Serial Interface Kit

Soundweb™

BSS Audio

Limited warranty

No warranties: BSS Audio expressly disclaims any warranty for the ‘Soundweb Interface Kit’. The ‘Soundweb Interface Kit’ and any related documentation is provided ‘as is’ without warranty of any kind, either express or implied, including, without limitation, the implied warranties or merchantability, fitness for a particular purpose, or non infringement. The entire risk arising out of use or performance of the ‘Soundweb Interface Kit’ remains with you.

No Liability for damages: In no event shall BSS Audio or its suppliers be liable for any damages whatsoever (including, without limitation, damages for loss of business profits, business interruption, loss of business information, or any other pecuniary loss) arising out of the use of, misuse of, or inability to use this BSS Audio product, even if BSS Audio has been advised of the possibility of such damages. Because some states/jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, the above limitation may not apply to you.

BSS Audio is a division of Harman International Industries Ltd., 4th floor, Windsor House, Pepper Street, Chester CH1 1DF, United Kingdom.

Introduction

This document is intended for Soundweb users who wish to provide their own user interface for a Soundweb system. The user interface can be based on an PANJA system, a PC running a custom application, or even a custom piece of hardware.

There are two ways in which the developer can control a Soundweb network. There’s the PANJA interface which is a very simple RS232 protocol designed for use with PANJA devices running the AXCESS development system. However, the PANJA RS232 protocol is tailored for PANJA, and is therefore not generic enough to control all of Soundweb’s processing objects. For this reason we created the RAW_MSG extension which gives almost complete control of a Soundweb network via RS232.

Both protocols share the same message protocol and use the rear RS232 port of a Soundweb device.

The Serial Interface Kit FAQ

v1.4 29/11/00

Q1. PANJA/AMX interfacing - where do I start?
Q2. Are there any examples?
Q3. What if I already have an PANJA/AMX system in use?
Q4. How do handles work?
Q5. Where do I get handles from?
Q6. So, what are method codes then?
Q7. Are presets global?
Q8. How do I control a matrix mixer?
Q9. What's the most efficient way to switch between two configurations with the control port?

Q1. PANJA/AMX interfacing - where do I start?

A1. A complete walk-through of setting a system up to interface with an AMX panel is in the Soundweb help. It describes both hardware and software requirements and is for a user who wants to add a panel to his Soundweb network. The idea is that you should not need to write any code if you just want a panel on your system.

Q2. Are there any examples?

A2. Yes. There is a file called Example.SDF available which is a system with various objects already set up.

Q3. What if I already have an AMX system in use?

A3. If, after generating the code from Soundweb Designer, you have channel codes that conflict with ones already on the panel, you can edit SWFUNC.AXI which is where PUSH events (etc.) meet our messaging functions. You must include the library functions which are to be run inside mainline for messages to be sent and received.

Q4. How do handles work?

A4. Each processing object in a design file (.SDF) is allocated a unique number to identify it when the design is compiled. If there is more than one unit in the design, a range of numbers are allocated to each box. This range is big enough for more processing objects than you could possible need or even fit in one unit. It is 100000 in hexadecimal (written 0x100000 or

100000h) which is 1048576 in decimal.

For example a gain element in the first unit may be allocated the number 2. The first handle, 1, is reserved for the box itself for permanent objects associated with the box rather than the design you have created such as phantom power on the inputs, gain trim on the outputs. The second unit in a design will have the handle 0x100001 and processing objects will start from 0x100002. Because of these ranges, the number will always be a dword, or 4 bytes or 32 bit value when used in a messaging protocol.

Q5. Where do I get handles from?

A5. They do not exist until you have compiled your design file (.SDF) in Soundweb Designer. Once compiled, you can find them by moving the mouse over the object in the design and the handle will appear in the status bar which is in the bottom left of the main Soundweb designer window. This is one of the places where an application would normally tell you what buttons do as you pass the mouse pointer over them.

Q6. So, what are method codes then?

A6. Each processing object has a unique handle to identify it but this one reference is not enough to uniquely identify a particular control. For example, our gain element has three controls associated with it. These are the gain itself or the fader, the mute button and the phase button. We give these three controls numbers which we call method codes because they identify a function or method of an object. You may have noticed that these numbers are re-used on different processing objects - this is because the handle makes the control unique.

Think of method codes like doors, windows and chimneys and handles as the house number. Two houses can have the same chimney but will be unique if referred to with the house number as well. Method codes are also dwords or 4 bytes or 32 bit values, however you like to think of them.

Q7. Are Presets global?

A7. There are two type of presets. Parameter presets involve all the controls on the map window in which they are created. A common technique is to have a special map window for the controls your are interested in and make the preset there. This preset bar is then copied to your main control panel.

Major presets have the option to include devices at your discretion to save memory. These presets can affect the entire network. Each box has a copy of the control ids and parameter values for each control in the preset. Activation is a single message rather than all the values sent around. If a box doesn't contain any controls in the preset, it doesn't need a copy of the preset.

Q8. How do I control a matrix mixer?

A8. The gain of each route channel is controlled by usung a method code calculated as follows:
Identify the output you want to route.
Find the base method code for this in this Soundweb interface kit document. They start at 0x01000800.
Add to this the input channel-1.

Routing is similar but the base method codes start at 0x01001000, e.g. the button for Output 5 Input 4 has a method of 0x010010C0+4-1 = 0x010010C3.

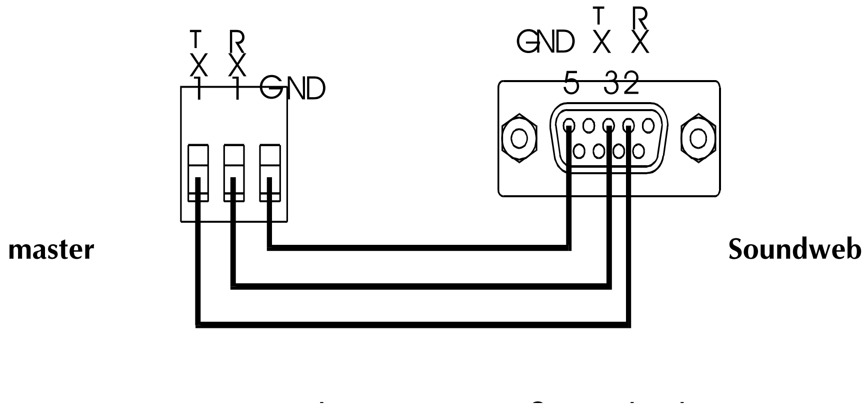
Q9. What's the best way to switch between two configurations with the control port?

