ASHLY

Open API For FX Series DSP Amplifiers for Installers

September 2024

Table of Contents

This document describes the Line based API in the Pascal series of amplifiers.

1 Revision History

Firmware/Revision	Date	Changed By
1.6	21/11-2023	MAM
1.5	27/6-2023	MAM
1.4	16/1-2023	MAM
1.3	5/9-2022	MAM
1.2	9/3-2022	MAM
1.1	16/12-2021	MAM
1.0	11/11-2021	MAM

1.1 Firmware 1.6 Changes

- Updated: Input Channels updated to 8 channels See {IID} Input Channels (see page 20)

1.2 Firmware 1.5 Changes

- Added: Dante Info Registers (`SYSTEM.DANTE.*`)
- Updated: Input Channels updated with Dante. See {IID} Input Channels (see page 20)
- Updated: Input Sources updated with Dante. See {SID} Input Source (see page 21)
- Updated: Added Mix to Output Route Sources. See {RSID} Route Source (see page 22)

1.2.1 Registers Added

Register Name
`SYSTEM.DANTE.SOFTWARE_VERSION`
`SYSTEM.DANTE.FIRMWARE_VERSION`
`SYSTEM.DANTE.IP`
`SYSTEM.DANTE.MAC`
`SYSTEM.DANTE.LINK_SPEED`
`SYSTEM.DANTE.AES67_ENABLED`
`SYSTEM.DANTE.DEVICE_NAME`
`SYSTEM.DANTE.ENCODING`
`SYSTEM.DANTE.SAMPLE_RATE`
`SYSTEM.DANTE.CLOCK_STATE`
`SYSTEM.DANTE.MUTE_STATE`

1.2.2 Registers Modified

- Added Mixes to Route Sources
- Added PowerMode: `AUDIO_DSP` to register `SETUP.POWER.POWER_ON`

1.3 Firmware 1.4 Changes

- Added: Input HPF
- Added: 5-Band Input EQ
- Added: Mixes as Zone Primary Src.
- Added: Zone Priority Src for Zone
- Added: Option to disable Mute to Zone
- Added: Zone Ducker
- Added: Option to limit zone sources (Wall Controller Specific)
- Added: Option to select output SPDIF source
- Added: Bandwidth limitation for Pink Noise Generator
- Added: Sine Generator
- Removed: Generator cannot be disabled

1.3.1 Registers Added

Register Name
IN.EQ.COUNT (see page 49)
IN-{IID}.HPF_ENABLE (see page 47)
IN-{IID}.EQ.BYPASS (see page 49)
IN-{IID}.EQ-{EID}.TYPE (see page 50)
IN-{IID}.EQ-{EID}.GAIN (see page 51)
IN-{IID}.EQ-{EID}.FREQ (see page 51)
IN-{IID}.EQ-{EID}.Q (see page 52)
IN-{IID}.EQ-{EID}.BYPASS (see page 52)
ZONE-{ZID}.PRIORITY_SRC (see page 55)
ZONE-{ZID}.MUTE_ENABLE (see page 59)

Register Name
ZONE-{ZID}.SRC-{SID}.ENABLED (see page 59)
ZONE-{ZID}.DUCK.MODE (see page 61)
ZONE-{ZID}.DUCK.AUTO (see page 62)
ZONE-{ZID}.DUCK.THRESHOLD (see page 62)
ZONE-{ZID}.DUCK.ATTACK (see page 63)
ZONE-{ZID}.DUCK.RELEASE (see page 64)
ZONE-{ZID}.DUCK.HOLD (see page 64)
ZONE-{ZID}.DUCK.OVERRIDE_GAIN (see page 65)
ZONE-{ZID}.DUCK.OVERRIDE_GAIN_ENABLE (see page 65)
GENERATOR.TYPE (see page 79)
GENERATOR.SINE.FREQ (see page 79)
GENERATOR.PINK.LPF_ENABLE (see page 80)
GENERATOR.PINK.LPF_FREQ (see page 80)
GENERATOR.PINK.HPF_ENABLE (see page 81)
GENERATOR.PINK.HPF_FREQ (see page 81)
`MIX.COUNT`
`MIX-{MID}.NAME`
`MIX-{MID}.GAIN-{IID}`

