

Eventide® Harmonizer™

MIDI SysEx Reference

This document describes the use of the MIDI SysEx protocols offered by the Eventide Harmonizer which are suitable for a PC or Mac based remote control system. Be aware that actual computer programming may be required to make use of them. These protocols may also be offered on future Eventide audio products, and are, in general, independent of software versions. The text below refers often to *Harmonizers*, but should be taken to apply to any of the supported systems.

This document assumes some familiarity with MIDI and the MIDI SysEx protocol, and the use of hex (hexadecimal) numbers.

These SysEx protocols are currently supported by the following Eventide Products:

- Orville V2.705 or later
- DSP7000/7500/4000B+ (all)
- Eclipse V1.1 or later

(This page was intentionally left blank.)

Table of Contents

MIDI SysEx Message Formats.....	1
Message Schematic.....	1
The Format of <Message-Data>.....	1
An Example.....	2
UserObjects.....	4
Overview.....	4
Schematic of UserObjects.....	4
The Various Types of UserObjects.....	5
COL “Collection”.....	5
NUM “Number”.....	6
STR “String”.....	6
CON “Constant”.....	6
INF “Information”.....	6
SET “Set Of Strings”.....	6
TRG “Trigger”.....	7
UserObject Trees.....	8
A Note about Keys.....	9
The Root Collection.....	9
SYSExc_BANKCHANGE.....	13
SYSExc_CARD_DUMP.....	14
SYSExc_CARD_WANT.....	15
SYSExc_ERROR.....	15
SYSExc_FILES_DUMP.....	16
SYSExc_FILES_WANT.....	17
SYSExc_INFO_DUMP.....	18
SYSExc_INFO_WANT.....	18
SYSExc_INTERNAL_DUMP.....	19
SYSExc_INTERNAL_WANT.....	20
SYSExc_KEYPRESS.....	21
SYSExc_OBJECTINFO_DUMP.....	22
SYSExc_OBJECTINFO_WANT.....	23
SYSExc_OK.....	25
SYSExc_PARAMETERS_DUMP.....	26
SYSExc_PARAMETERS_WANT.....	27
SYSExc_PROGRAM_WANT.....	28
SYSExc_SCREEN_DUMP.....	29
SYSExc_SCREEN_WANT.....	30
SYSExc_SETUP_DUMP.....	31
SYSExc_SETUP_WANT.....	32
SYSExc_SIGDBASE_DUMP.....	33
SYSExc_SIGDBASE_WANT.....	33
SYSExc_SIGFILE_DUMP.....	34
SYSExc_SIGFILE_DUMP_REMOTE.....	35
SYSExc_SIGFILE_WANT.....	35
SYSExc_SIGFILE_WANT_QUICK.....	36
SYSExc_VALUE_DUMP.....	37
SYSExc_VALUE_PUT.....	38
SYSExc_USEROBJECT.....	39
Appendix A: Summary of Formats.....	40
F_AsciiDecimal.....	40
F_AsciiFloat.....	40
F_AsciiHex.....	40
F_AsciiSimpleString.....	40
F_AsciiString.....	40
F_AsciiText.....	41

Table of Contents

F_BinaryBytes	41
F_BinaryNibbles.....	41
Appendix B: List of SYSEX_ Messages.....	42

MIDI SysEx Message Formats

All MIDI SysEx messages have the same general format, which is dictated by The MIDI Specification. Even if one uses a serial cable to connect the Harmonizer to the computer, the stream of bytes forming the communications still adheres to this specification. In addition to the MIDI specification, Eventide has ensured that all SysEx messages sent to or received from a Harmonizer follows a consistent extended format.

Message Schematic

The format of every message set to a Harmonizer or received from a Harmonizer follows the same basic schematic:

```
0xF0 <Eventide> <H4000> <ID> <message-code> <message-data> 0xF7
```

Let's take a more detailed look at each part of this schematic:

0xF0	The leading byte of all MIDI SysEx messages (decimal 240).
<Eventide>	The byte 0x1C (decimal 28). All MIDI SysEx messages sent to or received from Eventide equipment will have this byte as the second byte in the message. (The particular value 0x1C was assigned by the body that governs the MIDI specification to identify equipment built by Eventide, Inc.)
<H4000>	The byte 0x70 (decimal 112). All MIDI SysEx messages sent to or received from a modern Harmonizer will contain this byte as the third byte in the message. It is used by Eventide to identify MIDI SysEx messages intended for use by Harmonizer 4000 series or better.
<ID>	A byte containing the ID of the Harmonizer for which this message is intended, or from which this message was received. Messages set through MIDI with an <ID> of zero (0x00) will cause all Harmonizers to listen (and possibly respond to) the message.
<message-code>	A byte that determines the specific message being sent to or from the Harmonizer. It can be any one of the SYSEXC_ values explained later in this document.
<message-data>	Zero or more bytes in a format that is specific to the particular <message-code>.
0xF7	The trailing byte of all MIDI SysEx messages (decimal 247).

The Format of <Message-Data>

While the exact format of the <message-data> portion of any Harmonizer SysEx message is determined by the value of the <message-code> byte, there are some general guidelines that are followed. There is also some useful background information which will help explain why things are organized the way they are.

One very important limitation in the MIDI SysEx specification is that it prohibits any SysEx message from containing a byte with the high-bit set, except for the leading 0xF0 and the trailing 0xF7. This means that if binary data is to be transferred across MIDI, it must some how be encoded so that this high-bit is never used. When dealing with SysEx messages for a Harmonizer, you will encounter three distinctly different solutions to this problem:

1. If the binary data being transferred always results in bytes less than 0x80, then the high-bit will never be set, and the binary data can be transferred as an un-encoded sequence of bytes. For example, <message-code> is a byte that directly encodes an identifier of the message being sent.
2. Encode everything in ASCII. This is the easiest to understand solution, although it does require careful attention when dealing with integer values: sometimes integers are transferred as a sequence of ASCII characters representing hex digits, other times as ASCII characters representing decimal digits. So the binary value 0x94 could be transferred in ASCII as the two bytes 0x39 0x34 (which are the ASCII characters for "9" and "4", respectively) if we encode in hex. Or, if we encode in decimal, the bytes transferred would be 0x31 0x34 0x38 ("1" "4" and "8", or "148", which is the decimal equivalent of hex 94). When this method is being used, spaces (the byte 0x20) will usually be used to delimit one number from another.
3. Break each binary byte into two separate nibbles*. For example, the byte 0x94 could be encoded as the two bytes 0x09 and 0x04.

* A nibble is a 4-bit number, just like a byte is an 8-bit number.

Since these different ways of formatting information can get confusing, this document names each of these formats. For example, #3 is called the F_BinaryNibbles format. See *Appendix A: Summary of Formats* for details.

An Example

As a manner of example, we will pretend to send a SYSEX_C_VALUE_PUT message and take a careful look at the encoding of the message and at the encoding of the response. At this point, our goal is mainly to understand the general MIDI SysEx format, not the specifics of the SYSEX_C_VALUE_PUT message.

Sample SysEx Message

Let's pretend that we are going to send our Harmonizer a SYSEX_C_VALUE_PUT message. This message, as documented later in this document, has the following schematic:

```
SYSEX_C_VALUE_PUT <key> <value>
```

Let's pretend we want to set the value of <key> 0x1000 to (decimal) 3.4 (let's not concern ourselves with what this might actually mean for now):

```
SYSEX_C_VALUE_PUT 0x1000 3.4
```

Let's now take a look at formatting this message the way a Harmonizer expects.

Interpretation of Bytes* in SysEx Message

First, all MIDI SysEx messages must begin with the standard format we described above:

```
F0    SysEx start byte
1C    Eventide, Inc.
70    H4000
```

If we know the <ID> of the machine we wish to talk to, we could use that number here instead of zero. That way we wouldn't be broadcasting the message to more than one Harmonizer (should we be so lucky!).

```
00    ID of Harmonizer—zero means all Harmonizers will listen
```

The next byte is the message we wish to send, in our case, the byte that corresponds to SYSEX_C_VALUE_PUT.

```
2D    SYSEX_C_VALUE_PUT
```

According to the documentation of SYSEX_C_VALUE_PUT, the <key> is sent as a hex number encoded in ASCII:

```
31    1    <key>, which is a hex number encoded in ASCII = "1000"
30    0
30    0
30    0
```

When dealing with ASCII, numbers are usually separated by a space:

```
20    a space
```

According to the documentation of SYSEX_C_VALUE_PUT, the exact format of <value> depends on the kind of <key>. Let's pretend <key> 0x1000 is a NUM userobject, in which case <value> would be the floating point decimal number encoded in ASCII:

```
33    3    floating point decimal <value> in ASCII = "3.4", assuming this is a NUM userobject
2E    .
34    4
```

The last byte of a message is always the same...

```
F7    SysEx end byte
```

* Note that we could also look at the SysEx message as a sequence of bytes displayed in decimal, which would ultimately result in:

```
(decimal) 240 28 112 0 45 49 48 48 48 32 46 52 247
```

But when working with binary data, hex is much more convenient and will be used exclusively for the rest of this document when dealing with bytes that constitute a message.

Sample SysEx Message as Bytes

So we now have the set of bytes to send via MIDI to the Harmonizer.

```
F0 1C 70 00 2D 31 30 30 30 20 33 2E 34 F7
```

The Harmonizer will recognize this message and (if all goes well) will send an appropriate response.

Response SysEx Message as Bytes

After sending the above message, let's pretend we received the following bytes in response.

```
F0 1C 70 01 2E 31 30 30 30 20 33 2E 34 F7
```

Interpretation of Bytes in Response SysEx Message

We can use the reverse procedure to interpret these bytes, and extract whatever information we desire from the message.

F0	SysEx start byte
1C	Eventide, Inc.
70	H4000
01	ID of Harmonizer
2E	SYSEXC_VALUE_DUMP
31	1 <key>, the hex number 0x1000 encoded in ASCII = "1000"
30	0
30	0
30	0
20	a space
33	3 floating point decimal <value> in ASCII = "3.4"
2E	.
34	4
F7	SysEx end byte

Received SysEx Message

So the received message was a SYSEXC_VALUE_DUMP, summarized as:

```
SYSEXC_VALUE_DUMP 1000 3.4
```

where <key> was the hex number 0x1000, and <value> was the floating point decimal value 3.4.