A9. Use a couple of Presets. We recommend two momentary buttons. If you change to each configuration and store this in a preset, you can add a preset button for each, then drag these preset buttons onto control inputs.

See the section on Presets in the online help in Soundweb Designer for further information.

Hardware

Soundweb to PANJA



3-wire Null modem cable for PANJA to Soundweb connection.

PC to Soundweb (Rear RS232) Cable

7-wire Null modem cable.

Messaging protocol

- Always use 8-bit data with no parity.
- Bit rate 38400 bps.

The following bytes have special meanings:

- 0x02 **STX**
- 0x03 **ETX**
- 0x06 **ACK**
- 0x15 **NAK**
- 0x1B **Escape**
- Any other single byte can be used within a message body

To use one of the special bytes within a message body, do the following:

- 0x02 - substitute with 0x1B 0x82
- 0x03 - substitute with 0x1B 0x83
- 0x06 - substitute with 0x1B 0x86
- 0x15 - substitute with 0x1B 0x95
- 0x1B - substitute with 0x1B 0x9B

The following bytes are command bytes to appear at the beginning of a message after **STX**.

- 0x80 **SET_VALUE**
- 0x81 **SET_STRING**
- 0x82 **REQUEST_VALUE**
- 0x83 **REQUEST_STRING**
- 0x84 **RAW_MSG**

Message Format

<***message***> = <***STX***> <***Body***> <***Checksum byte***> <***ETX***>
<***Checksum byte***> is the xor of all the bytes in <***Body***>

Note:
If the checksum is one of the special characters it must be substituted in the same way as body bytes.

Message Body Format

This is one of the following:

<*Body*> = <**SET_VALUE**> <***group***> <***id***> <***value Hi***> <***value Lo***>
or <**SET_STRING**> <***group***> <***id***> <***char 0***> <***char 1***> <***char 2***>...<***Zero***>
or <**REQUEST_VALUE**> <***group***> <***id***>
or <**REQUEST_STRING**> <***group***> <***id***>
or <**RAW_MSG**> <***Soundweb Message***>

<***group***>
a byte identifying which type of control you are changing.
<***id***>
a byte uniquely identifying which control of the group you are changing.
<***Value Hi***> <***value Lo***>
a sixteen bit value.
<***char 0***> <***char 1***> <***char 2***>...<***Zero***>
an ASCII null-terminated string in the standard Windows character set.
<***Soundweb Message***>
a raw Soundweb message.

The following group numbers are allocated.
SW_AMX_BUTTON = 0 (* MOMENTARY BUTTONS (255) *)
SW_AMX_TOGGLE = 1 (* LATCHING BUTTONS. (255) *)
SW_AMX_LED = 2 (* LED'S (255) *)
SW_AMX_PRESET = 3 (* MOMENTARY, AND LINKED TO PAGE CHANGE. *)
SW_AMX_SPIN = 4 (* CONTINUOUSLY SENDS WHILST HELD DOWN. *)
SW_AMX_LEVEL = 5 (* THERE ARE 32 LEVELS - FADERS AND BARGRAPHS *)
SW_AMX_SOURCE = 6 (* SOURCE SELECTORS - MUTUALLY EXCLUSIVE *)
SW_AMX_TEXT = 7 (* THERE ARE 255 SUB TEXT FEEDBACKS *)

Message Types

SET_VALUE is sent from Soundweb to PANJA to indicate that a value has been changed which must be reflected on the PANJA controller. The same message from PANJA to Soundweb indicates that the user has moved a control which must be reflected on the Soundweb system.

On issuing a **SET_VALUE** the Soundweb to PANJA messaging code detects when a value has been changed as a result of activity on the PANJA system, but does NOT echo the changes. The PANJA system is expected to do the same - this is necessary in order to avoid an endless loop whereby the two systems send messages to each other continuously.

RAW_MSG is used to send raw Soundweb messages to and from external equipment via the RS232 port. The PANJA system ignores these messages.

SET_STRING is reserved for future use.

REQUEST_VALUE is reserved for future use.
This message would be sent by one or other side in order to demand a **SET_VALUE** for a particular parameter and would typically be used at start-up to ensure that the two sides are in sync. The other side responding with a **SET_VALUE** message.

REQUEST_STRING is reserved for future use.
This would be sent in order to demand a piece of text. The correct response is a **SET_STRING** message.

Protocol details

When a message is received successfully, an **ACK** is returned.
This should be done within one second of receiving the **ETX**.

When a message is received unsuccessfully, (determined by the checksum being incorrect or the frame incorrectly formed with start and end characters), a **NAK** is returned.
This should be done within one second of receiving the **ETX** (or the last character received).

If an **ACK** or **NAK** is not received within 1 second of sending a message, then the message should be resent.

Implementing the Soundweb/Panja protocol on other equipment

It is quite possible for other equipment to talk to a Soundweb device using the Soundweb/Panja protocol. It is simply a matter of implementing the protocol on the chosen platform.

Sending out a message

The following pseudo code sends a message by putting in escape characters, checksum **STX** and **ETX**.

```
SEND (STX)

CHAR CHECKSUM = 0

FOR EACH CHARACTER IN MESSAGE BODY
{
     CHECKSUM = CHECKSUM XOR CHARACTER

     IF (IS_SPECIAL (CHARACTER))
     {
          SEND (ESCAPE)
          SEND (CHARACTER + 128)
     }
     ELSE
     {
          SEND (CHARACTER)
     }
}

IF (IS_SPECIAL (CHECKSUM))
{
     SEND (ESCAPE)
     SEND (CHECKSUM + 128)
}
ELSE
{
     SEND (CHECKSUM )
}

SEND (ETX)

/* NOW WAIT FOR AN ACK OR NAK */
```

