Export DSP Data to Text

A Nexia™* system, as shown in its document view, consists of DSP blocks, each of which is identified by a unique object code. Each block contains data that will describe its characteristics to the DSPs when the overall system configuration is sent to the Nexia™ units. The DSP data found in these blocks may be exported from the software in a textual format useful for printing or for automatic processing. This document describes the form this output will take.

When the ***Export DSP data to text*** command is selected from Nexia's file menu, under the subheading ***Export***, the blocks are sorted in alphabetical order by their DSP object codes, including the numbers, and each block in turn is output to a comma-delimited text file, suitable to be imported into another application. Except for an initial line giving the format version number, no header or footer information is included; only the DSP data is written.

The format version number identifies changes to the general output format of the DSP data. It is not associated with a specific document. The format version number will be incremented if the text output format for the DSP blocks changes, and may be used, when upgrading the Nexia™ software, to ensure backward compatibility with existing automated scripts which process the text file. The format version number is an integral value, which will be written to the first line of the output text file, preceded by an identifier and the application name:

FileFormatVersion, Nexia, 1

Every line of the text output—except the first, which gives the version—begins with the object code of the DSP block, followed by a block line number. Some characteristics of a DSP block apply to all types. This information is written as the first line (line number 0) of the output for that block, and may be used by automated software to determine what follows. Line number 0 is written for every block. Almost all blocks will have at least one additional line; many will have more than one, depending on their channel or band count. Blocks that have no DSP data, such as text blocks, will include line 0 only.

Each DSP object takes a different form after the first line. The fields which are written for each of them will be described in tables below, with the exception of line 0, which is described here, and has the following fields:

Object code, 0 (line number), "class code", "title", input control count, output control count, input signal count, output signal count, unit number, gang number, sample rate

In most cases, the object codes describe the object uniquely; however, in a few instances, only the class of object is shown in the object code. It is then necessary to distinguish the

types of objects by their class code. For example, all audio input and output blocks have the general class code **I/O**; a number is appended to this, in ascending order, whenever new input or output blocks are created. This same base code is used for all Inputs and Outputs, whether mono-channel or stereo. The class codes for these various types of blocks will differ.

Text fields will be surrounded by double quotes (“); the double quotes are shown in the table, as is done above around the title field, and serve to indicate which fields are textual data. All other fields, including Boolean values, those that are either true or false, and enumerated values, are represented as numbers. Boolean values will be zero (0) for false values, and one (1) for true. The object code that starts each line is not written out in double quotes. Some object codes may include spaces.

The specific format for each object type is given in the tables that follow. Subscripts indicate the channel or band number of the data they are associated with. The number N indicates the data for the last line of the output, corresponding to the number of channels or bands. For two-dimensional objects such as mixers and routers, N will be used for the input count, and M for the output count. Normally the channel count is given in line 1 of the output. When N is shown in an expression such as N+1, this indicates that the usage does not match the number of bands or channels; in the case of an Input block below, for instance, it indicates a line number one greater than the number of channels.

Each table below first gives the block type, then lists the fields included in each line of the output for that block type. The first field is always the object code. In each table, the object code will be given as for the actual output, but the number following the object code will always be 1. Thus, for example, each line of the table for an Input block will be shown starting with **I/O1**.

If the value for a field is known, or required in the context, the value will be shown, and the field name will follow it in parentheses; this is done in the description of line 0 above.

Lines may be quite long, and single channel objects will typically all appear on a single line, which may be up to several hundred characters. Hence, the lines in the tables below may wrap around; however, if the next line does not start with the object code, it is actually part of the line above.

The tables that follow give a block type name in bold letters, and then list the fields, on successive lines, that are part of the text output for the given type of DSP block. The objects are listed in the order that they appear in the toolbar menu.

Text:

No DSP data beyond line 0 is included. The base object code is Text. The text of the block is shown in the title field..

Input:

I/O1, 1, N (input channel count)
I/O1, 2, $gain_1$, phantom $power_1$, $mute_1$, $level_1$, $invert_1$
I/O1, 3, $gain_2$, phantom $power_2$, $mute_2$, $level_2$, $invert_2$
I/O1, 4, $gain_3$, phantom $power_3$, $mute_3$, $level_3$, $invert_3$
…
I/O1, N+1, $gain_N$, phantom $power_N$, $mute_N$, $level_N$, $invert_N$

Note that phantom power will be omitted if mic-line levels are not allowed, such as for SP units.