UserObjects

If you SysEx message SYSEXC_PARAMETERS_WANT/DUMP, SYSEXC_OBJECTINFO_WANT/DUMP, or SYSEXC_VALUE_PUT/DUMP, you will need to understand what userobjects are, and how decipher them.

Overview

The user interface of the Harmonizer operating system is made up of *userobjects*. These may display a value, provide the functionality of a push-button, or act as a menu page. For example, the Harmonizer's display of parameters for the "Oscillator (440)" patch on the Orville (Bank 64 "Utilities", program number 15), looks like:

B: Oscillator (440)		oscillator parms	
level :	-20.0 db	duty :	50 %
freq :	440 hz	fm rate :	1.0 hz
fmod :	0 hz	fm shape :	sine
shape :	sine	fm duty :	50 %
osc		info	

Each of the visual elements displayed is internally represented by a userobject. All of the visual elements share some common properties, such as some text to display. Some have additional properties, such as a value which gets formatted and inserted into the text for display.

Schematic of UserObjects

All userobjects adhere to the following schematic when represented as a string of characters. Such strings are returned by SYSEXC_PARAMETERS_DUMP and SYSEXC_OBJECTINFO_DUMP messages.

```
<type> <subtype> <key> <parent-key><statement> <tag> [<type-specific-information>]
```

Details

Name	Format	Description
<type>	F_AsciiSimpleString	The kind of userobject. Possible values are: COL, NUM, STR, CON, INF, SET, TRG, all of which are detailed below.
<subtype>	F_AsciiSimpleString	Userobjects of a particular <type> also have a <subtype>. This can typically be ignored.
<key>	F_AsciiHex	Each userobject is identified by a unique ID, known as the <key>.
<parent-key>		Userobjects are arranged into a tree, so every userobject has a parent. The <parent-key> is the <key> of the userobject that contains this userobject. See below for more on userobject trees.
<statement>	F_AsciiString	This is the text to display, typically with %...f or %...s embedded in it to indicate where the value of the parameter should be inserted, and how that parameters should be formatted.
<tag>	F_AsciiString	This is a short (eight character) string which is displayed on buttons, and is really only used by COL userobjects, and is often empty for other <type>s of userobjects.
<type-specific-information>	See below.	Each <kind> of userobject has information which is specific to that <kind>, and this information would start at this location in the string of characters representing this userobject. See below for details of this information.

Also note the following conventions:

1. When more than one userobject is returned, they are separated by a carriage return line feed sequence (i.e. `\r\n` in C/C++).
2. There are special considerations for strings. These formatting conventions for strings are referred to as F_AsciiString formatting throughout this document; details can be found in *Appendix A: Summary of Formats*.

UserObjects

- All strings containing spaces are placed in single quotes (i.e. apostrophes, or '). Strings not containing spaces generally are not enclosed in single quotes.
 - An empty string is represented as two single quotes (i.e. ' '), which is not the same as a double quote character (").
 - If a string contains a single quote ('), then the string is wrapped with double quotes ("). For example:
0 4010007 0 "Clrmtn's NemWhipper" NWhip 6
3. Integral values are represented in hex (e.g. keys, index for SET), which is known as F_AsciiHex formatting. See SYSEXC_VALUE_PUT for one notable exception that uses F_AsciiDecimal formatting.
 4. Floating-point numbers are represented in decimal (values for NUM and CON), which is known as F_AsciiFloat formatting.

Example

For example, the highlighted parameter “level: -20.0 db” in the above picture is represented by a userobject that can be represented something like:

```
NUM 0 80d0001 8030001 'level: %5.1f db' '' -20 -96 20 0.1
```

Let’s pick this userobject apart to better understand how the Harmonizer makes use of them:

NUM	The <type>. NUM is a numeric (floating point) value.
0	The <subtype>. Typically is not interesting, so just ignore.
80d0001	The <key>. A hex number which uniquely identifies this userobject.
8030001	The <parent-key>. A hex number which is the key of the <i>parent</i> of this userobject.
'level: %5.1f db'	The <statement>. This is the text to display, typically with %...f or %...s embedded in it to indicate where the value of the parameter should be inserted.
''	The <tag>. Only The use of <tag> depends upon <type>. Here, the two single quotes here indicate that the <tag> is empty.

These fields are unique to the <type> NUM:

-20	The <current-value>. This is the currently displayed value (i.e. -20.0 db).
-96	The <minimum>. This is the minimum value that this “level” parameter can have.
20	The <maximum>. This is the maximum value that this “level” parameter can have.
0.1	The <resolution>. This is how much is added to or subtracted from the parameter’s current value when the input knob is spun. That is, you can change the value of the “level” parameter in units of 0.1 db.

The Various Types of UserObjects

COL “Collection”

A *collection* acts as a container for two or more userobjects. Beyond this it has no data. If the <tag> is blank, it implies that this userobject is not intended to be displayed—it will serve the function of *ganging* either multiple controls to bind them together, or multiple menu pages, to create a *stacked* menu. (See the Harmonizer User Manual for explanation of these terms.) If the <tag> is non-blank, it means that the collection should be viewed as a menu page, with its contained userobjects displayed upon the page. In this instance, the statement might be used as a title for the menu page. Each object contained in the collection is known as a *subobject*. A subobject refers to their container as their *parent*.

Schematic

```
COL <subtype> <key> <parent-key> <statement> <tag> <number-of-subobjects>
```

Details

<number-of-subobjects> F_AsciiHex This is a number that indicates how many subobjects are part of this collection (i.e. how many subobjects have <key> as their <parent-key>).

NUM “Number”

A *number* userobject contains a floating point numeric value. The user can both read and write this value.

Schematic

NUM <subtype> <key> <parent-key> <statement> <tag> <current-value> <minimum> <maximum>
<resolution>

Details

<current-value>	F_AsciiFloat	Contains the current value of this parameter as a decimal, floating-point number. This value is usually substituted into the <statement> where the %...f is found.
<minimum>	F_AsciiFloat	The minimum value this parameter is allowed to take.
<maximum>	F_AsciiFloat	The maximum value this parameter is allowed to take.
<resolution>	F_AsciiFloat	The units by which <current-value> should be adjusted for each click of the wheel or a button.

STR “String”

A *string* userobject contains a text value. The user can both read and write this value.

Schematic

STR <subtype> <key> <parent-key> <statement> <tag> <current-value>

Details

<current-value>	F_AsciiString	Contains the string to be substituted into the <statement> where the %...s is found.
-----------------	---------------	--

CON “Constant”

A *constant* userobject contains a floating point numeric value. The user can only read this value; it cannot be changed directly. Note that a constant value may change as a result of changes to other userobjects, or as a result of changes in the system’s state.

Schematic

CON <subtype> <key> <parent-key> <statement> <tag> <current-value>

Details

<current-value>	F_AsciiFloat	Contains the current value of this parameter as a decimal, floating-point number. This value is usually substituted into the <statement> where the %...f is found.
-----------------	--------------	--

INF “Information”

An *information* userobject contains a text value. The user can only read this value, it cannot be changed directly. An information value will not usually changes as a result of changes to other userobjects.

Schematic

INF <subtype> <key> <parent key> <statement> <tag> <current-value>

Details

<current-value>	F_AsciiString	Contains the string to be substituted into the <statement> where the %...s is found.
-----------------	---------------	--

SET “Set Of Strings”

A *set* userobject contains a number of text strings, one of which is selected at any given time. Thus, its data is the index of the selected string, which can be read or written.

Schematic

SET <subtype> <key> <parent-key> <statement> <tag> <current-index> <selected-string> <number-of-strings> <string-text0> <string-text1>...

Details

<code><current-index></code>	F_AsciiHex	A value that indicates which string from <code><string-textn></code> is the current string to display (i.e. to substitute into the <code><statement></code> where the %...s is found).
<code><selected-string></code>	F_AsciiString	The current string to display; identical to <code><string-textn></code> , where n is the <code><current-index></code> .
<code><number-of-strings></code>	F_AsciiHex	A value that indicates the number of <code><string-textn></code> strings which follow. This also indicates the valid values of <code><current-index></code> , which range from zero to <code><number-of-strings></code> minus one.
<code><string-textn></code>	F_AsciiString	The list of strings which would correspond to the different possible values of <code><current-index></code> . Useful to fill a listbox with in order to display the different choices to a user.

TRG “Trigger”

