How to build a remote control unit for the Eventide Orville™

This document describes the use of the MIDI Sysex protocols offered by the Eventide Orville 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 *Orville*, but should be taken to apply to any of the supported systems.

They are currently supported by the following Eventide Products:
Orville V2.705 or later
DSP7000/7500/4000B+ (all)
Eclipse V1.1 or later

(i) The Orville operating system is made up of *userobjects.* These may, for example, display a value, give the function of a push-button, or act as a menu page. Userobjects have the following features:
(ii) Each one is identified by a unique ID number, known as the *key*.
(iii) They usually contain some data, in various formats.
(iv) Each one contains a *statement* which is a "C" type format string, to be used in displaying their data. This may be empty.
(v) Each one contains a *tag,* which can be used as a label for buttons. This may be empty.
(vi) There are a number of different *types* of userobject. See below for detailed information.
(vii) A userobject of a particular type also has a *subtype*. This can usually be ignored.
(viii) Some userobjects are read-only or *constant* (these display information without allowing it to be changed, for example the sample rate of an incoming signal), some userobjects are write-only (these are *triggers*, which act as push-button switches).
(ix) Most userobjects contain a number of *attributes*, which give further information necessary for their use.

Userobject Types

(i) Collection "COL":

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 Orville 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*.

Attributes: the number of subobjects in the collection.

Representation:
COL <subtype> <key> <parent key> <statement> <tag> <number subobjects>

(ii) Number "NUM":

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

Attributes: the *maximum* and *minimum* values, and the *resolution,* which is the amount the value should be increased or decreased for each click of a wheel or button.

Representation:
NUM <subtype> <key> <parent key> <statement> <tag> <current value> <minimum> <maximum> <resolution>.

(iii) String "STR":

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

Attributes: none.

Representation:
STR <subtype> <key> <parent key> <statement> <tag> <current value>

(iv) Constant "CON":

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.

Attributes: none.

Representation:
CON <subtype> <key> <parent key> <statement> <tag> <current value>

(v) Information "INF":

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.

Attributes: none.

Representation:
INF <subtype> <key> <parent key> <statement> <tag> <current value>

(vi) Set "SET":

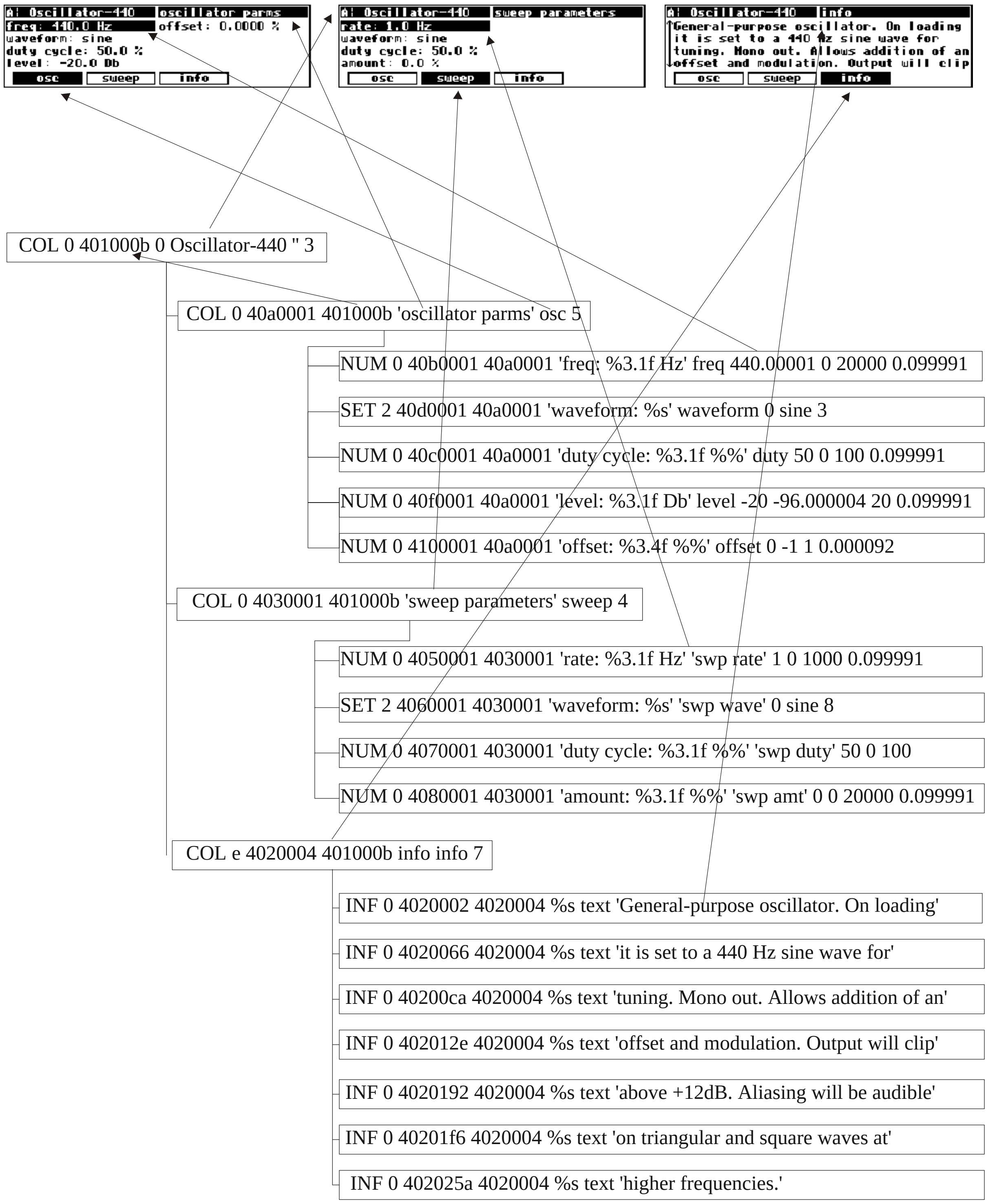
A *set* userobject contains a number of text strings, one of which is selected at any given time. Thus, its data is the index of the selected string, which can be read or written.