Receiving a message

The following pseudo code receives a message, takes out escape characters, and makes sure the message is valid by looking at the checksum.

```
/* TELLS US THAT THE PREVIOUS CHARACTER WAS ESCAPE*/
BOOL GOT_ESCAPE

CHAR CHECKSUM = 0

ON RECEIVED CHARACTER
{
        IF (CHARACTER = STX)
        {
                /* START OF MESSAGE */
                CHECKSUM = 0

                /* CLEAR THE MESSAGE BUFFER */
                CLEAR_MESSAGE_BUFFER()
                GOT_ESCAPE = FALSE
        }
        ELSE IF (CHARACTER = ETX)
        {
                /* END OF MESSAGE, CHECK THE CHECKSUM */
                IF (GET_LAST_BYTE_IN_MESSAGE_BUFFER() = CHECKSUM)
                {
                        /* THE MESSAGE IS OK */
                        SEND (ACK)
                }
                ELSE
                {
                        SEND (NAK)
                }
                GOT_ESCAPE = FALSE
        }
        ELSE IF (CHARACTER = ESCAPE)
        {
                GOT_ESCAPE = TRUE
        }
        ELSE
        {
                IF (GOT_ESCAPE = TRUE)
                {
                        ADD_BYTE_TO_MESSAGE_BUFFER (CHARACTER - 128)
                        CHECKSUM = CHECKSUM XOR (CHARACTER - 128)

                }
                ELSE
                {
                        ADD_BYTE_TO_MESSAGE_BUFFER (CHARACTER)
                        CHECKSUM = CHECKSUM XOR CHARACTER
                }

                GOT_ESCAPE = FALSE
        }
}
```

PANJA control

Equipment required:

PANJA AXCESS or NETLINX master device
PANJA touch panel or other input device
12V DC supply
AX link cable

PANJA touch panel

There are 255 available channels which can be assigned to buttons or icon objects on the panel. These allow presses to be processed and sent to Soundweb and for visual feedback to be echoed to the panel. In addition to this, there are 32 level channels which are linked to level controls or bar graph meters.

The library functions make use of a variable number for every control on the touch panel. How this variable number maps to a channel code is explained below for each control, the result being that some channel codes are not used. We recommend that channel codes for other devices in a shared system, i.e. one that has device control in addition to Soundweb on the same master, be allocated from outside the range of channel codes used by the export process, even if it appears they are not being used.

For example:
A level control requires a variable number to store it's value and subsequently this variable number will not be available for use as a channel code within the Panja to Soundweb communication system.

Source selectors require a channel code for each button in the exclusive group but only use one variable number to store the overall state.

Spin pairs require a channel code for each of the up and down buttons but have only one variable to store the current setting.

All other controls: presets and buttons have a direct mapping of channel code to variable number.

The master, AXB-EM232, is one of a number of available masters which stores and executes the AXCESS programming language.
The library, SWLIB.AXI has been written in such a way that it can be compiled for Netlinx or AXCESS.

Procedure

1. Having designed your Soundweb audio layout, open the **Serial** Input/Output map window.

2. Place the controls you wish to export to PANJA on this map window.

3. Go to the Serial menu and choose Panja export to generate the file **SWFUNC.AXI**.

This file contains functions to map Soundweb controls to equivalents on the touch panel. When opened in a text editor, the file will contain comments at the top describing how to assign channel codes on the touch panel.

4. Run the **AXCESS** compiler program or PANJA Studio and create a new project, importing the files **BSSMAIN.AXS**, **SWLIB.AXI** and **SWFUNC.AXI**.

5. **BSSMAIN.AXS** is supplied as an example minimum to get your system running. For Soundweb controls to coexist with controls for other systems the relevant calls from this file may be placed in another .AXS file or you may add extra functions to this one.
Avoid using the channel codes that have been allocated in **SWFUNC.AXI**.

6. Compile as **AXCESS** or **Netlinx** as appropriate (the export and library are compatible with both) and download to the **Master**.

Before anything will work, you must have buttons and levels on the panel with the correct channel codes as described in the **SWFUNC.AXI** comments.

7. Set up the buttons with the following attributes:
   - Toggling buttons are to be channel feedback.
   - Text objects have a variable text number and a channel number of zero.
   - Level objects have a unique level number from 1 to 32.
   - Preset change buttons can have a page change associated with them and are momentary feedback.
   - Spin pairs are momentary feedback.
   - Source selectors are momentary buttons and have mutually exclusive and latching attributes associated with them in the master. This is done automatically by the exported code.

8. Finally, compile the Soundweb layout and load the Soundweb device.
Upon loading, the back serial port will be configured for PANJA control (currently at 38,400 baud 8N1 and no handshaking) and messaging will commence.

Example code

Parts of **SWFUNC.AXI** will look like this:

```
(* --> ‘U1/P5/Mute’ *)
    PUSH [TP, 10]
         CALL ‘SW TOGGLE BUTTON’ (10,SW_TOGGLE)
    [TP,10] =  SW_CONTROL_VAL[10]
(* <-- *)
```

Placeholding comments surround each exported control:

```
(* --> *)
(* <-- *)
```

All calls to **SWLIB**, the library module, start with **SW**.

SWLIB.AXI provides the background handling of buttons, level controls, spin pairs, metering and all the messaging to the Soundweb unit. This has much more functionality as library functions than the exported code puts to use. It serves as an example of more complex control types that can be implemented on a PANJA touch panel.

All the designer need do to get a system up and running is to have an INCLUDE ‘SWFUNC.AXI’ statement at the top of the main file and the following two calls:

```
DEFINE_START
CALL ‘SW INIT CONTROLS’

DEFINE_PROGRAM
CALL ‘SW RUN CONTROLS’
```

In this way, Soundweb controls and background messaging can coexist with any other code in the master by being completely encapsulated in the include files.

Object detail

A toggle button is realised by using script for updating the feedback and sending the status to the Soundweb unit each time it changes. Similarly, Soundweb can toggle the control on the touch panel.

- An LED is a button with attributes that can be changed remotely. Its presses are not serviced.

- A preset is a momentary button which fires off a message. Soundweb Designer exports a list of associated controls in an exclusive group. Source selectors are set up in a similar way.

Controls in a parameter preset need to be set up as mutually exclusive. It is also necessary for these controls to be defined as latching. The PANJA master handles update of mutually exclusive controls so all that is required is an ON message for a particular source or preset and the others are cleared.

In order for the relevant controls to come in and out of scope for a particular preset, it is proposed that a preset change involves a change in touch panel page so that there is no chance of a user pressing a button that is not relevant for that preset. If the user changes page, it is up to the designer to ensure that the Soundweb unit is notified with a preset change at the same time.

This can be done by allocating the page change button a channel code and sending a preset change. SWFUNC contains some extra code associated with preset buttons for changing page.

Uncomment this and change to your page name:

```
(* --> ‘U1/Preset 1’ *)
    PUSH [TP, 20]
      CALL ‘SW RECALL PRESET’ (20)
(* uncomment the next line and change to your page name *)
(*      CALL ‘SW SEND PAGE CMD’ (“‘Page name’”)          *)
(* <-- *)
```

- A level control is an on-screen fader which is continuously monitored by AXCESS. Any changes in these controls will be sent to the Soundweb unit.