Output:

I/O1, 1, N (output channel count), is mic level output allowed
I/O1, 2, $gain_1$, $mute_1$, $level_1$, $invert_1$, full $scale_1$
I/O1, 3, $gain_2$, $mute_2$, $level_2$, $invert_2$, full $scale_2$
I/O1, 4, $gain_3$, $mute_3$, $level_3$, $invert_3$, full $scale_3$
…
I/O1, N+1, $gain_N$, $mute_N$, $level_N$, $invert_N$, full $scale_N$

If mic level output is allowed, then each channel may include full scale values of –31 dBu.

Input, Stereo:

I/O1, 1, 12 (input channel count)
I/O1, 2, $gain_1$, $mute_1$, $level_1$, $invert_1$, $stereo_1$
I/O1, 3, $gain_2$, $mute_2$, $level_2$, $invert_2$, $stereo_2$
I/O1, 4, $gain_3$, $mute_3$, $level_3$, $invert_3$, $stereo_3$
…
I/O1, 13, $gain_{12}$, $mute_{12}$, $level_{12}$, $invert_{12}$, $stereo_{12}$

Output, Stereo:

I/O1, 1, 6 (output channel count)
I/O1, 2, $gain_1$, $mute_1$, $level_1$, $invert_1$, full $scale_1$, $stereo_1$
I/O1, 3, $gain_2$, $mute_2$, $level_2$, $invert_2$, full $scale_2$, $stereo_2$
I/O1, 4, $gain_3$, $mute_3$, $level_3$, $invert_3$, full $scale_3$, $stereo_3$
…
I/O1, 7, $gain_6$, $mute_6$, $level_6$, $invert_6$, full $scale_6$, $stereo_6$

Auto Mixer:

Mixer1, 1, N (input channel count), input $mute_1$, input $level_1$,
input $mute_2$, input $level_2$, input $mute_3$, input $level_3$, …, input $mute_N$, input $level_N$
Mixer1, 2, 1 (output channel count), output mute, output level
Mixer1, 3, is direct out enabled, mic logic, logic outs follow mic logic,
open mic limits enabled, open mic limits
Mixer1, 4,
$state_1$, is $manual_1$, enable NOM $gain_1$, off $attenuation_1$, gate hold $time_1$, output $type_1$,
$state_2$, is $manual_2$, enable NOM $gain_2$, off $attenuation_2$, gate hold $time_2$, output $type_2$,
$state_3$, is $manual_3$, enable NOM $gain_3$, off $attenuation_3$, gate hold $time_3$, output $type_3$,
…,
$state_N$, is $manual_N$, enable NOM $gain_N$, off $attenuation_N$, gate hold $time_N$, output $type_N$
Mixer1, 5, L (logic out count), logic $invert_1$, gate $logic_1$, logic $invert_2$, gate $logic_2$,
logic $invert_3$, gate $logic_3$, …, logic $invert_L$, gate $logic_L$

In line 3, mic logic is either 0 (None), 1 (Last Mic Hold), or one more than the channel to follow. In line 4, the output type may be either 0 (post gate / pre-NOM), or 1 (post gate / pre-NOM). In line 5, the gate logic value will be either 0 (follow gate), 1 (on), or 2 (off). Note that L may be 0, in which nothing will follow it on line 5.

Standard Mixer:

Mixer1, 1, N (input channel count), input $mute_1$, input $level_1$,
input $mute_2$, input $level_2$, input $mute_3$, input $level_3$, …, input $mute_N$, input $level_N$
Mixer1, 2, M (output channel count), output mute output$_1$, output $level_1$,
output $mute_2$, output $level_2$, output $mute_3$, output $level_3$, …,
output $mute_M$, output $level_M$
Mixer1, 3, $state_{1,1}$, $state_{1,2}$, $state_{1,3}$, …, $state_{1,M}$
Mixer1, 4, $state_{2,1}$, $state_{2,2}$, $state_{2,3}$, …, $state_{2,M}$
Mixer1, 5, $state_{3,1}$, $state_{3,2}$, $state_{3,3}$, …, $state_{3,M}$
…
Mixer1, N+2, $state_{N,1}$, $state_{N,2}$, $state_{N,3}$, …, $state_{N,M}$