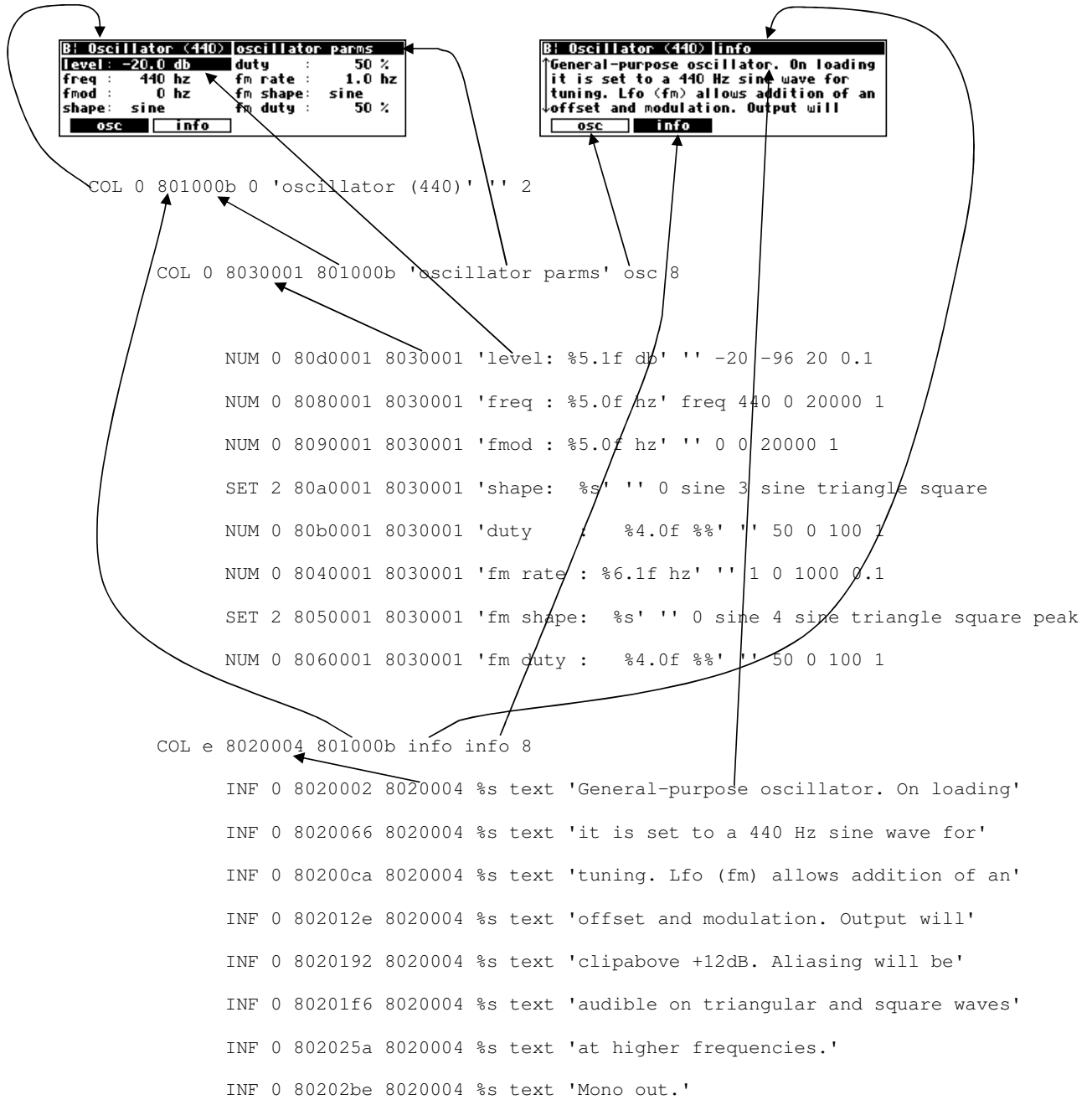
A *trigger* userobject represents a push-button. Note that `<statement>` is the caption displayed by the Harmonizer for this button, and that `<tag>` is not present.

Schematic

TRG `<subtype>` `<key>` `<parent-key>` `<statement>`

UserObject Trees

The userobjects in a Harmonizer are arranged as a *tree*, with collections acting as the branching points. Consider, for example, the PARAMETER display of the preset below, consisting of two menu pages, each one of which is shown. The drawing below shows the tree structure and the contributions made by data and attributes of the various userobjects to the Harmonizer screens.



The entire user interface of a Harmonizer may be accessed via its root object, which has a key value of 0. This is a collection containing the roots of the tree for each main mode, for example PROGRAM, SETUP, etc. The contents of the tree may vary between different products and software versions, but as long as it is reloaded each time it is accessed, everything will stay in step.

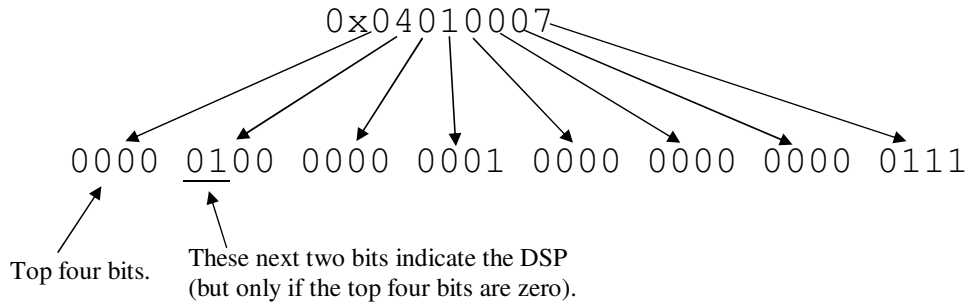
A Note about Keys

As we've mentioned before, a *<key>* is a 32-bit integer that uniquely identifies a userobject. It is always represented in hex, usually with leading zeros dropped. It turns out that keys are not totally opaque: the top four bits are zero for a *<key>* that represents a parameter of a loaded program; the next two bits tell you whether that program is loaded into DSP A (01), DSP B* (10) or DSP C† (11).

For example,

```
COL 0 4010007 0 "Clrmtn's NemWhipper" NWhip 6
```

This userobject's key is 0x04010007. Notice that the leading zero was dropped. The highest four bits are zero, indicating that this userobject represents some parameter of a program. The next two bits are 01 in binary, and thus, the program is running on DSP A.



The Root Collection

Here is an example of the contents of the main, root collection.

```
(i) COL 0 4010007 0 "Clrmtn's NemWhipper" NWhip 6
(ii) COL 0 801000b 0 Oscillator-440 '' 3
(iii) COL 0 10010000 0 'setup functions' setup d
(iv) 8 0 10040000 0 '' ''
(v) COL 0 10020000 0 'program functions' program 9
(vi) COL 0 10030000 0 'level functions' level 4
(vii) COL 0 10030500 0 'bypass functions' bypass 4
```

Some notes about the above:

- (i) and (ii) are the PARAMETER trees for the A and B machines. Note the double quotes in (i), while the apparent double quotes in (ii) are in fact a pair of single quotes as a place mark for the absent tag.
- (iii) is the SETUP tree.
- (iv) is an undocumented type and should be ignored.
- (v) is the tree for the PROGRAM functions.
- (vi) gives access to the LEVEL functions.
- (vii) is the BYPASS root.

The above ordering and contents should not be assumed, but should be scanned every time communication is established between the remote control and a Harmonizer. For example, a product where the BYPASS button acts directly (DSP7000), rather than via a menu as on Orville, might have a trigger (TRG) rather than a collection (COL) at that point.

* This exists only in the Orville.

† This exists only in the Eclipse, and refers to the modulator block.

Part II:

Reference of

`SysExc_`

Messages

SysExc_BankChange

Message Schematic

SYSEXC_BANKCHANGE <flag-external> <bank-number>

Description

This message is received from a Harmonizer as the result of parameter changes, if the Harmonizer is set up to do so.

These messages can then be recorded by a MIDI sequencer, and when replayed, will cause the control changes to be duplicated. When sent, the Harmonizer will respond with a SYSEXC_OK or SYSEXC_ERROR message.

Details

Name	Format	Description
SYSEXC_BANKCHANGE	F_BinaryBytes	The value 0x03.
<flag-external>	F_BinaryNibbles	The 8-bit number 0 for an internal bank, or 1 for an external bank.
<bank-number>	F_BinaryNibbles	An 8-bit unsigned integer indicating the bank number.

Notice that each of these 8-bit numbers is transmitted in F_BinaryNibbles, which means each 8-bit number will take two bytes in the message.

Sample

Let's pretend we received the following message from a Harmonizer.

Bytes in SysEx Message

F0 1C 70 01 03 00 01 03 02 F7

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 03 SYSEXC_BANKCHANGE
 00 high nibble of <flag-external>
 01 low nibble of <flag-external>
 03 high nibble of <bank-number>
 02 low nibble of <bank-number>
 F7 SysEx end byte

Schematic

SYSEXC_BANKCHANGE 0x01 0x32

Thus, the Harmonizer is notifying us that it has changed to *external* bank number (decimal) 50.

SYSEX_CARD_DUMP

Message Schematic

SYSEX_CARD_DUMP <size-of-block> <block> <checksum>

Description

This message is received from a Harmonizer when requested by SYSEX_CARD_WANT, and contains a <block> which is data that represents the complete contents of the memory card inserted into the Harmonizer.

This message can also be sent to a Harmonizer, causing to the contents of the memory card inserted into the Harmonizer with the contents of this message.

Details

Name	Format	Description
SYSEX_CARD_DUMP	F_BinaryBytes	The value 0x13.
<size-of-block>	F_BinaryNibbles	A 32-bit integer (sent as nibbles) which indicates the size of <block>.
<block>	F_BinaryNibbles	A sequence of 8-bit integers (sent as nibbles) which are the files on the Harmonizer.
<checksum>	F_BinaryNibbles	An 8-bit integer (sent as two nibbles) which is a 1-byte checksum. This byte added to all of the bytes (after the bytes have been reconstituted from nibbles), including the size bytes, should equal zero.

Sample

Bytes in Received SysEx Message

F0 1C 70 01 13 00 00 00 00 00 01 00 00 ... 0A 01 F7

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 13 SYSEX_CARD_DUMP
 00 <size-of-block> = 0x00000100
 00
 00
 00
 00
 01
 00
 00
 0V High nibble of 1st byte of <block> (first byte = 0xVW)
 0W Low nibble of 1st byte of <block>
 ... Rest of nibbles that make up <block>
 0Y High nibble of last byte of <block> (last byte = 0xYZ)
 0Z Low nibble of last byte of <block>
 0A High nibble of <checksum> = 0xA1
 01 Low nibble of <checksum>
 F7 SysEx end byte

Note that the bytes comprising $\langle \text{size-of-block} \rangle + \langle \text{block} \rangle + \langle \text{checksum} \rangle$ should equal zero. Thus, in our example, $0 \times 00 + 0 \times 00 + 0 \times 01 + 0 \times 00 + 0 \times VW + \dots + 0 \times YZ + 0 \times A1$ should equal zero.

SYSExc_CARD_WANT

Message Schematic

SYSExc_CARD_WANT

Description

This message is sent to a Harmonizer to cause it to respond with SYSExc_CARD_DUMP, which will dump the memory card currently inserted into the Harmonizer. It takes quite some time for the Harmonizer to transmit the SYSExc_CARD_DUMP.

Details

Name	Format	Description
SYSExc_CARD_WANT	F_BinaryBytes	The value 0×14 .

Sample

Schematic

SYSExc_CARD_WANT

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 14 SYSExc_CARD_WANT
 F7 SysEx end byte

Bytes in Corresponding SysEx Message

F0 1C 70 01 14 F7

SYSExc_ERROR

Message Schematic

SYSExc_ERROR $\langle \text{message} \rangle$

Description

This message is received from a Harmonizer in response to assorted commands. This message indicates that an error occurred with the last command.