Interfacing to Non-PANJA Equipment

Most external equipment will use the RAW_MSG message to communicate with Soundweb, as this enables detailed and accurate control of every processing object. However, if you are using control equipment such as Crestron and Dataton, and require a simpler, less accurate method, **SET_VALUE** type message strings can be used by entering them into the design software.

To generate these strings and to serve as a test application, we have supplied the Soundweb message tool. This application allows you to construct messages with a simple dialogue and send them to a Soundweb device. The actual bytes sent are displayed in a text field in hexadecimal.
e.g.
SET_VALUE, Button, 1, 1 translates as:
2, 80, 0, 1, 0, 1, 80, 3
Hex 80 may also be written as $80 or 0x80 or &H80 or 80H, depending on your control system.

Debugging

A good way of debugging a system is to be connected to the front port of a Soundweb device from Soundweb Designer and to run the message tool connected to the rear port of the device. With this method, messages can be tested from both directions; sending from the unit by adjusting a control in Soundweb designer and by sending from the message tool. Messages sent from Soundweb will appear in the incoming box and serve as examples of message construction for sending from your piece of equipment, (since they will be the same).

Remember, start simple with perhaps just a couple of mute buttons to establish you have everything cabled correctly and each device configured correctly.

A Soundweb Designer file (Example.sdf) is supplied which has an example of each type of control that can be remotely controlled with this method.
Refer also to the Serial Interface Kit FAQ for more details.

Setting up a device for raw messages

Soundweb devices do not normally send and receive the Soundweb messages out of the rear RS232 port. Instead the back port is usually used as an alternative to the front port for connecting to Soundweb Designer. Therefore the device must be told to use the back port for Soundweb messages. This is done by renaming the device object in the layout so that the name ends with the characters '+s' e.g. ***"MyDevice +s"***. The device will need rebooting after the name change for the change to take effect.

NOTE: This naming convention is only used as a temporary device until a version of Soundweb Designer is released which allows the user to set the back port mode explicitly.

Message Format

The format of the RAW_MSG message is:

RAW_MSG <***Soundweb Message***>

where the format of the '*Soundweb Message*' is

<Handle> <Method> <Value>

where

- ***Handle*** (32 bit) is the identifier of the object that the message is to/from.
- ***Method*** (32 bit) is the object's 'method' that has changed.
- ***Value*** (16 or 32 bit) is the new value of the method.

NOTE: In the following paragraphs, numbers are represented in both decimal and hexadecimal depending on context. Hexadecimal numbers are preceded by the characters '0x'.
Any numbers NOT preceded by '0x' are decimal.

Processing Object Handles

Each audio processing object in a Soundweb device is given a network ***Handle***. This handle is a 32 bit unique number which allows Soundweb Designer to communicate with it. Once a device has been compiled, you can find the handles for the objects by placing the mouse over each object in turn, and reading the handle displayed in the status bar.

Processing Object Methods

A **method** of a processing object is a 'property'. For example, a gain processing object has 3 methods: Mute, Gain and Phase. If you look at the gain control panel you will see the three controls which are used to change those three methods. For processing object method numbers see Appendix A.

Method Values

Soundweb uses 16 bit and 32 bit methods values. These values are the actual settings for the objects controls. For example, the gain method might have a value of +3dB, or the mute method might have a value of OFF.

- Object methods come in two flavours ***continuous*** and ***discrete***.
  Continuous: Decibels, Hz, μs or scalar (generic floating point values).
  Discrete: Anything that can be represented by a fixed number of states. For example, a mute method has two states (on and off).

Nearly all continuous and discrete values are 16 bits. To represent a realistic range of values, each type is encoded in a different way.

Decibels

16 bits, a signed word in the range –32,768 to +32,767.
To find the value in dBs simply divide by 256.0.
Example
+1536 represents +6 dB.
(1536/256 = 6).

Hz

16 bits, unsigned word in the range 0 to 65535.
The frequency is 10 to the power of (value divided by 10000).
Example
33010 represents 2000 Hz.
(10 to the power of (33010/10000) = 2000).

µS

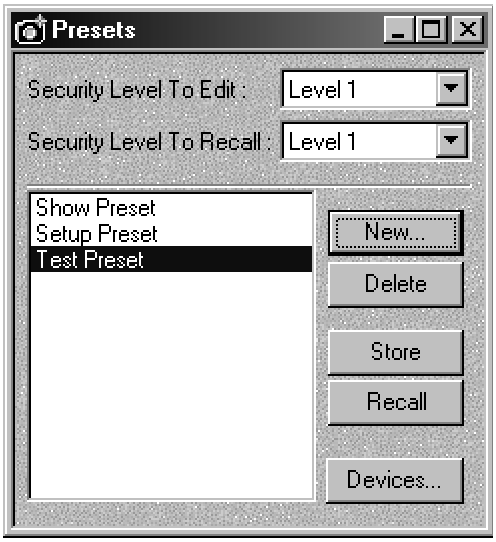
32 bits, unsigned long in the range 0 to 4,294,967,295.
Example
1530000 represents 1.53 seconds.

Scalar

16 bits, a signed word in the range –32,768 to 32,767.
To find the value simply divide by 256.
Example
320 represents -1.25.
(-320/256 = -1.25).

Activating Presets

Presets are activated by sending a broadcast message. A broadcast message has a handle of 0xFFFFFFFF. The method should be set to 0x0000000B and the value is the preset ID (32 bits). The preset ID is a unique identifier which is it's index in the list of presets.
To find a preset's ID goto the preset view and find the preset:



Security Level To Edit : Level 1
Security Level To Recall : Level 1

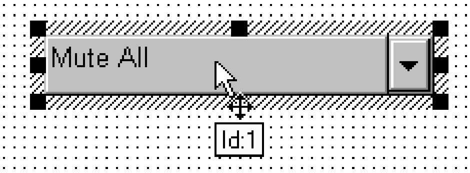
Show Preset
Setup Preset
Test Preset

Now counting from the top, the presets have the ID's 0, 1, 2, 3, 4 etc. So, the three presets shown above have the following Ids:

Show Preset	0
Setup Preset	1
Test Preset	2

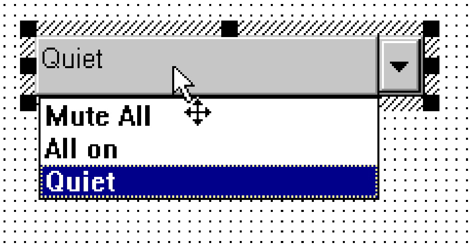
Activating Parameter Presets

Parameter presets are activated in a similar way, except that the preset ID is made up of two numbers, the *parameter preset ID*, and the *state*. The *parameter preset ID* can be found by letting the mouse pause over a parameter preset bar, and a ***tool tip*** will pop up with the ID:



Note: The *parameter preset ID* can only be found once the system has been loaded.