Attributes: number of strings, value of each string, value of selected string.

Representation:
SET <subtype> <key> <parent key> <statement> <tag> <current index> <selected string> <number of strings>.<string text>*

Userobject trees

The userobjects in the Orville are arranged as a *tree,* with collections acting as the branching points. Consider, for example, the PARAMETER display of the preset below, consisting of three 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 Orville screens. The entire user interface of the



```
A: Oscillator-440    oscillator parms
freq: 440.0 Hz       offset: 0.0000 %
waveform: sine
duty cycle: 50.0 %
level: -20.0 Db
  osc     sweep    info
```

```
A: Oscillator-440    sweep parameters
rate: 1.0 Hz
waveform: sine
duty cycle: 50.0 %
amount: 0.0 %
  osc     sweep    info
```

```
A: Oscillator-440    info
↑General-purpose oscillator. On loading
 it is set to a 440 Hz sine wave for
 tuning. Mono out. Allows addition of an
↓offset and modulation. Output will clip
  osc     sweep    info
```

```
COL 0 401000b 0 Oscillator-440 '' 3
    COL 0 40a0001 401000b '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 3
        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
    COL 0 4030001 401000b 'sweep parameters' sweep 4
        NUM 0 4050001 4030001 'rate: %3.1f Hz' 'swp rate' 1 0 1000 0.099991
        SET 2 4060001 4030001 'waveform: %s' 'swp wave' 0 sine 8
        NUM 0 4070001 4030001 'duty cycle: %3.1f %%' 'swp duty' 50 0 100
        NUM 0 4080001 4030001 'amount: %3.1f %%' 'swp amt' 0 0 20000 0.099991
    COL e 4020004 401000b info info 7
        INF 0 4020002 4020004 %s text 'General-purpose oscillator. On loading'
        INF 0 4020066 4020004 %s text 'it is set to a 440 Hz sine wave for'
        INF 0 40200ca 4020004 %s text 'tuning. Mono out. Allows addition of an'
        INF 0 402012e 4020004 %s text 'offset and modulation. Output will clip'
        INF 0 4020192 4020004 %s text 'above +12dB. Aliasing will be audible'
        INF 0 40201f6 4020004 %s text 'on triangular and square waves at'
        INF 0 402025a 4020004 %s text 'higher frequencies.'
```

Orville 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.

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

(i) COL 0 401000b 0 Oscillator-440 '' 3
(ii) COL 0 8010007 0 "Clrmtn's NemWhipper" NWhip 6
(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

The key to keys
A key is a 32 bit unique ID value.
Its top 4 bits (D28-31) are zero for a PARAMETER key.
The next 2 bits (D26,27) tell you if a PARAMETER key is for DSP A (01) or DSP B (1.

In Eclipse, the modulator block is DSP C (11)

(i),(ii) are the PARAMETER trees for the A and B machines. Note the double quotes in (ii), while the apparent double quotes in (i) 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 the Orville. For example, a product where the BYPASS button acts directly (DSP7000), rather than via a menu as on Orville, might have a *trigger* rather than a *collection* at that point.

MIDI Sysex message formats

The format of the messages described in this section is as follows:

0xF0 EVENTIDE H4000 <id> <message_code> < message data > 0xF7

the 0xF0 and 0xF7 are standard MIDI for system exclusive and end of system exclusive. All other bytes have d7 set to 0.

```
#define EVENTIDE 0x1C
#define H4000 0x70
```

<id> is the device id number. If this is zero, all Orvilles will listen to the message.