Details

Name	Format	Description
SYSExc_SIGDBASE_DUMP	F_BinaryBytes	The value $0 \times 0D$.
$\langle \text{message} \rangle$	F_AsciiText	Human-readable text that provides information about the error.

Sample

Bytes in Received SysEx Message

```
F0 1C 70 01 0D ... F7
```

Interpretation

```
F0    SysEx start byte
1C    Eventide, Inc.
70    H4000
01    ID of Harmonizer
0D    SYSEXC_ERROR
...   A whole bunch of ASCII characters, which comprise a human-readable message.
F7    SysEx end byte
```

SYSEXC_FILES_DUMP

Message Schematic

```
SYSEXC_FILES_DUMP <size-of-block> <block> <checksum>
```

Description

This message is received from a Harmonizer when requested by SYSEXC_FILES_WANT, and contains a *<block>* which is data that represents the set of files on the Harmonizer.

This message can also be sent to a Harmonizer, causing to replace its files with the contents of this message.

Details

Name	Format	Description
SYSEXC_FILES_DUMP	F_BinaryBytes	The value 0x0F.
<i><size-of-block></i>	F_BinaryNibbles	A 32-bit integer (sent as nibbles) which indicates the size of <i><block></i> .
<i><block></i>	F_BinaryNibbles	A sequence of 8-bit integers (sent as nibbles) which are the files on the Harmonizer.
<i><checksum></i>	F_BinaryNibbles	An 8-bit integer (sent as two nibbles) which is a 1-byte checksum. This byte added to all of the bytes (after the bytes have been reconstituted from nibbles), including the size bytes, should equal zero.

Sample

Bytes in Received SysEx Message

```
F0 1C 70 01 0F 00 00 00 00 00 01 00 00 ... 0A 01 F7
```

Interpretation

```
F0    SysEx start byte
1C    Eventide, Inc.
70    H4000
01    ID of Harmonizer
0F    SYSEXC_FILES_DUMP
00    <size-of-block> = 0x00000100
00
00
00
00
```

SysExc_Files_Want

00	
01	
00	
00	
0V	High nibble of 1st byte of <i><block></i> (first byte = 0xVW)
0W	Low nibble of 1st byte of <i><block></i>
. . .	Rest of nibbles that make up <i><block></i>
0Y	High nibble of last byte of <i><block></i> (last byte = 0xYZ)
0Z	Low nibble of last byte of <i><block></i>
0A	High nibble of <i><checksum></i> = 0xA1
01	Low nibble of <i><checksum></i>
F7	SysEx end byte

Note that the bytes comprising *<size-of-block>* + *<block>* + *<checksum>* should equal zero. Thus, in our example, 0x00 + 0x00 + 0x01 + 0x00 + 0xVW + ... + 0xYZ + 0xA1 should equal zero.

SYSEXC_FILES_WANT

Message Schematic

SYSEXC_FILES_WANT

Description

This message is sent to a Harmonizer to cause it to respond with SYSEXC_FILES_DUMP, which will dump the Harmonizer's current presets. It takes quite some time for the Harmonizer to transmit the SYSEXC_FILES_DUMP.

Details

<u>Name</u>	<u>Format</u>	<u>Description</u>
SYSEXC_FILES_WANT	F_BinaryBytes	The value 0x10.

Sample

Schematic

SYSEXC_FILES_WANT

Interpretation

F0	SysEx start byte
1C	Eventide, Inc.
70	H4000
01	ID of Harmonizer
10	SYSEXC_FILES_WANT
F7	SysEx end byte

Bytes in Corresponding SysEx Message

F0 1C 70 01 10 F7

SYSExc_INFO_DUMP

Message Schematic

SYSExc_INFO_DUMP <information>

Description

This message is received from a Harmonizer when requested by SYSExc_INFO_WANT, and contains text which provides information about the Harmonizer, such as OS version, ROM version, amount of memory, etc., all in a human-readable format.

Details

Name	Format	Description
SYSExc_INFO_DUMP	F_BinaryBytes	The value 0x19.
<information>	F_AsciiText	Human-readable text that provides information about the Harmonizer.

Sample

Bytes in Received SysEx Message

F0 1C 70 01 19 ... F7

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 19 SYSExc_INFO_DUMP
 ... A whole bunch of ASCII characters, which comprise <information>.
 F7 SysEx end byte

SYSExc_INFO_WANT

Message Schematic

SYSExc_INFO_WANT

Description

This message is sent to a Harmonizer to cause it to respond with SYSExc_INFO_DUMP, which will dump information about the Harmonizer, such as ROM versions, product version, etc.

Details

Name	Format	Description
SYSExc_INFO_WANT	F_BinaryBytes	The value 0x1A.

Sample

Schematic

SYSExc_INFO_WANT

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 1A SYSEXC_INFO_WANT
 F7 SysEx end byte

Bytes in Corresponding SysEx Message

F0 1C 70 01 1A F7

SYSEXC_INTERNAL_DUMP

Message Schematic

SYSEXC_INTERNAL_DUMP <size-of-block> <block> <checksum>

Description

This message is received from a Harmonizer when requested by SYSEXC_INTERNAL_WANT, and contains a <block> which is data that represents the complete contents of the internal non-volatile RAM on the Harmonizer.

This message can also be sent to a Harmonizer, causing to replace its non-volatile RAM with the contents of this message.

Details

Name	Format	Description
SYSEXC_INTERNAL_DUMP	F_BinaryBytes	The value 0x11.
<size-of-block>	F_BinaryNibbles	A 32-bit integer (sent as nibbles) which indicates the size of <block>.
<block>	F_BinaryNibbles	A sequence of 8-bit integers (sent as nibbles) which are the files on the Harmonizer.
<checksum>	F_BinaryNibbles	An 8-bit integer (sent as two nibbles) which is a 1-byte checksum. This byte added to all of the bytes (after the bytes have been reconstituted from nibbles), including the size bytes, should equal zero.

Sample

Bytes in Received SysEx Message

F0 1C 70 01 11 00 00 00 00 00 01 00 00 ... 0A 01 F7

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 11 SYSEXC_INTERNAL_DUMP
 00 <size-of-block> = 0x00000100
 00
 00
 00
 00
 00
 01

SysExc_Internal_Want

00
00
0V High nibble of 1st byte of <block> (first byte = 0xVW)
0W Low nibble of 1st byte of <block>
. . . Rest of nibbles that make up <block>
0Y High nibble of last byte of <block> (last byte = 0xYZ)
0Z Low nibble of last byte of <block>
0A High nibble of <checksum> = 0xA1
01 Low nibble of <checksum>
F7 SysEx end byte

Note that the bytes comprising <size-of-block> + <block> + <checksum> should equal zero. Thus, in our example, 0x00 + 0x00 + 0x01 + 0x00 + 0xVW + ... + 0xYZ + 0xA1 should equal zero.

SYSEXC_INTERNAL_WANT

Message Schematic

SYSEXC_INTERNAL_WANT

Description

This message is sent to a Harmonizer to cause it to respond with SYSEXC_INTERNAL_DUMP, which will dump the Harmonizer's non-volatile RAM. It takes quite some time for the Harmonizer to transmit the SYSEXC_INTERNAL_DUMP.

Details

Name	Format	Description
SYSEXC_INTERNAL_WANT	F_BinaryBytes	The value 0x12.

Sample

Schematic

SYSEXC_INTERNAL_WANT

Interpretation

F0 SysEx start byte
1C Eventide, Inc.
70 H4000
01 ID of Harmonizer
12 SYSEXC_INTERNAL_WANT
F7 SysEx end byte

Bytes in Corresponding SysEx Message

F0 1C 70 01 12 F7

SYSExc_KEYPRESS

Message Schematic

SYSEXC_KEYPRESS <key-code>

Responses

None.

Description

This message is sent to a Harmonizer in order to emulate the User pressing a key on the Harmonizer's front panel.

Details

Name	Format	Description
SYSEXC_KEYPRESS	F_BinaryBytes	The value 0x01.
<key-code>	F_BinaryNibbles	A 32-bit integer represented as nibbles (thus taking eight bytes in the message). One bit corresponds to each key, with the bit normally being set. While a button is depressed, the corresponding bit becomes zero, returning to one once the button is released. Hence, <key-code> can indicate when more than one key is depressed.

Here is a table of the possible <key-codes>:

Key Name	<key-code>	Formatted as Bytes to Transmit	Notes
BYPASS	0xFFFFFFFF	0F 0F 0F 0F 0F 0D 0F 0F	
SOFT1	0xFBFFFFFF	0F 0B 0F 0F 0F 0F 0F 0F	The "soft keys" under the display.
SOFT2	0xFFFBFFFF	0F 0F 0F 0B 0F 0F 0F 0F	
SOFT3	0xFFFFBFFF	0F 0F 0F 0F 0F 0B 0F 0F	
SOFT4	0xFFFFFFF	0F 0F 0F 0F 0F 0F 0F 0B	
DSP A/B	0xFDFFFFFF	0F 0D 0F 0F 0F 0D 0F 0F	<i>Only on Orville.</i>
PROGRAM	0xF7FFFFFF	0F 07 0F 0F 0F 0F 0F 0F	Same as holding down PARAMETER.
PARAMETER	0xFFF7FFFF	0F 0F 0F 07 0F 0F 0F 0F	
PATCH	0xFFFFFFF7	0F 0F 0F 0F 0F 0F 0F 07	
SELECT	0xFFFFFFF	0F 0F 0F 0F 0F 0E 0F 0F	
UP (▲)	0xFEFFFFFF	0F 0E 0F 0F 0F 0D 0F 0F	
DOWN (▼)	0xFFEFFFF	0F 0F 0F 0E 0F 0D 0F 0F	
PREVIOUS (◀)	0xFEFFFFFF	0F 0F 0F 0E 0F 0F 0F 0F	
NEXT (▶)	0xFEFFFFFF	0F 0E 0F 0F 0F 0F 0F 0F	
0 (ZERO)	0xFFFFFFF	0F 0F 0E 0F 0F 0F 0F 0F	
1 (ONE)	0x7FFFFFF	07 0F 0F 0F 0F 0F 0F 0F	
2 (TWO)	0xFF7FFFF	0F 0F 07 0F 0F 0F 0F 0F	
3 (THREE)	0xFFFF7FFF	0F 0F 0F 0F 07 0F 0F 0F	
4 (FOUR)	0xBFFFFFF	0B 0F 0F 0F 0F 0F 0F 0F	
5 (FIVE)	0xFFBFFFF	0F 0F 0B 0F 0F 0F 0F 0F	
6 (SIX)	0xFFFFBFFF	0F 0F 0F 0F 0B 0F 0F 0F	
7 (SEVEN)	0xDFFFFFF	0D 0F 0F 0F 0F 0F 0F 0F	
8 (EIGHT)	0xFFDFFFF	0F 0F 0D 0F 0F 0F 0F 0F	
9 (NINE)	0xFFFFDFFF	0F 0F 0F 0F 0D 0F 0F 0F	
DOT (.)	0xEFFFFFF	0E 0F 0F 0F 0F 0F 0F 0F	