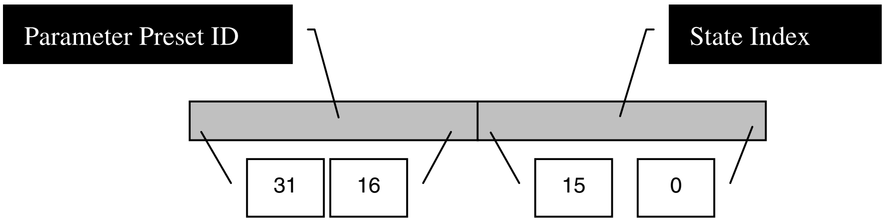
Once you have the *parameter preset ID*, you need to find the *state index* in the parameter preset:



The *state index* is simply its index in the list. So, for the above parameter preset the states have these indices

Mute All	0
All On	1
Quiet	2

Once you have found the *parameter preset ID*, and the *state index*, you are ready to create the preset ID. The preset ID is 32 bits long, 16 bits for the *parameter preset ID* and 16 bits for the *state index:*

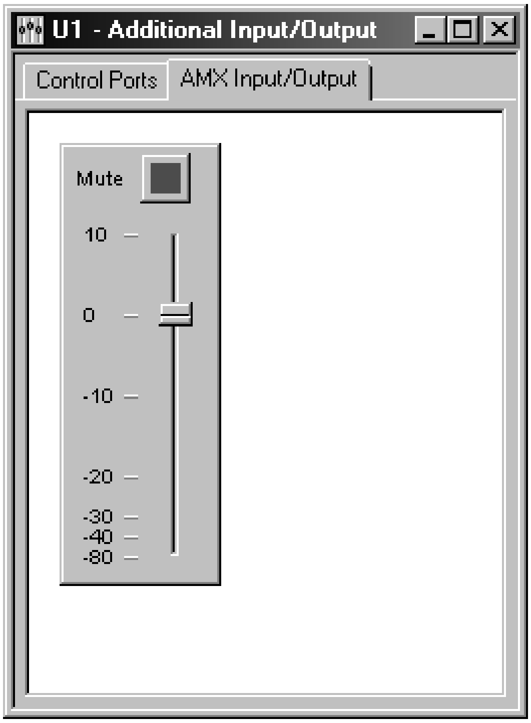


As with normal preset messages, Presets are activated by sending a broadcast message. A broadcast message has a handle of 0xFFFFFFFF. The method should be set to 0x0000000B and the value is the preset ID (32 bits).

Sending and Receiving Messages

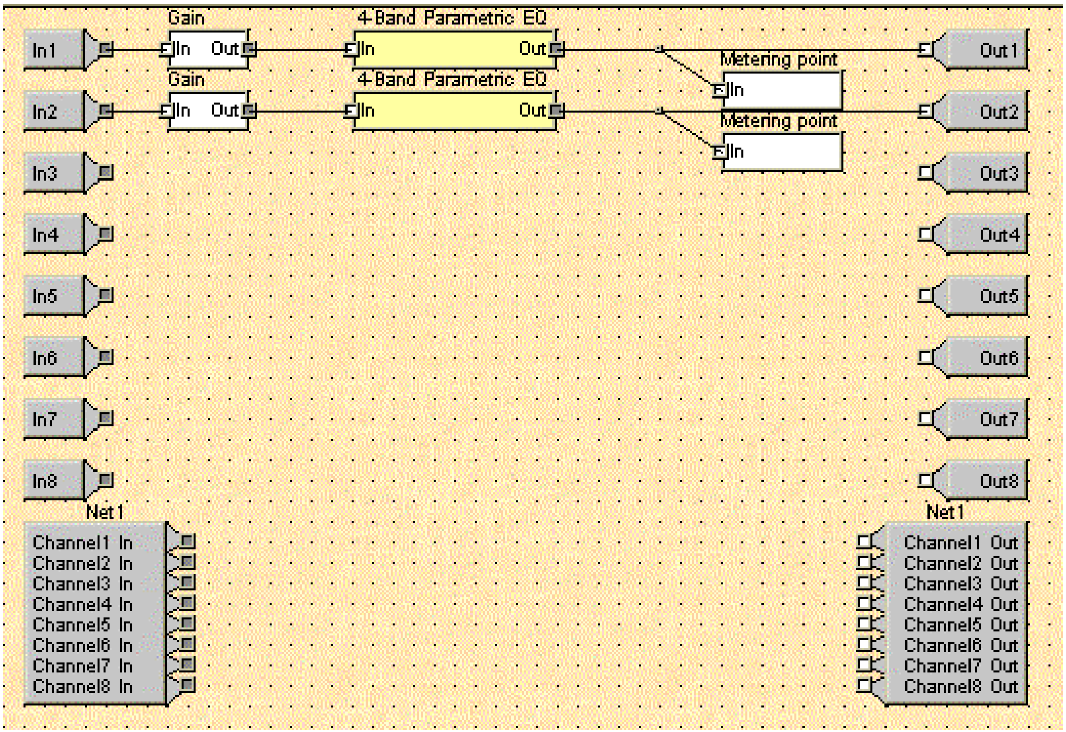
When a Soundweb device has been set up to send and receive Soundweb messages, then messages can be sent to any control via the RS232. However, when controls are changed via the PC or control ports a message will only be sent out of the RS232 if the control is placed on the Serial Input/Output panel of the device.

In the example Serial Input/Output panel below, the mute and level controls of a gain have been placed on U1's panel. Therefore, when the mute or level is changed a message will be sent out of U1's rear RS232 port. However, it is still possible to control other properties, such as phase, by sending messages to U1.