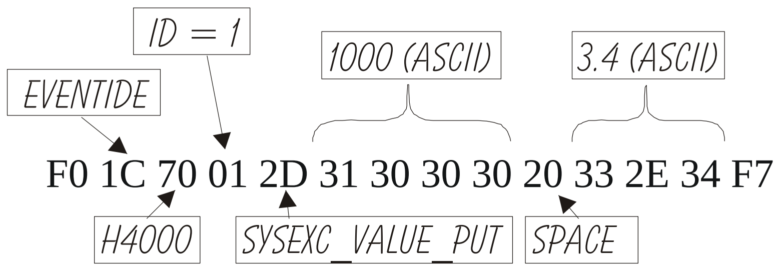
<message_code> is a byte determining the message function. For the purposes of this document its can be one of:

Message	*Code*	*Response*	*Code*
SYSEXC_PARAMETERS_WANT	0x2b	SYSEXC_PARAMETERS_DUMP	0x2c
SYSEXC_VALUE_PUT	0x2d	SYSEXC_VALUE_DUMP	0x2e
SYSEXC_OBJECTINFO_WANT	0x31	SYSEXC_OBJECTINFO_DUMP	0x32

All the above messages have one or more ascii parameters with spaces between them. They are described in detail below. Their parameters may be one of the following:

<key> : <long>	key value (hexadecimal assumed)
<long>	ascii hexadecimal number with leading zeros suppressed.
<float>	ascii floating point value with leading and non-significant trailing zeros and decimal points suppressed. Example: **1.0** is sent as **1**
<string>	ascii text string. If the first character is a single or double quote this is the start and end delimiter, otherwise the delimiter is a space. No string may contain all three of single quote, double quote, space.

Here is an example of the SYSEX message you would send to set the value of an object with a key value of 0x1000 to 3.4:



SYSEXC_PARAMETERS_WANT <key> [<flags>]

<key> is the key value for the object whose parameters are desired.
<flags> is a bitmapped value which optionally reduces the amount of information that is sent. It may have the following values:

<not present> | 0	no effect
1	make the returned value for the number of members for each COLLECTION zero - i.e. do not return the members.
2	make the returned value for the number of strings for each SET zero - i.e. do not return the strings.

The response to this message is a SYSEXC_PARAMETERS_DUMP message which is the entire parameter *tree* for the requested value. This can be large and take several seconds to send. Intelligent use of the *flags* to send only such data as is needed will significantly speed up operation. The representation for the *userobject* data is as described above, with each object terminated by a CR/LF. Example:

SYSEXC_PARAMETERS_WANT 401000b 0   //a parameter tree for the preset in DSP A might return the following data:

```
COL 0 401000b 401000b Oscillator-440 '' 3
 COL 0 40a0001 401000b '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 3 sine triangle square
  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
 COL 0 4030001 401000b 'sweep parameters' sweep 4
  NUM 0 4050001 4030001 'rate: %3.1f Hz' 'swp rate' 1 0 1000 0.099991
  SET 2 4060001 4030001 'waveform: %s' 'swp wave' 0 sine 8 sine triangle square peak 'warp sin'
'warp tri' 'half sin' 'half peak'
  NUM 0 4070001 4030001 'duty cycle: %3.1f %%' 'swp duty' 50 0 100 0.099991
  NUM 0 4080001 4030001 'amount: %3.1f %%' 'swp amt' 0 0 20000 0.099991
 COL e 4020004 401000b info info 7
  INF 0 4020002 4020004 %s text 'General-purpose oscillator. On loading'
  etc etc etc.
```

SYSEXC_OBJECTINFO_WANT <key> [<flags>]

This message receives the key for a single object and returns a SYSEXC_OBJECTINFO_DUMP message. It differs from SYSEXC_PARAMETERS_WANT in that if the key is for a COLLECTION, only the members of the COLLECTION will be sent - members of nested COLLECTION will not be sent. Examples:

SYSEXC_OBJECTINFO_WANT 0

```
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 '' ''                               //unknown type - ignore
 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 0 1

```
COL 0 0 0 'ORVILLE ROOT OBJECT' ORVILLE 0//no members of collection
```

SYSEXC_OBJECTINFO_WANT 40a0001 2

```
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// no strings in set
 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
```

SYSEXC_OBJECTINFO_WANT 40d0001

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

SYSEXC_VALUE_PUT <key> [<value>]

This message receives the key for a single object and returns a SYSEXC_VALUE_DUMP message. If <value> is present it will be written to the userobject, and should be of a suitable type for the object:

SET should be a <long>, being the index for the desired string.
NUM should be a <float>
STR should be a <string>
TRG can be anything (it is ignored - the act of writing causes the trigger to occur).
COL, CON, INF can not be written to (it will have no effect).

The returned SYSEXC_VALUE_DUMP will be of the form:

<key> <float> for a NUM
<key> <long> <string> for a SET // <value> will be the current index
<key> <string> for an INF
<key> <float> for a CON
<key> only for a TRG or a COL.

This message can be used to find the current value of an object without changing it simply by not sending a value.