Register Name
`ROUT-{RID}.SRC`
`ROUT-{RID}.SRC_CHANNEL`
`ROUT-{RID}.GAIN`

1.3.2 Registers Removed

Register Name
`GENERATOR.ENABLE`

1.4 Firmware 1.3 Changes

- Added: Output Gain
- Added: Clip Limiter Mode
- Added: Security Registers for WebPage Security
- Added: Input Gain Min + Input Gain Max
- Added: Analog Volume Control Value register as Value
- Remove: Analog Volume Control Volume register
- Update: Input Gain - Range increase from [-10, 10] to [-15, 15] dB
- Update: Zone Gain - when using Analog Volume Control
- Update: SETUP.LAN and SETUP.WIFI registers as readonly

1.4.1 Registers Added

Register Name
ZONE-{ZID}.GAIN_MIN (see page 56)
ZONE-{ZID}.GAIN_MAX (see page 57)
OUT-{OID}.GAIN (see page 75)

Register Name
`OUT-{OID}.CLIP_LIMITER.MODE`
`VC-{VID}.VALUE`
`SYSTEM.SECURITY.PASSWORD_ENABLE`
`SYSTEM.SECURITY.PASSWORD_HASH`

1.4.2 Registers Updated

Register Name	Change
IN-{IID}.GAIN (see page 46)	Limits
ZONE-{ZID}.GAIN (see page 56)	Limits, more
`SETUP.LAN.NETWORK_MODE`	Read Only
`SETUP.LAN.IP`	Read Only
`SETUP.LAN.MASK`	Read Only
`SETUP.LAN.GATEWAY`	Read Only
`SETUP.LAN.DNS1`	Read Only
`SETUP.LAN.DNS2`	Read Only
`SETUP.WIFI.ENABLE`	Read Only
`SETUP.WIFI.DISABLE_LAN_CONNECTED`	Read Only

Register Name	Change
`SETUP.WIFI.DISABLE_AFTER`	Read Only
`SETUP.WIFI.MODE`	Read Only
`SETUP.WIFI.AP_SSID`	Read Only
`SETUP.WIFI.AP_PASS`	Read Only
`SETUP.WIFI.STA_SSID`	Read Only
`SETUP.WIFI.STA_PASS`	Read Only

1.4.3 Registers Removed

Register Name
`VC-{VID}.VOLUME`

1.5 Firmware 1.2 Changes

- `INC` Command support for Input Gain
- Added Frequency parameter for `SUBSCRIBE` command
- Pink NoiseGenerator

1.5.1 Registers Added

Register Name
`SETUP.DEVICE.SERIAL`

Register Name
`SETUP.DEVICE.FIRMWARE`
`SETUP.DEVICE.FIRMWARE`
`SETUP.DEVICE.MAC`
`SETUP.DEVICE.WIFI_MAC`
`OUT-{OID}.LIMITER.AUTO`
`OUT-{OID}.LIMITER.THRESHOLD`
`OUT-{OID}.LIMITER.ATTACK`
`OUT-{OID}.LIMITER.RELEASE`
`OUT-{OID}.LIMITER.HOLD`

1.6 Firmware 1.1 Changes

1.6.1 Registers Added

Register Name
ZONE-{ZID}.COMPRESSOR.HOLD (see page 69)
`OUT-{OID}.PRESET.NAME`
`OUT-{OID}.PRESET.ID`