Matrix Mixer:

Mixer1, 1, N (input channel count), input $mute_1$, input $level_1$,
input $mute_2$, input $level_2$, input $mute_3$, input $level_3$, …, input $mute_N$, input $level_N$
Mixer1, 2, M (output channel count), output mute output$_1$, output $level_1$,
output $mute_2$, output $level_2$, output $mute_3$, output $level_3$, …,
output $mute_M$, output $level_M$
Mixer1, 3, is delay enabled
Mixer1, 4, $state_{1,1}$, $level_{1,1}$, delay $state_{1,1}$, delay $level_{1,1}$,
$state_{1,2}$, $level_{1,2}$, delay $state_{1,2}$, delay $level_{1,2}$,
$state_{1,3}$, $level_{1,3}$, delay $state_{1,3}$, delay $level_{1,3}$, …,
$state_{1,M}$, $level_{1,M}$, delay $state_{1,M}$, delay $level_{1,M}$
Mixer1, 5, $state_{2,1}$, $level_{2,1}$, delay $state_{2,1}$, delay $level_{2,1}$,
$state_{2,2}$, $level_{2,2}$, delay $state_{2,2}$, delay $level_{2,2}$,
$state_{2,3}$, $level_{2,3}$, delay $state_{2,3}$, delay $level_{2,3}$, …,
$state_{2,M}$, , $level_{2,M}$, delay $state_{2,M}$, delay $level_{2,M}$
Mixer1, 6, $state_{3,1}$, $level_{3,1}$, delay $state_{3,1}$, delay $level_{3,1}$,
$state_{3,2}$, $level_{3,2}$, delay $state_{3,2}$, delay $level_{3,2}$,
$state_{3,3}$, $level_{3,3}$, delay $state_{3,3}$, delay $level_{3,3}$, …,
$state_{3,M}$, , $level_{3,M}$, delay $state_{3,M}$, delay $level_{3,M}$
…
Mixer1, N+3, $state_{N,1}$, $level_{N,1}$, delay $state_{N,1}$, delay $level_{N,1}$,
$state_{N,2}$, $level_{N,2}$, delay $state_{N,2}$, delay $level_{N,2}$,
$state_{N,3}$, $level_{N,3}$, delay $state_{N,3}$, delay $level_{N,3}$, …,
$state_{N,M}$, $level_{N,M}$, delay $state_{N,M}$, delay $level_{N,M}$

Mix-Minus Combiner:

Combiner1, 1, N (input channel count)
Combiner1, 2, sound $space_1$, sound $space_2$, sound $space_3$, …, sound $space_N$

Room Combiner:

RoomCombiner1, 1, column count, row count, N (active cell count)
RoomCombiner1, 2, cell index$_1$, sound zone$_1$, mute$_1$, level$_1$, "room name$_1$"
RoomCombiner1, 3, cell index$_2$, sound zone$_2$, mute$_2$, level$_2$, "room name$_2$"
RoomCombiner1, 4, cell index$_3$, sound zone$_3$, mute$_3$, level$_3$, "room name$_3$"
…
RoomCombiner1, N+1, cell index$_N$, sound zone$_N$, mute$_N$, level$_N$, "room name$_N$"

The active cell count may (and probably will), be less than the column count multiplied by the row count. If, for example, the grid has eight columns and four rows, but only 19 active cells, six of which are combined into two rooms, then the active cell count would be 15 (19 – 6 + 2 < 8 ? 4); thus there would be 16 lines of output.

The cell index is the number obtained by counting cells from left to right and from top to bottom across the grid, whether or not they are active. The cell index for a room is the index of the first active cell encountered for that room. In our example combiner, cell index 11 would always be the third cell of the second row; it will not occur in the output, however, if that cell is inactive or not the first cell of a room.