Example Messages

The following messages would be sent to the Soundweb device to control the following configuration



Object handles in ***italics***:

Gain Objects: Gain1 ***(2)*** and Gain2 ***(3)***

4-Band Parametric Eqs: Eq1 ***(4)*** and Eq2 ***(8)***

Metering Points: Meter1 ***(12)*** and Meter2 ***(13)***

Messages

To unmute Gain1:

Handle	Method Id	Value
2	0x01000801 (Mute)	0 (off)

To unmute Gain2:

Handle	Method Id	Value
3	0x01000801 (Mute)	0 (off)

Change Eq1 filter 1 to 2000Hz, +10.0 dB, Width 1.0, bell filter type:

Handle	Method Id	Value
4	0x01000801 (Frequency)	$\log_{10}$(2000.0)x10000
4	0x01000802 (BoostCut)	10x256
4	0x01000803 (Width)	1x256
4	0x01000804 (FilterType)	0

Change Eq1 filter 2 to 3000Hz, -5.0 dB, Width 2.0, bell filter type:

Handle	Method Id	Value
5	0x01000801 (Frequency)	$\log_{10}$(3000)x10000
5	0x01000802 (BoostCut)	-5x256
5	0x01000803 (Width)	2x256
5	0x01000804 (FilterType)	0

Example messages received from meters Meter1 and Meter2:

Handle	Method Id	Value	Decoded Value
12	0x01000800 (Level)	-2476	- 9.67 dB
13	0x01000800 (Level)	-8947	- 34.95 dB
12	0x01000800 (Level)	-10	- 0.039 dB
13	0x01000800 (Level)	+258	+ 1.008 dB

Activating preset 0:

Handle	Method Id	Value
0xFFFFFFFF	0x0000000B (Preset Activate)	0

Activating preset 5:

Handle	Method Id	Value
0xFFFFFFFF	0x0000000B (Preset Activate)	5

Appendix A - Processing Object Methods

Soundweb 9088 DSP

These are the methods for the 9088 DSP device. These control the input and output gains, phase and phantom power.

Method Name	Method Id	Type
Phase1	0x01000809	Discrete 0 = off 1 = on
Phase2	0x0100080a	Discrete 0 = off 1 = on
Phase3	0x0100080b	Discrete 0 = off 1 = on
Phase4	0x0100080c	Discrete 0 = off 1 = on
Phase5	0x0100080d	Discrete 0 = off 1 = on
Phase6	0x0100080e	Discrete 0 = off 1 = on
Phase7	0x0100080f	Discrete 0 = off 1 = on
Phase8	0x01000810	Discrete 0 = off 1 = on
Phantom1	0x010008011	Discrete 0 = off 1 = on
Phantom2	0x010008012	Discrete 0 = off 1 = on
Phantom3	0x010008013	Discrete 0 = off 1 = on
Phantom4	0x010008014	Discrete 0 = off 1 = on
Phantom5	0x010008015	Discrete 0 = off 1 = on
Phantom6	0x010008016	Discrete 0 = off 1 = on
Phantom7	0x010008017	Discrete 0 = off 1 = on
Phantom8	0x010008018	Discrete 0 = off 1 = on
LineGain1	0x01000819	Discrete 0 = 0dB 1=12dB
LineGain2	0x0100081a	Discrete 0 = 0dB 1=12dB
LineGain3	0x0100081b	Discrete 0 = 0dB 1=12dB
LineGain4	0x0100081c	Discrete 0 = 0dB 1=12dB
LineGain5	0x0100081d	Discrete 0 = 0dB 1=12dB
LineGain6	0x0100081e	Discrete 0 = 0dB 1=12dB
LineGain7	0x0100081f	Discrete 0 = 0dB 1=12dB
LineGain8	0x01000820	Discrete 0 = 0dB 1=12dB
MicGain1	0x01000821	Discrete. Values are 0=0, 1=6, 2=12, 3=18, 4=24, 5=30,6=36 7=42, 8=48, 9=54, 10=60, 11=66, 12=72
MicGain2	0x01000822	As above.
MicGain3	0x01000823	As above.
MicGain4	0x01000824	As above.
MicGain5	0x01000825	As above.
MicGain6	0x01000826	As above.
MicGain7	0x01000827	As above.
MicGain8	0x01000828	As above.
OutGain1	0x01000829	dB (-15 to +15)
OutGain2	0x0100082a	dB (-15 to +15)

OutGain3	0x0100082b	dB (-15 to +15)
OutGain4	0x0100082c	dB (-15 to +15)
OutGain5	0x0100082d	dB (-15 to +15)
OutGain6	0x0100082e	dB (-15 to +15)
OutGain7	0x0100082f	dB (-15 to +15)
OutGain8	0x02000830	dB (-15 to +15)

Gain

Method Name	Method Id	Type
Value	0x01000800	dB
Mute	0x01000801	0 = off, 1 = on
Phase	0x01000802	0 = off, 1 = on

Delay

Method Name	Method Id	Type
Value	0x02000800	ms

Parametric Eq

The parametric eq comes in 5 different configurations: 1, 2, 4, 6 and 12 bands. Each band is a separate processing object, and therefore has it's own *handle*. Soundweb will display the handle for the first eq band on the status bar when the user moves the mouse over the parametric object. The other bands in the parametric eq can be found by incrementing the handle for each band. For example, if a 4 band parametric eq displays a handle of 23, then the individual eq band objects have handles of 23, 24, 25 and 26. So, to change filter 3, send messages to object 25.

Each eq band has the follow methods:

Method Name	Method Id	Type	
Bypass	0x01000800	0 = off, 1 = on	
Frequency	0x01000801	Hz	
BoostCut	0x01000802	dB	
Width	0x01000803	Scalar (0.05 - 3.0)	
FilterType	0x01000804	Discrete. Values are:	0 = Bell 1 = High Shelf 2 = Low Shelf

Crossover

Like the parametric eqs, the crossover comprises of several individual bands which are separate objects. There are currently 5 different crossovers defined :- 1,2,3,4 and 5 bands. There is also the mon sub object which is basically a 2:1 mixer followed by a crossover band.

Method Name	Method Id	Type
LowPassType	0x01000800	Band type (see below)
HighPassType	0x01000801	Band type (see below)
LowPassFrequency	0x01000802	Hz
HighPassFrequency	0x01000803	Hz
BandGain	0x01000804	dB
Threshold	0x01000805	dB

Band types are:

0 = Out , **1** = But, **2** = 12But, **3** = 12Bess, **4** = 12LR, **5** = 18But, **6** = 24But, **7** = 24Bess, **8** = 24LR, **9** = 48But, **10** = 48LR.