Register Name
`OUT-{OID}.PRESET.LOCKED`
`OUT-{OID}.POLARITY.PROTECTED`
`OUT-{OID}.OUTPUT_MODE.PROTECTED`
`OUT-{OID}.SPEAKER_DELAY.PROTECTED`
`OUT-{OID}.LIMITER.PROTECTED`
`OUT-{OID}.SPEAKER_EQ.PROTECTED`
`OUT-{OID}.XR.PROTECTED`
`OUT-{OID}.FIR.PROTECTED`
`OUT-{OID}.PEAK_LIMITER.BYPASS`
`OUT-{OID}.PEAK_LIMITER.KNEE`
`OUT-{OID}.RMS_LIMITER.BYPASS`
`OUT-{OID}.RMS_LIMITER.THRESHOLD`
`OUT-{OID}.RMS_LIMITER.ATTACK`
`OUT-{OID}.RMS_LIMITER.RELEASE`
`OUT-{OID}.RMS_LIMITER.HOLD`
`OUT-{OID}.RMS_LIMITER.KNEE`

Register Name
`OUT-{OID}.CLIP_LIMITER.BYPASS`
`OUT-{OID}.FIR.BYPASS`
`OUT-{OID}.FIR.TAPS`

2 Getting Started

2.1 Connecting to the Amplifier

Out of the Box the amplifier is hard-coded with the Ethernet Address 192.168.64.100.
It is also possible to connect to the amplifier using Wifi. Connect to the Wifi AP (SSID)
and connect using the default IP address of 192.168.4.1.

2.2 Discovery

If the application requires the amplifier to have a dynamic IP address, it is possible to
use mDNS to locate the amplifier.

The service type is: `_pasconnect._tcp`

The following properties is defined:

- **api_version** - the api version of the device
- **device_type** - the device type. For amplifiers this will always be `PasAmpControl`
- **model** - the model name of the device
- **software_id** - software id of the amplifier (Manufacturer and Model Specific)
- **hardware_id** - hardware id of the amplifier (Model ID)

Example (Avahi for Linux):

```
$> avahi-browse -t -r _pasconnect._tcp
+ enp0s8 IPv4 {{ api_mdns_hostname }}                        _pasconnect._tcp
local
= enp0s8 IPv4 {{ api_mdns_hostname }}                        _pasconnect._tcp
local
   hostname = [{{ api_mdns_hostname }}.local]
   address = [192.168.64.100]
   port = [80]
   txt = {{ api_mdns_txt }}
```

3 Definitions

3.1 Variable Types

- **Float** - Float format, delimited with '.'
- **Integer** - Normal integer
- **Enum** - Basically a string with a predefined set of options
- **String** - String - might have limitations on number of characters. String values containing spaces must be enclosed in double-quotes.

3.2 {IID} Input Channels

The following input channels is defined for the amplifier.

- **100** - Analog Input 1
- **101** - Analog Input 2
- **102** - Analog Input 3
- **103** - Analog Input 4
- **104** - Analog Input 5 *(8 channel version only)*
- **105** - Analog Input 6 *(8 channel version only)*
- **106** - Analog Input 7 *(8 channel version only)*
- **107** - Analog Input 8 *(8 channel version only)*
- **200** - SPDIF 1 (Left)
- **201** - SPDIF 1 (Right)
- **300** - Dante 1 *(Only for Dante Enabled Amplifiers)*
- **301** - Dante 2 *(Only for Dante Enabled Amplifiers)*
- **302** - Dante 3 *(Only for Dante Enabled Amplifiers)*
- **303** - Dante 4 *(Only for Dante Enabled Amplifiers)*
- **400** - Noise Generator

3.3 {MID} Mix Channels

The following mix channels is defined for the amplifier.

- **1** - Mix 1
- **2** - Mix 2
- **3** - Mix 3 *(4 and 8 channel versions only)*
- **4** - Mix 4 *(4 and 8 channel versions only)*
- **5** - Mix 5 *(8 channel version only)*
- **6** - Mix 6 *(8 channel version only)*

- **7** - Mix 7 *(8 channel version only)*
- **8** - Mix 8 *(8 channel version only)*

3.4 {ZID} Zones

The following zones is defined for the amplifier.