Equalizer (Parametric or Graphic):

Equalizer1, 1, N (band count), is all bypassed
Equalizer1, 2, frequency$_1$, frequency$_2$, frequency$_3$, …, frequency$_N$
Equalizer1, 3, band width$_1$, band width$_2$, band width$_3$, …, band width$_N$
Equalizer1, 4, gain$_1$, gain$_2$, gain$_3$, …, gain$_N$
Equalizer1, 5, bypass$_1$, bypass$_2$, bypass$_3$, …, bypass$_N$

Feedback Suppressor:

Feedback Suppressor1, 1, N (band count), is all bypassed
Feedback Suppressor1, 2, frequency$_1$, frequency$_2$, frequency$_3$, …, frequency$_N$
Feedback Suppressor1, 3, band width$_1$, band width$_2$, band width$_3$, …, band width$_N$
Feedback Suppressor1, 4, gain$_1$, gain$_2$, gain$_3$, …, gain$_N$
Feedback Suppressor1, 5, bypass$_1$, bypass$_2$, bypass$_3$, …, bypass$_N$
Feedback Suppressor1, 6, is fixed$_1$, is fixed$_2$, is fixed$_3$, …, is fixed$_N$

Filter (High- and low-pass):

Filter1, 1, frequency, gain, filter type, slope, maximum slope, is bypassed

Filter type is 1 for Butterworth filters, 2 for Linkwitz-Riley.

Filter (High and low shelf):

Filter1, 1, frequency, gain, is bypassed

Filter (All-Pass):

Filter1, 1, maximum band count, N (band count), is all bypassed
Filter1, 2, $frequency_1$, $frequency_2$, $frequency_3$, …, $frequency_N$
Filter1, 3, $band\ width_1$, $band\ width_2$, $band\ width_3$, …, $band\ width_N$
Filter1, 4, $bypass_1$, $bypass_2$, $bypass_3$, …, $bypass_N$

2-Way Crossover:

Crossover1, 1, 1 (crossover count), input level, input mute, maximum slope, keep synchronized
Crossover1, 2, $level_L$, $mute_L$, $polarity_L$, low-pass $frequency_L$, low-pass $filter\ type_L$, low-pass $slope_L$
Crossover1, 3, high-pass $frequency_H$, high-pass $filter\ type_H$, high-pass $slope_H$, $level_H$, $mute_H$, $polarity_H$

Filter type is 1 for Butterworth filters, 2 for Linkwitz-Riley. Polarity is 1 when the signal is inverted, 0 otherwise.

3-Way Crossover:

Crossover1, 1, 2 (crossover count), input level, input mute, maximum slope, keep synchronized
Crossover1, 2, $level_L$, $mute_L$, $polarity_L$, low-pass $frequency_L$, low-pass $filter\ type_L$, low-pass $slope_L$
Crossover1, 3, high-pass $frequency_M$, high-pass $filter\ type_M$, high-pass $slope_M$, $level_M$, $mute_M$, $polarity_M$, low-pass $frequency_M$, low-pass $filter\ type_M$, low-pass $slope_M$
Crossover1, 4, high-pass $frequency_H$, high-pass $filter\ type_H$, high-pass $slope_H$, $level_H$, $mute_H$, $polarity_H$

Filter type is 1 for Butterworth filters, 2 for Linkwitz-Riley. Polarity is 1 when the signal is inverted, 0 otherwise.

4-Way Crossover:

Crossover1, 1, 3 (crossover count), input level, input mute, maximum slope, keep synchronized
Crossover1, 2, $level_L$, $mute_L$, $polarity_L$, low-pass $frequency_L$, low-pass $filter\ type_L$, low-pass $slope_L$
Crossover1, 3, high-pass $frequency_{LM}$, high-pass $filter\ type_{LM}$, high-pass $slope_{LM}$, $level_{LM}$, $mute_{LM}$, $polarity_{LM}$, low-pass $frequency_{LM}$, low-pass $filter\ type_{LM}$, low-pass $slope_{LM}$
Crossover1, 4, high-pass $frequency_{MH}$, high-pass $filter\ type_{MH}$, high-pass $slope_{MH}$, $level_{MH}$, $mute_{MH}$, $polarity_{MH}$, low-pass $frequency_{MH}$, low-pass $filter\ type_{MH}$, low-pass $slope_{MH}$
Crossover1, 5, high-pass $frequency_H$, high-pass $filter\ type_H$, high-pass $slope_H$, $level_H$, $mute_H$, $polarity_H$

Filter type is 1 for Butterworth filters, 2 for Linkwitz-Riley. Polarity is 1 when the signal is inverted, 0 otherwise.