The monosub crossover object is a 2:1 mixer followed by a crossover band. The mixer's handle is 1+ the crossover band's handle. For the 2:1 mixer methods see the section headed **Mixers**.

Mixers

All the mixers share the same basic format, each input chanel has mute, gain and phase controls (plus pan if a stereo mixer). The mixer will have one output if mono or two if stereo. Each output has a mute and level. Optionally a mixer can have 1 or 2 aux buses which adds extra level controls for each input.

Method Name	Method Id	Type
Gain1	0x01000800	dB
Gain2	0x01000801	dB
Gain3	0x01000803	dB
Gain4	0x01000804	dB
Gain5	0x01000805	dB
Gain6	0x01000806	dB
Gain7	0x01000807	dB
Gain8	0x01000808	dB
Gain9	0x01000869	dB
Gain10	0x0100086a	dB
Gain11	0x0100086b	dB
Gain12	0x0100086c	dB
Gain13	0x0100086d	dB
Gain14	0x0100086e	dB
Gain15	0x0100086f	dB
Gain16	0x01000870	dB
GainOut (mono)	0x01000809	dB
GainOutL (stereo)	0x0100080a	dB
GainOutR (stereo)	0x0100080b	dB
Pan1	0x01000810	Scalar (0.0 - 1.0)
Pan2	0x01000811	Scalar (0.0 - 1.0)
Pan3	0x01000812	Scalar (0.0 - 1.0)
Pan4	0x01000813	Scalar (0.0 - 1.0)
Pan5	0x01000814	Scalar (0.0 - 1.0)
Pan6	0x01000815	Scalar (0.0 - 1.0)
Pan7	0x01000816	Scalar (0.0 - 1.0)
Pan8	0x01000817	Scalar (0.0 - 1.0)
Mute1	0x01000818	0 = off, 1 = on
Mute2	0x01000819	0 = off, 1 = on
Mute3	0x0100081a	0 = off, 1 = on
Mute4	0x0100081b	0 = off, 1 = on
Mute5	0x0100081c	0 = off, 1 = on
Mute6	0x0100081d	0 = off, 1 = on
Mute7	0x0100081e	0 = off, 1 = on
Mute8	0x0100081f	0 = off, 1 = on

Mute9	0x01000850	0 = off, 1 = on
Mute10	0x01000851	0 = off, 1 = on
Mute11	0x01000852	0 = off, 1 = on
Mute12	0x01000853	0 = off, 1 = on
Mute13	0x01000854	0 = off, 1 = on
Mute14	0x01000855	0 = off, 1 = on
Mute15	0x01000856	0 = off, 1 = on
Mute16	0x01000857	0 = off, 1 = on
MuteOut (mono)	0x01000828	0 = off, 1 = on
MuteOutL (stereo)	0x01000829	0 = off, 1 = on
MuteOutR (stereo)	0x0100082a	0 = off, 1 = on
Phase1	0x01000880	0 = off, 1 = on
Phase2	0x01000881	0 = off, 1 = on
Phase3	0x01000882	0 = off, 1 = on
Phase4	0x01000883	0 = off, 1 = on
Phase5	0x01000884	0 = off, 1 = on
Phase6	0x01000885	0 = off, 1 = on
Phase7	0x01000886	0 = off, 1 = on
Phase8	0x01000887	0 = off, 1 = on
Phase9	0x01000888	0 = off, 1 = on
Phase10	0x01000889	0 = off, 1 = on
Phase11	0x0100088a	0 = off, 1 = on
Phase12	0x0100088b	0 = off, 1 = on
Phase13	0x0100088c	0 = off, 1 = on
Phase14	0x0100088d	0 = off, 1 = on
Phase15	0x0100088e	0 = off, 1 = on
Phase16	0x0100088f	0 = off, 1 = on
AuxGain1a	0x01000840	dB
AuxGain1b	0x01000841	dB
AuxGain2a	0x01000842	dB
AuxGain2b	0x01000843	dB
AuxGain3a	0x01000844	dB
AuxGain3b	0x01000845	dB
AuxGain4a	0x01000846	dB
AuxGain4b	0x01000847	dB
AuxGain5a	0x01000848	dB
AuxGain5b	0x01000849	dB
AuxGain6a	0x0100084a	dB
AuxGain6b	0x0100084b	dB
AuxGain7a	0x0100084c	dB
AuxGain7b	0x0100084d	dB
AuxGain8a	0x0100084e	dB

AuxGain8b	0x0100084f	dB
AuxGain9a	0x01000858	dB
AuxGain9b	0x01000859	dB
AuxGain10a	0x0100085a	dB
AuxGain10b	0x0100085b	dB
AuxGain11a	0x0100085c	dB
AuxGain11b	0x0100085d	dB
AuxGain12a	0x0100085e	dB
AuxGain12b	0x0100085f	dB
AuxGain13a	0x01000860	dB
AuxGain13b	0x01000861	dB
AuxGain14a	0x01000862	dB
AuxGain14b	0x01000863	dB
AuxGain15a	0x01000864	dB
AuxGain15b	0x01000865	dB
AuxGain16a	0x01000866	dB
AuxGain16b	0x01000867	dB
AuxPreFade	0x01000868	0 = off, 1 = on

Matrix Routers and Mixers

To save space in this document, the methods are listed in groups of 48 buttons. Each group corresponds to a row of buttons or rotaries on the control panels. To find the method of a particular button or rotary, simply look down the table for the group corresponding to the output number (e.g. Output 5 methods start at 0x010010C0). Take the input number, subtract one, and add it to the start method. So, the button for **Output 5 In 4** has a method of 0x010010C0+4-1 = 0x010010C3.