SysExc_ObjectInfo_Dump

Key Name	<key-code>	Formatted as Bytes to Transmit	Notes
MINUS (-)	0xFFFFFFFF	0F 0F 0F 0F 0E 0F 0F 0F	
INC (↑)	0xFFFFFFFF7F	0F 0F 0F 0F 0F 0F 07 0F	
DEC (↓)	0xFFFFFFFFBF	0F 0F 0F 0F 0F 0F 0B 0F	
CXL	0xFFFFFFFFDF	0F 0F 0F 0F 0F 0F 0D 0F	
ENT	0xFFFFFFFFEF	0F 0F 0F 0F 0F 0F 0E 0F	
LEVELS	0xFFFFFFFFFD	0F 0F 0F 0F 0F 0F 0F 0D	
SETUP	0xFFFFF7FF	0F 0F 0F 0F 0F 07 0F 0F	
USER1	0xFDFFFFFF	0F 0D 0F 0F 0F 0F 0F 0F	
USER2	0xFFFDFFFF	0F 0F 0F 0D 0F 0F 0F 0F	

Sample

Let's send a message that presses the LEVELS button.

Schematic

```
SYSEXC_KEYPRESS LEVELS
```

Bytes in Corresponding SysEx Message

```
F0 1C 70 01 01 0F 0F 0F 0F 0F 0F 0F 0D F7
```

Interpretation

```
F0    SysEx start byte
1C    Eventide, Inc.
70    H4000
01    ID of Harmonizer
01    SYSEXC_KEYPRESS
0F    <key-code> for LEVELS (0xFFFFFFFFFD as F_BinaryNibbles)
0F
0F
0F
0F
0F
0F
0D
F7    SysEx end byte.
```

SYSEXC_OBJECTINFO_DUMP

Message Schematic

```
SYSEXC_OBJECTINFO_DUMP <userobjects>
```

Description

This message is received from a Harmonizer as the result of sending the Harmonizer a SYSEXC_OBJECTINFO_WANT message. This message is nearly the same as SYSEXC_PARAMETERS_DUMP; see SYSEXC_OBJECTINFO_WANT for details about the differences between these messages.

Details

Name	Format	Description
SYSEXC_OBJECTINFO_DUMP	F_BinaryBytes	The value 0x32.
<userobjects>	F_AsciiText	Text which represents multiple userobjects, each separated by a carriage return.

Sample

If we had sent a Harmonizer the following message

```
SYSEXC_PARAMETERS_WANT 0
```

It might respond with the following SysEx message:

Bytes in Corresponding SysEx Message

```
F0 1C 70 01 32 43 4F 4C 20 ... F7
```

Interpretation

```
F0    SysEx start byte
1C    Eventide, Inc.
70    H4000
01    ID of Harmonizer
32    SYSEXC_OBJECTINFO_DUMP
43    C
4F    O
4C    L
20    a space
...   (a whole bunch of text, which was omitted.)
F7    SysEx end byte
```

We can see that a whole bunch of text starts with the 6th byte (0x43). This text might correspond to something like the following. (See the chapter *UserObjects* in order to understand how to decode this information.)

```
COL 0 0 0 'ORVILLE ROOT OBJECT' ORVILLE 7
COL 0 401000b 0 Oscillator-440 '' 0
COL 0 801000b 0 '16mm Projector' '' 0
COL 0 10010000 0 'setup functions' setup 0
8 0 10040000 0 '' ''
COL 0 10020000 0 'program functions' program 0
COL 0 10030000 0 'level functions' level 0
COL 0 10030500 0 'bypass functions' bypass 0
```

SYSEXC_OBJECTINFO_WANT

Message Schematic

```
SYSEXC_OBJECTINFO_WANT <key>
```

```
SYSEXC_OBJECTINFO_WANT <key> <flags>
```

Responses

```
SYSEXC_OBJECTINFO_DUMP    Normal response.
```

```
SYSEXC_ERROR              If Harmonizer has a problem with the message.
```

Description

This message is sent to a Harmonizer in order to retrieve a bunch of userobjects associated with a particular <key>. This message is very similar to SYSEXC_PARAMETERS_WANT, except that if the requested userobject is a

collection (COL), then only the members of the collection will be sent; specifically, members of nested collections will not be sent.

Details

Name	Format	Description
SYSEXC_OBJECTINFO_WANT	F_BinaryBytes	The value 0x31.
<key>	F_AsciiHex	The key of the userobject whose parameters are being requested.
<flags>	F_AsciiHex	A bit-mapped value which optionally reduces the amount of information that is sent. See the table below for the values it may have. Omitting this field is the same as sending 0.

Value of <flags>	Notes
Not present	All data is dumped.
0	All data is dumped.
1	Do not return the members of a collection (COL). This has the side effect of returning zero for <number-of-subobjects> in the SYSEXC_OBJECTINFO_DUMP response message.
2	Do not return the list of strings for a set (SET). This has the side effect of returning zero for <number-of-strings> in the SYSEXC_OBJECTINFO_DUMP response message.
3	Combines the effects of 1 and 2.

Sample

Let's send a message that requests the information about userobjects for the root collection "0".

Schematic

```
SYSEXC_OBJECTINFO_WANT 0
```

Bytes in Corresponding SysEx Message

```
F0 1C 70 01 31 30 F7
```

Interpretation

```
F0 SysEx start byte
1C Eventide, Inc.
70 H4000
01 ID of Harmonizer
31 SYSEXC_OBJECTINFO_WANT
30 0 <key>
F7 SysEx end byte. (Note that <flags> was omitted.)
```

Further Examples

Here we omit all of the encoding and decoding, and concentrate on illustrating the difference between SYSEXC_OBJECTINFO_WANT and SYSEXC_PARAMETERS_WANT.

Send

Let's request the root collection.

```
SYSEXC_OBJECTINFO_WANT 0
```

Response

Notice that the response indicates that all of the non-root collections contain no items! This is because we sent a SYSEXC_OBJECTINFO_WANT.

SysExc_OK

```
COL 0 0 0 'ORVILLE ROOT OBJECT' ORVILLE 7
COL 0 401000b 0 Oscillator-440 '' 0
COL 0 801000b 0 '16mm Projector' '' 0
COL 0 10010000 0 'setup functions' setup 0
8 0 10040000 0 '' ''
COL 0 10020000 0 'program functions' program 0
COL 0 10030000 0 'level functions' level 0
COL 0 10030500 0 'bypass functions' bypass 0
```

Send

If we send the same message, but this time indicate via *<flags>* that we don't want the members of any collections...

```
SYSEXC_OBJECTINFO_WANT 0 1
```

Response

Then we only get the userobject that corresponds to the root collection. Notice that this response indicates that the collection is empty, even though we know that it is not. This is because we set *<flags>* to 1.

```
COL 0 0 0 'ORVILLE ROOT OBJECT' ORVILLE 0
```

Send

Let's request some parameters of a collection, but omit the string list from SET userobjects:

```
SYSEXC_OBJECTINFO_WANT 40a0001 2
```

Response

Notice that the SET claims that *<count-of-strings>* is zero, even though it really isn't.

```
COL 0 40a0001 40a0001 'oscillator parms' osc 5
NUM 0 40b0001 40a0001 'freq: %3.1f Hz' freq 440.00001 0 20000 0.099991
SET 2 40d0001 40a0001 'waveform: %s' waveform 0 sine 0
NUM 0 40c0001 40a0001 'duty cycle: %3.1f %%' duty 50 0 100 0.099991
NUM 0 40f0001 40a0001 'level: %3.1f Db' level -20 -96.000004 20 0.099991
NUM 0 4100001 40a0001 'offset: %3.4f %%' offset 0 -1 1 0.000092
```

Send

Let's now request the full set of strings for the SET above:

```
SYSEXC_PARAMETERS_WANT 40d0001 1
```

Response

Now we get exactly the one userobject, but this time with all of the strings.

```
SET 2 40d0001 40d0001 'waveform: %s' waveform 0 sine 3 sine triangle square
```

SysExc_OK

Message Schematic

```
SYSEXC_OK
```

Description

This message is received from a Harmonizer in response to sending the Harmonizer assorted commands. It simply acknowledges that everything was OK with the last command.

Details

Name	Format	Description
SYSEXC_OK	F_BinaryBytes	The value 0x00.

Sample

If we had sent a Harmonizer a SYSEXC_SIGFILE_DUMP_REMOTE command, it would respond with the following SysEx message if everything went OK.

Bytes in Corresponding SysEx Message

```
F0 1C 70 01 00 F7
```

Interpretation

```
F0    SysEx start byte
1C    Eventide, Inc.
70    H4000
01    ID of Harmonizer
00    SYSEXC_OK
F7    SysEx end byte
```

SYSEXC_PARAMETERS_DUMP

Message Schematic

```
SYSEXC_PARAMETERS_DUMP <userobjects>
```

Description

This message is received from a Harmonizer as the result of sending the Harmonizer a SYSEXC_PARAMETERS_WANT message.

Details

Name	Format	Description
SYSEXC_PARAMETERS_DUMP	F_BinaryBytes	The value 0x2C.
<userobjects>	F_AsciiText	Text which represents multiple userobjects, each separated by a carriage return.