Leveler:

Leveler1, 1, 1 (input channel count), response time, threshold, "ID", is bypassed

Compressor/Limiter:

CompLimiter1, 1, 1 (input channel count), attack time, release time, ratio, threshold, "ID", is bypassed

Ducker:

Ducker1, 1, mute, input level, sense is muted, sense level, threshold, ducking level, is bypassed, logic in is enabled, logic out is enabled, logic in is inverted, logic out is inverted, attack time, release time

Noise Gate:

NoiseGate1, 1, 1 (input channel count), attack time, release time, threshold, "ID", is bypassed

ANC:

ANC1, 1, program mute, program level, "program ID", ambient mute, ambient level, ambient threshold, ambient ramp rate, "ambient ID", gain minimum, gain maximum, gain ratio, gain time, "gain ID", is bypassed

Router:

Router1, 1, N (input channel count), M (output channel count)
Router1, 2, $node_{1,1}$, $node_{1,2}$, $node_{1,3}$, …, $node_{1,M}$
Router1, 3, $node_{2,1}$, $node_{2,2}$, $node_{2,3}$, …, $node_{2,M}$
Router1, 4, $node_{3,1}$, $node_{3,2}$, $node_{3,3}$, …, $node_{3,M}$,
…,
Router1, N+1, $node_{N,1}$, $node_{N,2}$, $node_{N,3}$, …, $node_{N,M}$

Delay:

Delay1, 1, delay, units, maximum delay, is bypassed

The units value will be one of 100, 10000, 254, 3048, or –1000, representing centimeters, meters, inches, feet, or milliseconds, respectively. The positive numbers may be taken to mean the number of 100 microns in the unit, while the negative number expresses the number of microseconds.

Level (not ganged):

Control1, 1, N (channel count), 0 (is ganged)
Control1, 2, $level_1$, $mute_1$, "ID_1", $level_2$, $mute_2$, "ID_2", $level_3$, $mute_3$, "ID_3", …, $level_N$, $mute_N$, "ID_N"

Level (ganged):

Control1, 1, channel count, 1 (is ganged)
Control1, 2, level, mute, "ID"

Level Inc/Dec (not ganged):

Control1, 1, N (channel count), 0 (is ganged), has ramping,
Control1, 2, $level_1$, $mute_1$, min_1, max_1, $step_1$, ramp $rate_1$,, "ID_1"
$level_2$, $mute_2$, min_2, max_2, $step_2$, ramp $rate_2$, "ID_2",
$level_3$, $mute_3$, min_3, max_3, $step_3$, ramp $rate_3$, "ID_3", …,
$level_N$, $mute_N$, min_N, max_N, $step_N$, ramp $rate_N$, "ID_N"

Ramp rate will be zero if has ramping is disabled.

Level Inc/Dec (ganged):

Control1, 1, channel count, 1 (is ganged), has ramping,
Control1, 2, level, mute, min, max, step, ramp rate, "ID"

Ramp rate will be zero if has ramping is disabled.

Invert (not ganged):

Control1, 1, N (channel count), 0 (is ganged)
Control1, 2, invert$_1$, "ID$_1$", invert$_2$, "ID$_2$", invert$_3$, "ID$_3$", …, invert$_N$, "ID$_N$"

Invert (ganged):

Control1, 1, channel count, 1 (is ganged)
Control1, 2, invert, "ID"

Mute (not ganged):

Control1, 1, N (channel count), 0 (is ganged)
Control1, 2, mute$_1$, "ID$_1$", mute$_2$, "ID$_2$", mute$_3$, "ID$_3$", …, mute$_N$, "ID$_N$"

Mute (ganged):

Control1, 1, channel count, 1 (is ganged)
Control1, 2, mute, "ID"

Preset Button:

Preset Button1, 1, N (channel count)
Preset Button1, 2, preset ID$_1$, preset ID$_2$, preset ID$_3$, …, preset ID$_N$

Remote Preset Button:

Control1, 1, N (channel count)
Control1, 2, preset ID$_1$, preset ID$_2$, preset ID$_3$, …, preset ID$_N$

Logic Gate (not a flip flop or logic state):

Control1, 1, "logic gate type", 0

Logic Gate (Flip Flop):

Control1, 1, "Flip Flop", N (control channel count)
Control1, 2, state$_1$, "ID$_1$", state$_2$, "ID$_2$", state$_3$,"ID$_3$", …, state$_N$, "ID$_N$"

