_number>	optional; required if Q_list is sent
00	delimiter
<Q_list>	optional; required if Q_path is sent
00	delimiter
<Q_path>	optional

10 GO/JAM_CLOCK

Starts a transition or fade to a cue simultaneous with forcing the clock time to the 'Go Time' if the cue is an 'Auto Follow' cue. Transition time is determined by the cue in the Controlled Device.

If no Cue Number is specified, the next cue in numerical sequence GOes and the clock of the appropriate Cue List JAMs to that cue's time. If the next cue in numerical sequence is a 'Manual' cue (i.e. if it has not been stored with a particular 'Go Time,' making it an 'Auto Follow' cue), the GO/JAM_CLOCK command is ignored.

If a Cue Number is specified, that cue Gop and the clock of the appropriate Cue List JAMs to the cue's time unless the cue is 'Manual' in which case no change occurs.

Rules for Controlled Devices with multiple Cue Lists are the same as for GO, above.

10	GO/JAM_CLOCK
<Q_number>	optional; required if Q_list is sent
00	delimiter
<Q_list>	optional; required if Q_path is sent
00	delimiter

```
 <Q_path>        optional
```

11 STANDBY_+

Places into standby position the next cue in numerical order after the cue currently in standby.

If Q_list is not sent, the Open Cue List containing the next cue in numerical order is used. If more than one Open Cue List have cues with an identical number then those cues will move to their respective standby positions.

If Q_list is sent in Standard Cue Number Form, only the next cue in the Cue List specified moves to the standby position.

```
 11              STANDBY_+
 <Q_list>        optional
```

12 STANDBY_-

Places into standby position the previous cue in numerical order prior to the cue currently in standby.

If Q_list is not sent, the Open Cue List containing the previous cue in numerical order is used. If more than one Open Cue List have cues with an identical number then those cues will move to their respective standby positions.

If Q_list is sent in Standard Form, only the previous cue in the Cue List specified moves to the standby position.

```
 12              STANDBY_-
 <Q_list>        optional
```

13 SEQUENCE_+

Places into standby position the next parent cue in numerical sequence after the cue currently in standby.

'Parent' refers to the integer value of the cue's number prior to the first decimal point (the "most significant number") For example, if cue 29.324.98.7 was in standby and the cues following were 29.325, 29.4, 29.7, 29.9.876, 36.7, 36.7.832, 36.8, 37., and 37.1, then cue 36.7 would be loaded to standby by SEQUENCE_+.

If Q_list is not sent, the Open Cue List containing the next cue in parental sequence is used. If more than one Open Cue List have cues with a completely identical number then those cues will move to their respective standby positions.

If Q_list is sent in Standard Form, only the next parent cue in the Cue List specified moves to the standby position.

```
 13              SEQUENCE_+
```

```
 <Q_list>        optional
```

14 SEQUENCE_-

Places into standby position the lowest numbered parent cue in the previous numerical sequence prior to the cue currently in standby.

'Parent' refers to the integer value of the cue's number prior to the first decimal point (the "most significant number") For example, if cue 37.4.72.18.5 was in standby and the cues preceding were 29.325, 29.4, 29.7, 29.9.876, 36.7, 36.7.832, 36.8, 37., and 37.1, then cue 36.7 would be loaded to standby by SEQUENCE_-.

If Q_list is not sent, the Open Cue List containing the previous parental sequence is used. If more than one Open Cue List have cues with identical lowest numbered parent cues in previous parental sequence then those cues will move to their respective standby positions.

If Q_list is sent in Standard Form, only the first parent cue in the previous sequence of the Cue List specified moves to the standby position.

```
 14              SEQUENCE_-
 <Q_list>        optional
```

15 START_CLOCK

Starts the 'Auto Follow' clock timer. If the clock is already running, no change occurs. The clock continues counting from the time value which it contained while it was Stopped.

If Q_list is not sent, the clocks in all Open Cue Lists Start simultaneously.

If Q_list is sent in Standard Form, only the clock in that Cue List Starts.

```
 15              START_CLOCK
 <Q_list>        optional
```

16 STOP_CLOCK

Stops the 'Auto Follow' clock timer. If the clock is already stopped, no change occurs. While the clock is stopped, it retains the time value which it contained at the instant it received the STOP command.