Sample

If we had sent a Harmonizer the following message

```
SYSEXC_PARAMETERS_WANT 401000b 0
```

It might respond with the following SysEx message:

Bytes in Corresponding SysEx Message

```
F0 1C 70 01 2C 43 4F 4C 20 ... F7
```

Interpretation

```
F0    SysEx start byte
1C    Eventide, Inc.
70    H4000
01    ID of Harmonizer
2C    SYSEXC_VALUE_DUMP
43    C
4F    O
4C    L
20           a space
...    (a whole bunch of text, which was omitted.)
F7    SysEx end byte
```

We can see that a whole bunch of text starts with the 6th byte (0x43). This text might correspond to something like the following. (See the chapter *UserObjects* in order to understand how to decode this information.)

```
COL 0 801000b 0 'oscillator (440)' '' 2
COL 0 8030001 801000b 'oscillator parms' osc 8
NUM 0 80d0001 8030001 'level: %5.1f db' '' -20 -96 20 0.1
NUM 0 8080001 8030001 'freq : %5.0f hz' freq 440 0 20000 1
NUM 0 8090001 8030001 'fmod : %5.0f hz' '' 0 0 20000 1
SET 2 80a0001 8030001 'shape: %s' '' 0 sine 3 sine triangle square
NUM 0 80b0001 8030001 'duty : %4.0f %%' '' 50 0 100 1
NUM 0 8040001 8030001 'fm rate : %6.1f hz' '' 1 0 1000 0.1
SET 2 8050001 8030001 'fm shape: %s' '' 0 sine 4 sine triangle square peak
NUM 0 8060001 8030001 'fm duty : %4.0f %%' '' 50 0 100 1
COL e 8020004 801000b info info 8
INF 0 8020002 8020004 %s text 'General-purpose oscillator. On loading'
INF 0 8020066 8020004 %s text 'it is set to a 440 Hz sine wave for'
INF 0 80200ca 8020004 %s text 'tuning. Lfo (fm) allows addition of an'
INF 0 802012e 8020004 %s text 'offset and modulation. Output will'
INF 0 8020192 8020004 %s text 'clipabove +12dB. Aliasing will be'
INF 0 80201f6 8020004 %s text 'audible on triangular and square waves'
INF 0 802025a 8020004 %s text 'at higher frequencies.'
INF 0 80202be 8020004 %s text 'Mono out.'
```

SYSEXC_PARAMETERS_WANT

Message Schematic

```
SYSEXC_PARAMETERS_WANT <key>
SYSEXC_PARAMETERS_WANT <key> <flags>
```

Responses

```
SYSEXC_PARAMETERS_DUMP Normal response.
SYSEXC_ERROR If Harmonizer has a problem with the message.
```

Description

This message is sent to a Harmonizer in order to retrieve a bunch of userobjects associated with a particular <key>.

Details

Name	Format	Description
SYSEXC_PARAMETERS_WANT	F_BinaryBytes	The value 0x2B.
<key>	F_AsciiHex	The key of the userobject whose parameters are being requested.
<flags>	F_AsciiHex	A bit-mapped value which optionally reduces the amount of information that is sent. See the table below for the values it may have. Omitting this field is the same as sending 0.

Value of <flags>	Notes
Not present	All data is dumped.
0	All data is dumped.
1	Do not return the members of a collection (COL). This has the side effect of returning zero for <number-of-subobjects> in the SYSEXC_PARAMETERS_DUMP response message.
2	Do not return the list of strings for a set (SET). This has the side effect of returning zero for <number-of-strings> in the SYSEXC_PARAMETERS_DUMP response message.
3	Combines the effects of 1 and 2.

Sample

Let's send a message that requests the parameters for the preset in DSP A.

Schematic

```
SYSEXC_PARAMETERS_WANT 401000b 0
```

Bytes in Corresponding SysEx Message

```
F0 1C 70 01 2B 34 30 31 30 30 30 62 20 30 F7
```

Interpretation

F0		SysEx start byte
1C		Eventide, Inc.
70		H4000
01		ID of Harmonizer
2B		SYSEXC_PARAMETERS_WANT
34	4	<key>= 0x401000b
30	0	
31	1	
30	0	
30	0	
30	0	
30	0	
62	b	
20		a space
33	0	<flags>
F7		SysEx end byte

SYSEXC_PROGRAM_WANT

Message Schematic

```
SYSEXC_PROGRAM_WANT
```

Description

This message is sent to a Harmonizer to cause it to respond with SYSEXC_PROGRAM_DUMP, which will dump the currently loaded program.

Details

Name	Format	Description
SYSEXC_PROGRAM_WANT	F_BinaryBytes	The value 0x06.

Sample

Schematic

```
SYSEXC_PROGRAM_WANT
```

Interpretation

F0		SysEx start byte
1C		Eventide, Inc.
70		H4000
01		ID of Harmonizer

06 SYSEXC_PROGRAM_WANT
 F7 SysEx end byte

Bytes in Corresponding SysEx Message

F0 1C 70 01 06 F7

SYSEXC_SCREEN_DUMP

Message Schematic

SYSEXC_SCREEN_DUMP <width-in-pixels> <height-in-pixels> <size> <bitmap> <checksum>

Description

This message is received from a Harmonizer when requested by SYSEXC_SCREEN_WANT, and contains a bitmap snapshot of the screen of the Harmonizer.

Details

Name	Format	Description
SYSEXC_SCREEN_DUMP	F_BinaryBytes	The value 0x17.
<width-in-pixels>	F_BinaryNibbles	A 32-bit integer (sent as nibbles) which indicates the width of the bitmap in pixels.
<height-in-pixels>	F_BinaryNibbles	A 32-bit integer (sent as nibbles) which indicates the height of the bitmap in pixels.
<size>	F_BinaryNibbles	A 32-bit integer (sent as nibbles) which indicates the size of <bitmap> in bytes.
<bitmap>	F_BinaryNibbles	The actual bits that comprise the bitmap. The bitmap is a simple 1-bit per pixel representation. It has a variable size given by the product of <width-in-pixel> (rounded up to the nearest 8) and <height-in-pixels>. See below for more details on how the bitmap is laid out.
<checksum>	F_BinaryNibbles	An 8-bit integer (sent as two nibbles) which is a 1-byte checksum. This byte added to all of the bytes (after the bytes have been reconstituted from nibbles) should equal zero.

Sample

Bytes in Received SysEx Message

F0 1C 70 01 17 00 00 00 00 00 0F 00 ... 0A 01 F7

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 17 SYSEXC_SCREEN_DUMP
 00 <width-in-pixels> = 0x000000F0 = 240 pixels wide
 00
 00
 00
 00
 00
 00
 0F

SysExc_Screen_Want

```
00
00   <height-in-pixels> = 0x00000040 = 64 pixels high
00
00
00
00
00
04
00
00   <size> = 0x00000780 = 1920 bytes
00
00
00
00
07
08
00
0V   High nibble of 1st byte of <bitmap> (first byte = 0xVW)
0W   Low nibble of 1st byte of <bitmap>
...   Rest of nibbles that make up <bitmap>
0Y   High nibble of last byte of <bitmap> (last byte = 0xYZ)
0Z   Low nibble of last byte of <bitmap>
0A   High nibble of <checksum> = 0xA1
01   Low nibble of <checksum>
F7   SysEx end byte
```

Note that the bytes comprising $\langle width-in-pixels \rangle + \langle height-in-pixels \rangle + \langle size \rangle + \langle checksum \rangle$ should equal zero. Thus, in our example, $0x00 + 0x00 + 0x00 + 0xF0 + 0x00 + 0x00 + 0x00 + 0x40 + 0x00 + 0x00 + 0x07 + 0x80 + 0xVW + \dots + 0xYZ + 0xA1$ should equal zero.

Pixels are drawn on the screen left-to-right, top-to-bottom. For any particular byte, the highest bit is the left-most pixel, and the lowest bit is the right-most pixel (that is, assuming the byte has already been reconstituted from its nibbles). The next byte contains the next eight pixels immediately to the right of the first byte, continuing until $\langle width-in-pixels \rangle$ (rounded up to the nearest 8, if necessary) have been drawn, in which case one starts the next line below.

SYSExc_SCREEN_WANT

Message Schematic

SYSEXC_SCREEN_WANT

Description

This message is sent to a Harmonizer to cause it to respond with SYSEXC_SCREEN_DUMP, which will dump a bitmap of the Harmonizer's display. Note that this functionality is intended to create screen snap-shots for use in documents (such as this one), and not intended for live control of the Harmonizer. If used for live control, please note that there are states you get the Harmonizer in to where the screen display gets "out-of-synch" with what the Harmonizer thinks the screen looks like.

Details

Name	Format	Description
SYSEXC_SCREEN_WANT	F_BinaryBytes	The value 0x18.

Sample

Schematic

SYSEXC_SCREEN_WANT

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 18 SYSEXC_SCREEN_WANT
 F7 SysEx end byte

Bytes in Corresponding SysEx Message

F0 1C 70 01 18 F7

SYSEXC_SETUP_DUMP

Message Schematic

SYSEXC_SETUP_DUMP <size-of-block> <block> <checksum>

Description

This message is received from a Harmonizer when requested by SYSEXC_SETUP_WANT, and contains a <block> which is data that represents the state of the Harmonizer.

This message can also be sent to a Harmonizer, causing to revert its state to that represented in this message.

Details

Name	Format	Description
SYSEXC_SETUP_DUMP	F_BinaryBytes	The value 0x16.
<size-of-block>	F_BinaryNibbles	A 32-bit integer (sent as nibbles) which indicates the size of <block>.
<block>	F_BinaryNibbles	A sequence of 8-bit integers (sent as nibbles) which are the files on the Harmonizer.
<checksum>	F_BinaryNibbles	An 8-bit integer (sent as two nibbles) which is a 1-byte checksum. This byte added to all of the bytes (after the bytes have been reconstituted from nibbles), including the size bytes, should equal zero.