Logic Gate (Logic State):

Control1, 1, "Logic State", N (control channel count)
Control1, 2, state$_1$, "ID$_1$", state$_2$, "ID$_2$", state$_3$,"ID$_3$", …, state$_N$, "ID$_N$"

Logic Delay:

Control1, 1, N (channel count)
Control1, 2, on delay$_1$, off delay$_1$, is bypassed$_1$, on delay$_2$, off delay$_2$, is bypassed$_2$, on delay$_3$, off delay$_3$, is bypassed$_3$, …, on delay$_N$, off delay$_N$, is bypassed$_N$

Command String

Control1, 1, N (command count)
Control1, 2, "command string$_1$", "label$_1$", "command string$_2$", "label$_2$", "command string$_3$"," label$_3$", …, "command string$_N$", "label$_N$"

Volume 8:

Control1, 1, N (channel count)
Control1, 2, "object code$_1$", instance ID$_1$, "control ID$_1$", "ID$_1$", "object code$_2$", instance ID$_2$, "control ID$_2$", "ID$_2$", "object code$_3$", instance ID$_3$, "control ID$_3$", "ID$_3$", …, "object code$_4$", instance ID$_4$, "control ID$_4$", "ID$_4$"

Select 8:

No DSP data beyond line 0 is included. The base object code is Control.

Volume/Select 8:

Control1, 1, N (channel count)
Control1, 2, "object code$_1$", instance ID$_1$, "control ID$_1$", "ID$_1$", "object code$_2$", instance ID$_2$, "control ID$_2$", "ID$_2$", "object code$_3$", instance ID$_3$, "control ID$_3$", "ID$_3$", …, "object code$_4$", instance ID$_4$, "control ID$_4$", "ID$_4$"

Logic Box:

No DSP data beyond line 0 is included. The base object code is Control.

Signal Present Meter:

Meter1, 1, N (channel count)
Meter1, 2, threshold$_1$, "ID$_1$", threshold$_2$, "ID$_2$", threshold$_3$, "ID$_3$", …, threshold$_N$, "ID$_N$"

Peak Meter:

Meter1, 1, N (channel count)
Meter1, 2, hold time$_1$, is held$_1$, is indefinite$_1$, "ID$_1$", hold time$_2$, is held$_2$, is indefinite$_2$, "ID$_2$", hold time$_3$, is held$_3$, is indefinite$_3$, "ID$_3$", …, hold time$_N$, is held$_N$, is indefinite$_N$, "ID$_N$"

RMS Meter:

Meter1, 1, N (channel count)
Meter1, 2, hold $time_1$, is $held_1$, is $indefinite_1$, "ID_1",
hold $time_2$, is $held_2$, is $indefinite_2$, "ID_2", hold $time_3$, is $held_3$, is $indefinite_3$, "ID_3", …,
hold $time_N$, is $held_N$, is $indefinite_N$, "ID_N"

Tone Generator:

Generator1, 1, frequency, mute, level, start frequency, stop frequency,
frequency interval, interval type, time interval

Interval type will be 0 if the frequency interval is in octaves, and 1 if it is in hertz.

Pink Noise Generator:

Generator1, 1, mute, level

White Noise Generator:

Generator1, 1, mute, level

Transfer Function:

No DSP data beyond line 0 is included. The base object code is Diagnostic.

Pass-Through:

PassThrough1, 1, "type of pass-through"

The type of pass-through will be a 3-letter combination. The first letter indicates whether it is an audio or logic pass-through; A indicates audio, and L logic. The second and third letters indicate where the input comes in and where the output goes out; these may be L, R, T, or B, for left, right, top, or bottom, respectively.

Split Pass-Through Input:

SplitPassThrough1, 1, "type of pass-through"

The type of pass-through will be a 3-letter combination, either ASL or LST. The first, ASL, indicates a split audio input pass-through with the inputs on the left, the second, LST, indicates a split logic input pass-through with the inputs on top.

Split Pass-Through Output:

SplitPassThrough1, 1, “type of pass-through”

The type of pass-through will be a 3-letter combination, either ASR or LSB. The first, ASR, indicates a split audio output pass-through with the outputs on the right, the second, LSB, indicates a split logic output pass-through with the outputs on the bottom.

---

* Nexia is a trademark of Biamp Systems.