x8310 - equalizer settings for audio devices

Version 2

AudioEqualizerSettings

This feature describes configuration of equalizer properties for an audio device

[0] **getEqInfo**() → count
[1] **getFrequencies**(frequencyBandIndex) → frequencies
[2] **getFrequencyGains**(location) → gain(i)Value, gain(i+1)Value, ..
[3] **setFrequencyGains**(persistence, gain0Value, gain1Value, ..) → gain0Value, gain1Value, ..
[4] **getMicNoiseReduction**() → enabled
[5] **setMicNoiseReduction**(enabled)

Overview

Equalizer tables are used to customize the gain at different frequencies. This feature allows modification of that table. Only a single EQ table is currently supported.

Functions and Events

[0] getEqInfo() → count

Returns information related to the eq table.

Parameters

none

Returns

Table 1. getEqInfo() response packet format

byte \ bit	7	6	5	4	3	2	1	0
0	bandCount							
1	dbRange							
2	capabilities							
3	dbMin							
4	dbMax							

byte \ bit	7	6	5	4	3	2	1	0
5..15	reserved							

bandCount

Number of frequency bands.

dbRange

Range of dB. If dbMin and dbMax are both zero, the db min and max are calcuated as -dbRange and dbRange respectively.

capabilities

Combination of the following:

```
0x00 = default
0x01 = EQ values are stored as gains
0x02 = EQ values are stored as coefficients
```

dbMin

Minimum gain as a signed 8-bit value. If zero, a min of -dBRange is implied.

dbMax

Maximum gain as a signed 8-bit value. If zero, a max of +dBRange is implied.

Errors

None

[1] getFrequencies(frequencyBandIndex) → frequencies

Retrieves as many frequencies (in Hz) from the specified index as supported by the device. Frequencies are returned as unsigned 16-bit values in big-endian format. Using a HID++ long format, up to 7 frequencies can be returned at a time.

- Example: Device has 10 frequencies and using HID++ long reports (16 bytes payload). 2 calls are needed:
  - getFrequencies(0, 6) → retrieves first 6 bands
  - getFrequencies(7, 3) → retrieves last 3 bands

Parameters

Table 2. getFrequencies request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	frequencyBandIndex							

byte \ bit	7	6	5	4	3	2	1	0
1..15	reserved							

frequencyBandIndex

Value from 0 to bandCount-1

Returns

Table 3. getFrequencies response packet format

byte \ bit	7	6	5	4	3	2	1	0
0	frequencyBandIndex							
1	frequencyMSB (at frequencyBandIndex)							
2	frequencyLSB (at frequencyBandIndex)							
3	frequencyMSB (at frequencyBandIndex+1)							
4	frequencyLSB (at frequencyBandIndex+1)							
5	…							

frequencyBandIndex

frequency index passed from request

frequencyMSB

MSB of frequency (Hz) returned for that index

frequencyLSB

LSB of frequency (Hz) returned for that index

Errors

None

[2] getFrequencyGains(location) → gain(i)Value, gain(i+1)Value, ..

Gets the active EQ gain values (in dB) for each frequency. Frequency gains are stored as 1 byte signed values.

It is the responsibility of the device to ensure that the HID++ payload is large enough to receive each gain.

For HID++ long reports, up to 15 bands can be retrieved at a time.

Parameters

Table 4. getFrequencyGains request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	location							
1..15	reserved							

location

Determines where the gains are retrieved from.

```
0 = EEPROM (custom EQ)
1 = RAM (active EQ)
```

NOTE To maintain compatibility with the version 0 of this feature, the default of '0' retrieves gains from EEPROM.

Returns

Table 5. getFrequencyGains response packet format

byte \ bit	7	6	5	4	3	2	1	0
1	gainValue (band index 0)							
2	gainValue (band index 1)							
3	gainValue (band index 2)							
..	..							
bandCount-1	gainValue (band index bandCount-1)							

gainValue

Value in dB (signed 8-bit) of band at index N into the main EQ table

Errors

INVALID_ARGUMENT (2)

frequencyBandIndex, or gain value out of range

[3] setFrequencyGains(persistence, gain0Value, gain1Value, ..) → gain0Value, gain1Value, ..