Sample

Bytes in Received SysEx Message

F0 1C 70 01 16 00 00 00 00 00 01 00 00 ... 0A 01 F7

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 16 SYSEXC_SETUP_DUMP
 00 <size-of-block> = 0x00000100

SysExc_SetUp_Want

00
00
00
00
01
00
00
0V High nibble of 1st byte of <block> (first byte = 0xVW)
0W Low nibble of 1st byte of <block>
... Rest of nibbles that make up <block>
0Y High nibble of last byte of <block> (last byte = 0xYZ)
0Z Low nibble of last byte of <block>
0A High nibble of <checksum> = 0xA1
01 Low nibble of <checksum>
F7 SysEx end byte

Note that the bytes comprising <size-of-block> + <block> + <checksum> should equal zero. Thus, in our example, 0x00 + 0x00 + 0x01 + 0x00 + 0xVW + ... + 0xYZ + 0xA1 should equal zero.

SYSEXC_SETUP_WANT

Message Schematic

SYSEXC_SETUP_WANT

Description

This message is sent to a Harmonizer to cause it to respond with SYSEXC_SETUP_DUMP, which will dump the current state of the unit.

Details

Name	Format	Description
SYSEXC_SETUP_WANT	F_BinaryBytes	The value 0x07.

Sample

Schematic

SYSEXC_SETUP_WANT

Interpretation

F0 SysEx start byte
1C Eventide, Inc.
70 H4000
01 ID of Harmonizer
07 SYSEXC_SETUP_WANT
F7 SysEx end byte

Bytes in Corresponding SysEx Message

F0 1C 70 01 07 F7

SYSExc_SIGDBASE_DUMP

Message Schematic

SYSExc_SIGDBASE_DUMP <sigDBase>

Description

This message is received from a Harmonizer when requested by SYSExc_SIGDBASE_WANT, and contains a SigDBase representation of the programming capabilities of the Harmonizer, which is required by visual editors, such as VSigFile. This dump is very big (currently just over 160K), and so takes a long time. Sending this message to a Harmonizer does nothing, as the SigDBase is determined by the OS version being run. The format of the SigDBase file is beyond the scope of this document.

Details

Name	Format	Description
SYSExc_SIGDBASE_DUMP	F_BinaryBytes	The value 0x0C.
<sigDBase>	F_AsciiText	A text SigDBase, which represents the programming capabilities of the Harmonizer.

Sample

Bytes in Received SysEx Message

```
F0 1C 70 01 0C ... F7
```

Interpretation

```
F0    SysEx start byte
1C    Eventide, Inc.
70    H4000
01    ID of Harmonizer
0C    SYSExc_SIGDBASE_DUMP
...   A whole bunch of ASCII characters, which comprise a SigDBase file.
F7    SysEx end byte
```

SYSExc_SIGDBASE_WANT

Message Schematic

SYSExc_SIGDBASE_WANT

Description

This message is sent to a Harmonizer to cause it to respond with SYSExc_SIGDBASE_DUMP, which will dump the Harmonizer's SigDBase, which is a file required by visual editors, such as VSigFile, in order to correctly interpret SigFiles. It takes quite some time for the Harmonizer to transmit the SYSExc_SIGDBASE_DUMP.

Details

Name	Format	Description
SYSExc_SIGDBASE_WANT	F_BinaryBytes	The value 0x0E.

Sample

Schematic

SYSEXC_SIGDBASE_WANT

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 0E SYSEXC_SIGDBASE_WANT
 F7 SysEx end byte

Bytes in Corresponding SysEx Message

F0 1C 70 01 0E F7

SYSEXC_SIGFILE_DUMP

Message Schematic

SYSEXC_SIGFILE_DUMP <sigfile>

Description

This message is received from a Harmonizer when requested by SYSEXC_SIGFILE_WANT, and contains a SigFile representation of the currently loaded program. This version of the SigFile is heavily commented as to data type, name, min, max, etc. Details about SigFiles are beyond the scope of this document.

This message can also be sent to Harmonizer, and will cause the Harmonizer to encode the program, compile it, and load it. This does take some time. If there are any errors, they will be reported on the screen; no response message is sent.

Details

Name	Format	Description
SYSEXC_SIGFILE_DUMP	F_BinaryBytes	The value 0x08.
<sigfile>	F_AsciiText	A text SigFile, which represents the currently loaded program.

Sample

Bytes in Received SysEx Message

F0 1C 70 01 08 ... F7

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 08 SYSEXC_SIGFILE_DUMP
 ... A whole bunch of ASCII characters, which comprise a SigFile.
 F7 SysEx end byte

SYSExc_SIGFILE_DUMP_REMOTE

Message Schematic

SYSExc_SIGFILE_DUMP_REMOTE <sigfile>

Description

This message is received from a Harmonizer when requested by SYSExc_SIGFILE_WANT_QUICK, and contains a SigFile representation of the currently loaded program. This version of the SigFile is not commented, so as to transmit more quickly. Details about SigFiles are beyond the scope of this document.

This message can also be sent to Harmonizer, and will cause the Harmonizer to encode the program, compile it, and load it. This does take some time. If there are any errors, they will be reported by a SYSExc_ERROR response; no message is displayed on the Harmonizer screen.

Details

Name	Format	Description
SYSExc_SIGFILE_DUMP_REMOTE	F_BinaryBytes	The value 0x0A.
<sigfile>	F_AsciiText	A text SigFile, which represents the currently loaded program.

Sample

Bytes in Received SysEx Message

```
F0 1C 70 01 0A ... F7
```

Interpretation

```
F0    SysEx start byte
1C    Eventide, Inc.
70    H4000
01    ID of Harmonizer
0A    SYSExc_SIGFILE_DUMP_REMOTE
...   A whole bunch of ASCII characters, which comprise a SigFile.
F7    SysEx end byte
```

SYSExc_SIGFILE_WANT

Message Schematic

SYSExc_SIGFILE_WANT

Description

This message is sent to a Harmonizer to cause it to respond with SYSExc_SIGFILE_DUMP, which will dump the currently loaded program as a SigFile. This SigFile is heavily commented as to data type, name, min, max, etc. It does take quite some time for the Harmonizer to transmit a SYSExc_SIGFILE_DUMP.

Details

Name	Format	Description
SYSExc_SIGFILE_WANT	F_BinaryBytes	The value 0x09.

Sample

Schematic

SYSEXC_SIGFILE_WANT

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 09 SYSEXC_SIGFILE_WANT
 F7 SysEx end byte

Bytes in Corresponding SysEx Message

F0 1C 70 01 09 F7

SYSEXC_SIGFILE_WANT_QUICK

Message Schematic

SYSEXC_SIGFILE_WANT_QUICK

Description

This message is sent to a Harmonizer to cause it to respond with SYSEXC_SIGFILE_DUMP_REMOTE, which will dump the currently loaded program as a SigFile. This SigFile has no comments, making this command much faster than SYSEXC_SIGFILE_WANT. It still does, however, take some time for the Harmonizer to transmit the SYSEXC_SIGFILE_DUMP_REMOTE.

Details

Name	Format	Description
SYSEXC_SIGFILE_WANT_QUICK	F_BinaryBytes	The value 0x0B.

Sample

Schematic

SYSEXC_SIGFILE_WANT_QUICK

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 0B SYSEXC_SIGFILE_WANT_QUICK
 F7 SysEx end byte

Bytes in Corresponding SysEx Message

F0 1C 70 01 0B F7

SYSEXC_VALUE_DUMP

Message Schematic

SYSEXC_VALUE_DUMP <key> (COL, TRG)
 SYSEXC_VALUE_DUMP <key> [<value>] (CON, INF, NUM, STR)
 SYSEXC_VALUE_DUMP <key> [<value> <string>] (SET)

Description

This message is received from a Harmonizer as the result of sending the Harmonizer a SYSEXC_VALUE_PUT message.

Details

Name	Format	Description
SYSEXC_VALUE_DUMP	F_BinaryBytes	The value 0x2E.
<key>	F_AsciiHex	The key of the userobject whose value is to be altered.
<value>	See table below.	A value represented in ASCII, where the exact representation depends upon the <type> of the userobject represented by the <key>.
<string>	F_AsciiString	This is only present for SET userobjects, and is the actual string represented by the value <value>.

<type> of UserObject	Format of <value>	Notes
COL	N/A	<value> is not present.
CON	F_AsciiFloat	
INF	F_AsciiString	
NUM	F_AsciiFloat	
SET	F_AsciiHex*	For details, see the description of SET in “UserObjects”.
STR	F_AsciiString	
TRG	N/A	<value> is not present.

Sample

Setting the <key> 0x1000, which it is of the NUM type, to the value 3.4.

Bytes in Received SysEx Message

F0 1C 70 01 2E 31 30 30 30 20 33 2E 34 F7

Interpretation

F0	SysEx start byte
1C	Eventide, Inc.
70	H4000
01	ID of Harmonizer
2E	SYSEXC_VALUE_DUMP
31	1 <key> 0x1000 in ASCII
30	0
30	0
30	0
20	a space
33	3 floating point <value> in ASCII = “3.4”
2E	.

* Note that this is asymmetrical with respect to SYSEXC_VALUE_PUT, which requires the SET value to be in the format F_AsciiDecimal.

34 4
F7 SysEx end byte

Schematic

SYSEXC_VALUE_DUMP 1000 3.4 (where <key> 0x1000 is a NUM)

SYSEXC_VALUE_PUT

Message Schematic

SYSEXC_VALUE_PUT <key> Used to generate a SYSEXC_VALUE_DUMP message.
SYSEXC_VALUE_PUT <key> <value> Used to set a new value into <key>, or to trigger a TRG.

Responses

SYSEXC_VALUE_DUMP Normal response.
SYSEXC_ERROR If Harmonizer has a problem with the message.
None Normal response if <key> is of the TRG type.

Description

This message is sent to a Harmonizer in order to:

1. Request a SYSEXC_VALUE_DUMP for a particular <key>, that is, fetch <key>'s current value.
2. Set a new value for a particular <key>, which also generates a SYSEXC_VALUE_DUMP message in response.
3. To cause a trigger to occur.

Details

Name	Format	Description
SYSEXC_VALUE_PUT	F_BinaryBytes	The value 0x2D.
<key>	F_AsciiHex	The key of the userobject whose value is to be altered.
<value>	See table below.	A value represented in ASCII, where the exact representation depends upon the <type> of the userobject represented by the <key>. If omitted, then the current value is not changed and the SYSEXC_VALUE_DUMP message is still returned, providing one with the current value of the <key>.