If Q_list is not sent, the clocks in all Open Cue Lists Stop simultaneously.

If Q_list is sent in Standard Form, only the clock in that Cue List Stops.

```
 16              STOP_CLOCK
 <Q_list>        optional
```

17 ZERO_CLOCK

Sets the 'Auto Follow' clock timer to a value of 00:00:00:00.00, whether or not it is running. If the clock is already stopped and Zeroed, no change occurs. ZERO_CLOCK does not affect the clock's running status.

If Q_list is not sent, the clocks in all Open Cue Lists Zero simultaneously.

If Q_list is sent in Standard Form, only the clock in that Cue List Zeros.

17	ZERO_CLOCK
<Q_list>	optional

18 SET_CLOCK

Sets the 'Auto Follow' clock timer to a value equal to the Standard Time sent, whether or not it is running. SET_CLOCK does not affect the clock's running status.

If Q_list is not sent, the clocks in all Open Cue Lists Set simultaneously.

If Q_list is sent in Standard Form, only the clock in that Cue List Sets.

18	SET_CLOCK
hr mn sc fr ff	Standard Time Specification
<Q_list>	optional

19 MTC_CHASE_ON

Causes the 'Auto Follow' clock timer to continuously contain a value equal to incoming MIDI Time Code. If no MTC is being received when this command is received, the clock remains in its current running or stopped status until MTC is received, at which time the clock continuously exhibits the same time as MTC. If MTC becomes discontinuous, the clock continues to display the last valid MTC message value received.

If Q_list is not sent, the clocks in all Open Cue Lists Chase simultaneously.

If Q_list is sent in Standard Form, only the clock in that Cue List Chases.

19	MTC_CHASE_ON
<Q_list>	optional

1A MTC_CHASE_OFF

Causes the 'Auto Follow' clock timer to cease Chasing incoming MIDI Time Code. When MTC_CHASE_OFF is received, the clock returns to running or stopped status according to its operating status at the instant MTC_CHASE_ON was received.

MTC_CHASE_OFF does not change the clock time value; i.e.

if the clock is stopped, it retains the last valid MTC message value received (or simply the most recent time in the clock register); if the clock is running, it continues to count from the most recent time in its register.

If Q_list is not sent, the clocks in all Open Cue Lists stop Chasing simultaneously.

If Q_list is sent in Standard Form, only the clock in that Cue List stops Chasing.

1A	MTC_CHASE_OFF
<Q_list>	optional

1B OPEN_CUE_LIST

Makes a Cue List available to all other commands and includes any cues it may contain in the current show.

When OPEN_CUE_LIST is received, the specified Cue List becomes active and cues in it can be accessed by normal show requirements. Q_list in Standard Form must be sent.

If the specified Cue List is already Open or if it does not exist, no change occurs.

1B	OPEN_CUE_LIST
<Q_list>	required

1C CLOSE_CUE_LIST

Makes a Cue List unavailable to all other commands and excludes any cues it may contain from the current show.

When CLOSE_CUE_LIST is received, the specified Cue List becomes inactive and cues in it cannot be accessed by normal show requirements, but the status of the cues in the list does not change. Q_list in Standard Form must be sent.

If the specified Cue List is already Closed or if it does not exist, no change occurs.

1C	CLOSE_CUE_LIST
<Q_list>	required

1D OPEN_CUE_PATH

Makes a Cue Path available to all other MIDI Show Control commands and to all normal show cue path access requirements as well.

When OPEN_CUE_PATH is received, the specified Cue Path becomes active and cues in it can be accessed by the Controlled Device. Q_path in Standard Form must be sent.

If the specified Cue Path is already Open or if it does not exist, no change occurs.

1D	OPEN_CUE_PATH
<Q_path>	required

1E CLOSE_CUE_PATH

Makes a Cue Path unavailable to all other MIDI Show Control commands and to all normal show cue path access requirements as well.

When CLOSE_CUE_PATH is received, the specified Cue Path becomes inactive and cues in it cannot be accessed by the Controlled Device. Q_path in Standard Form must be sent.

If the specified Cue Path is already Closed or if it does not exist, no change occurs.

1E	CLOSE_CUE_PATH
<Q_path>	required