Sets EQ gain values.

It is the responsibility of the device to ensure that the HID++ payload is large enough to send each gain.

For HID++ long reports, up to 15 bands can be sent at a time.

Parameters

Table 6. setFrequencyGains request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	persistence							
1	gainValue (band index 0)							
2	gainValue (band index 1)							
3	gainValue (band index 2)							
..	..							
bandCount-1	gainValue (band index bandCount-1)							

persistence

Determines how the frequency gains are persisted through a power cycle

```
0 = volatile (RAM)
1 = volatile and non-volatile (RAM and EEPROM)
2 = non-volatile only (EEPROM)
```

gainValue

Value in dB (signed 8-bit) of band at index N into the main EQ table

Returns

The setFrequencyGains response packet echoes the data from the request packet

Errors

INVALID_ARGUMENT (2)

gain value out of range

[4] getMicNoiseReduction() → enabled

Returns whether the hardware noise reduction is currently enabled

Parameters

none

Returns

Table 7. getMicNoiseReduction() response packet format

byte \ bit	7	6	5	4	3	2	1	0
0	enabled							

byte \ bit	7	6	5	4	3	2	1	0
1..15	reserved							

enabled

0 = disabled, 1 = enabled

Errors

None

[5] setMicNoiseReduction(enabled)

Sets the current hardware noise reduction

Parameters

Table 8. setFrequencyGains request packet format

byte \ bit	7	6	5	4	3	2	1	0
0	enabled							
1..15	reserved							

enabled

0 = disabled, 1 = enabled

Returns

none

Errors

None

Examples

- Assumptions:
  - Feature is on index 0x01 (will vary for device)
  - Software id = 0x0c
  - Device has 10 bands: { 32, 64, 125, 250, 500, 1000, 2000, 4000, 8000, 16000 } Hz
    - with gains: { 0, -12, 12, 0, 0, 0, 0, 0, 0, 0 } dB
  - Range of -12dB to 12dB

Table 9. Example Control ID table

Action	Request	Response	Comments

Get eq info	11 FF 01 0c	11 FF 00 0c **0a 0c**	bands=**10**
			dbRange=**0x0c**

Get the first 7 frequencies	11 FF 01 1c **00**	11 FF 01 1c **00 00 20 00 40 00 7D 00 FA 01 F4 03 E8 07 D0**	starting index=**0x00** **0x0002** = 32 **0x0040** = 64 **0x007D** = 125 **0x00FA** = 250 **0x01F4** = 500 **0x03E8** = 1000 **0x07D0** = 2000 (values in Hz)
Get the last 3 frequencies	11 FF 01 1c **07**	11 FF 01 1c **07 0F A0 1F 40 3E 80**	starting index=**0x07** **0x0FA0** = 4000 **0x1F40** = 8000 **0x3E80** = 16000 (values in Hz)
Get the frequency gains	11 FF 01 2c	11 FF 01 2c **00 F4 0C 00 00 00 00 00 00 00**	[0] → **0x00** (0) dB [1] → **0xF4** (-12) dB [2] → **0x0C** (+12) dB [3] → **0x00** (0) dB [4] → **0x00** (0) dB [5] → **0x00** (0) dB [6] → **0x00** (0) dB [7] → **0x00** (0) dB [8] → **0x00** (0) dB [9] → **0x00** (0) dB (values in Hz)
Adjust 500Hz to 4db, and 125Hz to -4dB	11 FF 01 3c **00 00 FC 00 0x04 00 00 00 00 00**	Same as request	[0] → **0x00** (0) dB [1] → **0x00** (0) dB [2] → **0xFC** (-4) dB [3] → **0x00** (0) dB [4] → **0x04** (4) dB [5] → **0x00** (0) dB [6] → **0x00** (0) dB [7] → **0x00** (0) dB [8] → **0x00** (0) dB [9] → **0x00** (0) dB (values in Hz)

ChangeLog

- Version 2: Added capabilities field for gains/coefficients, dbMin, dbMax to 'getInfo' and created new functions set/getMicNoiseReduction.

- Version 1: Added 'location' parameter in 'getFrequencyGains' to retrieve values from EEPROM or RAM
- Version 0: Initial version