<type> of UserObject	Format of <value>	Notes
NUM	F_AsciiFloat	
SET	F_AsciiDecimal*	
STR	F_AsciiString	
TRG	F_AsciiFloat	An arbitrary value, such as 1. The actual value supplied is ignored. However, some value must be present in order to cause the trigger.
COL, CON, INF	N/A	Not applicable, as these userobjects are read-only.

Sample

Setting the <key> 0x1000, which is of the NUM type, to the value 3.4.

Schematic

SYSEXC_VALUE_PUT 1000 3.4 (where <key> 0x1000 is a NUM)

* Note that this is asymmetrical with respect to SYSEX_VALUE_DUMP, in which the SET value is in the format F_AsciiHex.

Bytes in Corresponding SysEx Message

F0 1C 70 01 2D 31 30 30 30 20 33 2E 34 F7

Interpretation

F0 SysEx start byte
 1C Eventide, Inc.
 70 H4000
 01 ID of Harmonizer
 2D SYSEXC_VALUE_PUT
 31 1 <key> 0x1000 in ASCII
 30 0
 30 0
 30 0
 20 a space
 33 3 floating point <value> in ASCII = "3.4"
 2E .
 34 4
 F7 SysEx end byte

SYSEXC_USEROBJECT**Message Schematic**

SYSEXC_USEROBJECT <lots-of-bytes>

Description

This message is received from a Harmonizer as the result of parameter changes, if the Harmonizer is set up to do so. These messages can then be recorded by a MIDI sequencer, and when replayed, will cause the control changes to be duplicated. Every userobject message gets a response.

Details

Name	Format	Description
SYSEXC_USEROBJECT	F_BinaryBytes	The value 0x02.
<lots-of-bytes>	F_BinaryBytes	The data associated with a parameter changing.

Appendix A: Summary of Formats

This appendix acts as a summary of ways numbers and messages are formatted into the bytes that flow between a computer and a Harmonizer. For convenience, each format is given a name, and that name is referred to from the rest of this document.

F_AsciiDecimal

This format only applies to unsigned integer values within a larger sequence of ASCII characters. It is *only* used in the SYSEXC_VALUE_PUT message to send the <value> of a SET userobject—in all other integer cases F_AsciiHex is used instead. This format simply represents an integer value as a sequence of decimal digits in ASCII. The number is usually delimited from other data with a space character. For example, the decimal number 314 would be represented as the following sequence of bytes:

```
0x33  ASCII for the character '3'
0x31  ASCII for the character '1'
0x34  ASCII for the character '4'
```

F_AsciiFloat

This format is used to represent floating point values within a larger sequence of ASCII characters. This format simply represents a value as a sequence of ASCII digits (in decimal), as well as the minus sign and decimal point, if needed. The number is usually delimited from other data with a space character. For example, the number -3.14 would be represented as the following sequence of bytes:

```
0x2D  ASCII for the character '-'
0x33  ASCII for the character '3'
0x2E  ASCII for the character '.'
0x31  ASCII for the character '1'
0x34  ASCII for the character '4'
```

F_AsciiHex

This format applies to unsigned integer values within a larger sequence of ASCII characters. This format represents an integer value as a sequence of hex digits in ASCII. The hex number is usually delimited from other data with a space character. For example, the number 314 (decimal) would be represented as the following sequence of bytes (since 314 decimal is 13A in hexadecimal):

```
0x31  ASCII for the character '1'
0x33  ASCII for the character '3'
0x61  ASCII for the character 'a'
```

F_AsciiSimpleString

This format only applies to an arbitrary sequence of characters, contained within a larger sequence of ASCII characters, that will never contain a space character, a single-quote character, or a double-quote character. This format represents the string as a sequence of ASCII characters. Such strings are often delimited from other data by spaces. For example, the string "SET" would be represented as the following sequence of bytes:

```
0x53  ASCII for the character 'S'
0x45  ASCII for the character 'E'
0x54  ASCII for the character 'T'
```

F_AsciiString

This format applies to a nearly arbitrary sequence of characters contained within a larger sequence of ASCII characters. This format represents the string as a sequence of ASCII characters. If the original string contains any space characters, then this format encloses the string in single quotes. If the original string contains any single-quote characters, then this format encloses the string in double-quotes.

Appendix A: Summary of Formats

For example, the string “SET” would be represented as the following sequence of bytes:

```
0x53  ASCII for the character ‘S’
0x45  ASCII for the character ‘E’
0x54  ASCII for the character ‘T’
```

For example, the string “A cat.” would be represented as the following sequence of bytes:

```
0x27  ASCII for the character ‘”’ (single quote)
0x41  ASCII for the character ‘A’
0x20  ASCII for the character ‘ ’ (space)
0x63  ASCII for the character ‘c’
0x61  ASCII for the character ‘a’
0x74  ASCII for the character ‘t’
0x2E  ASCII for the character ‘.’ (period)
0x27  ASCII for the character ‘”’ (single quote)
```

For example, the string “Jo’s E” would be represented as the following sequence of bytes:

```
0x27  ASCII for the character ‘”’ (double quote)
0x4A  ASCII for the character ‘J’
0x6F  ASCII for the character ‘o’
0x27  ASCII for the character ‘”’ (single quote)
0x73  ASCII for the character ‘s’
0x20  ASCII for the character ‘ ’ (space)
0x45  ASCII for the character ‘E’
0x27  ASCII for the character ‘”’ (single quote)
```

F_AsciiText

This format is used when the entire *<message-data>* of a message is a sequence of ASCII characters. In this case, an ASCII string can be created from the data between the *<message-code>* and the MIDI SysEx end-byte (0xF7).

F_BinaryBytes

If one or more bytes of data already adheres to the MIDI SysEx Specification (which requires that no byte ever has its high-bit set; i.e. no byte is ever greater than 0x7F or 127 decimal), then they don’t really need any formatting or encoding. Such bytes can simply be placed right in the input or output stream. For example, the byte that represents a SYSEX_ message is guaranteed to never exceed 0x7F, so such that byte never needs encoding.

F_BinaryNibbles

If one or more bytes of data do not adhere to the MIDI SysEx Specification, then the bytes must be encoded in a manner that guarantees that the encoded message never contains bytes that have a high-bit set.* This becomes a problem when transmitting binary data, such as a bitmap, or when transferring 8-, 16-, or 32-bit integers in as their constituent bytes. The solution is to split every byte of the un-encoded message into two bytes in the coded message. The details for 8-, 16-, and 32-bit integers are presented below.

This format encodes a 32-bit integer into eight bytes as follows: the highest four bits of the integer become the low four bits of the first byte; the second highest four bits of the integer become the low four bits of the second byte, the third highest four bits of the integer become the low four bits of the third byte, etc. For example, the integer 0xABCDEF12 becomes the following eight bytes: 0x0A, 0x0B, 0x0C, 0x0D, 0x0E, 0x0F, 0x01, 0x02.

Similarly, a 16-bit integer can be encoded into four bytes. For example, the integer 0x34F1 becomes the following four bytes: 0x03, 0x04, 0x0F, 0x01.

And of course, an 8-bit integer still needs to be encoded since it may have its high-bit set; so an 8-bit integer is encoded into two bytes. For example, the integer 0xA5 becomes the following two bytes: 0x0A, 0x05.

* With the obvious exception of the special byte that terminates all MIDI SysEx messages, 0x7F, which is always the very last byte of a SysEx message, and thus not part of the *contents* of the SysEx message.

Appendix B: List of SysExc_ Messages

This appendix presents the list of SYSEXC_ messages sorted by message number. For an alphabetical list or for complete descriptions of these messages, see *Part II* of this document.

Messages marked with “command” are commands that may be sent to a Harmonizer. Messages marked with “response” are messages received from the Harmonizer, usually in response to a command having been sent. Messages marked with “both”, can be used as a command, but may also be received from a Harmonizer.

<message-code> (hex)	(decimal)	Command / Response	Message Name	
0x00	0	response	SYSEXC_OK	
0x01	1	command	SYSEXC_KEYPRESS	
0x02	2	response	SYSEXC_USEROBJECT	
0x03	3	response	SYSEXC_BANKCHANGE	
0x04	4	both	SYSEXC_PROGRAM_DUMP_OLD	<i>(Obsolete—do not use.)</i>
0x05	5	both	SYSEXC_SETUP_DUMP_OLD	<i>(Obsolete—do not use.)</i>
0x06	6	command	SYSEXC_PROGRAM_WANT	
0x07	7	command	SYSEXC_SETUP_WANT	
0x08	8	both	SYSEXC_SIGFILE_DUMP	
0x09	9	command	SYSEXC_SIGFILE_WANT	
0x0A	10	both	SYSEXC_SIGFILE_DUMP_REMOTE	
0x0B	11	command	SYSEXC_SIGFILE_WANT_QUICK	
0x0C	12	response	SYSEXC_SIGDBASE_DUMP	
0x0D	13	response	SYSEXC_ERROR	
0x0E	14	command	SYSEXC_SIGDBASE_WANT	
0x0F	15	both	SYSEXC_FILES_DUMP	
0x10	16	command	SYSEXC_FILES_WANT	
0x11	17	both	SYSEXC_INTERNAL_DUMP	
0x12	18	command	SYSEXC_INTERNAL_WANT	
0x13	19	both	SYSEXC_CARD_DUMP	
0x14	20	command	SYSEXC_CARD_WANT	
0x15	21	both	SYSEXC_PROGRAM_DUMP	
0x16	22	both	SYSEXC_SETUP_DUMP	
0x17	23	response	SYSEXC_SCREEN_DUMP	
0x18	24	command	SYSEXC_SCREEN_WANT	
0x19	25	response	SYSEXC_INFO_DUMP	
0x1A	26	command	SYSEXC_INFO_WANT	
0x2B	43	command	SYSEXC_PARAMETERS_WANT	
0x2C	44	response	SYSEXC_PARAMETERS_DUMP	
0x2D	45	command	SYSEXC_VALUE_PUT	
0x2E	46	response	SYSEXC_VALUE_DUMP	
0x31	49	command	SYSEXC_OBJECTINFO_WANT	
0x32	50	response	SYSEXC_OBJECTINFO_DUMP	