Methods	**Method Ids**	**Type**
Route inputs (1-32) to output 1	0x01001000 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 2	0x01001030 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 3	0x01001060 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 4	0x01001090 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 5	0x010010C0 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 6	0x010010F0 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 7	0x01001120 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 8	0x01001150 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 9	0x01001180 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 10	0x010011B0 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 11	0x010011E0 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 12	0x01001210 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 13	0x01001240 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 14	0x01001270 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 15	0x010012A0 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 16	0x010012D0 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 17	0x01001300 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 18	0x01001330 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 19	0x01001360 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 20	0x01001390 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 21	0x010013C0 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 22	0x010013F0 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 23	0x01001420 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 24	0x01001450 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 25	0x01001480 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 26	0x010014B0 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 27	0x010014E0 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 28	0x01001510 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 29	0x01001540 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 30	0x01001570 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 31	0x010015A0 + (Input-1)	0 = off, 1 = on
Route inputs (1-32) to output 32	0x010015D0 + (Input-1)	0 = off, 1 = on
Mix inputs (1-32) to output 1	0x01000800 + (Input-1)	dB

Mix inputs (1-32) to output 2	0x01000830 + (Input-1)	dB
Mix inputs (1-32) to output 3	0x01000860 + (Input-1)	dB
Mix inputs (1-32) to output 4	0x01000890 + (Input-1)	dB
Mix inputs (1-32) to output 5	0x010008C0 + (Input-1)	dB
Mix inputs (1-32) to output 6	0x010008F0 + (Input-1)	dB
Mix inputs (1-32) to output 7	0x01000920 + (Input-1)	dB
Mix inputs (1-32) to output 8	0x01000950 + (Input-1)	dB
Mix inputs (1-32) to output 9	0x01000980 + (Input-1)	dB
Mix inputs (1-32) to output 10	0x010009B0 + (Input-1)	dB
Mix inputs (1-32) to output 11	0x010009E0 + (Input-1)	dB
Mix inputs (1-32) to output 12	0x01000A10 + (Input-1)	dB
Mix inputs (1-32) to output 13	0x01000A40 + (Input-1)	dB
Mix inputs (1-32) to output 14	0x01000A70 + (Input-1)	dB
Mix inputs (1-32) to output 15	0x01000AA0 + (Input-1)	dB
Mix inputs (1-32) to output 16	0x01000AD0 + (Input-1)	dB
Mix inputs (1-32) to output 17	0x01000B00 + (Input-1)	dB
Mix inputs (1-32) to output 18	0x01000B30 + (Input-1)	dB
Mix inputs (1-32) to output 19	0x01000B60 + (Input-1)	dB
Mix inputs (1-32) to output 20	0x01000B90 + (Input-1)	dB
Mix inputs (1-32) to output 21	0x01000BC0 + (Input-1)	dB
Mix inputs (1-32) to output 22	0x01000BF0 + (Input-1)	dB
Mix inputs (1-32) to output 23	0x01000C20 + (Input-1)	dB
Mix inputs (1-32) to output 24	0x01000C50 + (Input-1)	dB
Mix inputs (1-32) to output 25	0x01000C80 + (Input-1)	dB
Mix inputs (1-32) to output 26	0x01000CB0 + (Input-1)	dB
Mix inputs (1-32) to output 27	0x01000CE0 + (Input-1)	dB
Mix inputs (1-32) to output 28	0x01000D10 + (Input-1)	dB
Mix inputs (1-32) to output 29	0x01000D40 + (Input-1)	dB
Mix inputs (1-32) to output 30	0x01000D70 + (Input-1)	dB
Mix inputs (1-32) to output 31	0x01000DA0 + (Input-1)	dB
Mix inputs (1-32) to output 32	0x01000DD0 + (Input-1)	dB

Graphic Eq

Method Name	Method Id	Type
BandGain1	0x01000820	dB
BandGain2	0x01000821	dB
BandGain3	0x01000822	dB
BandGain4	0x01000823	dB
BandGain5	0x01000824	dB
BandGain6	0x01000825	dB
BandGain7	0x01000826	dB
BandGain8	0x01000827	dB
BandGain9	0x01000828	dB
BandGain10	0x01000829	dB
BandGain11	0x0100082a	dB
BandGain12	0x0100082b	dB
BandGain13	0x0100082c	dB
BandGain14	0x0100082d	dB
BandGain15	0x0100082e	dB
BandGain16	0x0100082f	dB
BandGain17	0x01000830	dB
BandGain18	0x01000831	dB
BandGain19	0x01000832	dB
BandGain20	0x01000833	dB
BandGain21	0x01000834	dB
BandGain22	0x01000835	dB
BandGain23	0x01000836	dB
BandGain24	0x01000837	dB
BandGain25	0x01000838	dB
BandGain26	0x01000839	dB
BandGain27	0x0100083a	dB
BandGain28	0x0100083b	dB
BandGain29	0x0100083c	dB
BandGain30	0x0100083d	dB
Selectivity	0x01000841	Scalar (1.0 - 1.5)
Bypass	0x01000842	0 = off, 1 = on

Metering Point

To implement metering in an external device it must be able to receive meter messages. These will be ***MeterLevel*** messages from a metering point object.

Method Name	Method Id	Type
MeterLevel	0x01000800	dB
MeterAttack	0x02000801	ms
MeterRelease	0x02000802	ms
MeterReference	0x01000803	dB

Source Selector

Method Name	Method Id	Type
SourceSel	0x01000800	Discrete. Values are: 0 = none 1 = Input 1 2 = Input 2 3 = Input 3 4 = Input 4 5 = Input 5 6 = Input 6 7 = Input 7 8 = Input 8 9 = Input 9 10 = Input 10 11= Input 11 12 = Input 12 13 = Input 13 14 = Input 14 15 = Input 15 16 = Input 16

Compressor

Method Name	Method Id	Type
CompBypass	0x01000800	Discrete . Values are 0=off , 1 = bypass
CompThreshold	0x01000801	dB
CompRatio	0x01000802	Scalar
CompAttack	0x02000803	ms
CompRelease	0x02000804	ms
CompGainReduction	0x01000805	dB
CompGain	0x01000807	dB
CompAutoRelease	0x01000808	Discrete . Values are 0=off, 1 = Auto Release

Limiter

Method Name	Method Id	Type
LimThreshold	0x01000801	dB
LimAttack	0x02000803	ms
LimRelease	0x02000804	ms

Leveller

Method Name	Method Id	Type
LevBypass	0x01000800	Disctrete. Values are 0 = off, 1 = Bypass
LevRatio	0x01000801	Scalar
LevThreshold	0x01000802	dB
LevMeter	0x01000804	dB
LevTarget	0x01000805	dB
LevMax	0x01000806	dB
LevSpeed	0x02000807	ms
LevActive	0x01000808	Disctrete. Values are 0 = off, 1 = Leveller Active

Filter High Pass And Low Pass

Method Name	Method Id	Type
FilterBypass	0x01000800	Disctrete. Values are 0 = off, 1 = bypass
Filter Frequency	0x01000801	dB
FilterType	0x01000804	Discrete. Values are 0=Out 1=6But 2=12But 3=12Bess 4=12LR 5=18But 6=24But 7=24Bess 8=24LR 9=48But 10=48LR