- **A** - Zone A
- **B** - Zone B
- **C** - Zone C *(4 and 8 channel versions only)*
- **D** - Zone D *(4 and 8 channel versions only)*
- **E** - Zone E *(8 channel version only)*
- **F** - Zone F *(8 channel version only)*
- **G** - Zone G *(8 channel version only)*
- **H** - Zone H *(8 channel version only)*

3.5 {OID} Output Channels

- **1** - Output 1
- **2** - Output 2
- **3** - Output 3 *(4 and 8 channel versions only)*
- **4** - Output 4 *(4 and 8 channel versions only)*
- **5** - Output 5 *(8 channel version only)*
- **6** - Output 6 *(8 channel version only)*
- **7** - Output 7 *(8 channel version only)*
- **8** - Output 8 *(8 channel version only)*

3.6 {RID} Output Route Channels

- **1** - Output 1
- **2** - Output 2

3.7 {SID} Input Source

The following input sources is defined for the amplifier.

- **0** - Unused Input (Silent)
- **100** - Analog Input 1
- **101** - Analog Input 2
- **102** - Analog Input 3
- **103** - Analog Input 4
- **104** - Analog Input 5 *(8 channel version only)*

- **105** - Analog Input 6 *(8 channel version only)*
- **106** - Analog Input 7 *(8 channel version only)*
- **107** - Analog Input 8 *(8 channel version only)*
- **200** - SPDIF 1 (Left)
- **201** - SPDIF 1 (Right)
- **300** - Dante 1 *(Only for Dante Enabled Amplifiers)*
- **301** - Dante 2 *(Only for Dante Enabled Amplifiers)*
- **302** - Dante 3 *(Only for Dante Enabled Amplifiers)*
- **303** - Dante 4 *(Only for Dante Enabled Amplifiers)*
- **400** - Noise Generator
- **500** - Mix 1
- **501** - Mix 2
- **502** - Mix 3 *(4 and 8 channel versions only)*
- **503** - Mix 4 *(4 and 8 channel versions only)*
- **504** - Mix 4 *(8 channel version only)*
- **505** - Mix 4 *(8 channel version only)*
- **506** - Mix 4 *(8 channel version only)*
- **507** - Mix 4 *(8 channel version only)*

3.8 {RSID} Route Source

The following route sources is defined for the amplifier.

- **0** - Unused (Silent)
- **100** - Analog Input 1
- **101** - Analog Input 2
- **102** - Analog Input 3
- **103** - Analog Input 4
- **104** - Analog Input 5 *(8 channel version only)*
- **105** - Analog Input 6 *(8 channel version only)*
- **106** - Analog Input 7 *(8 channel version only)*
- **107** - Analog Input 8 *(8 channel version only)*
- **200** - SPDIF 1 (Left)
- **201** - SPDIF 1 (Right)
- **300** - Dante 1
- **301** - Dante 2
- **302** - Dante 3
- **303** - Dante 4
- **500** - Mix A
- **501** - Mix B
- **502** - Mix C
- **503** - Mix D
- **1000** - Zone A
- **1001** - Zone B
- **1002** - Zone C *(4 and 8 channel versions only)*

- **1003** - Zone D *(4 and 8 channel versions only)*
- **1004** - Zone E *(8 channel version only)*
- **1005** - Zone F *(8 channel version only)*
- **1006** - Zone G *(8 channel version only)*
- **1007** - Zone H *(8 channel version only)*

3.9 {VID} Volume Controls

The following Volume Controls is defined for the amplifier.

- **0** - OFF
- **1** - GPIO PIN 4 Volume Control
- **2** - GPIO PIN 5 Volume Control
- **3** - GPIO PIN 6 Volume Control
- **4** - GPIO PIN 7 Volume Control

3.10 {EID} Equalizer Bands

The following Equalizer Bands are defined in the amplifier

- **1** - Equalizer Band 1
- **2** - Equalizer Band 2
- **3** - Equalizer Band 3
- **4** - Equalizer Band 4
- **5** - Equalizer Band 5
- **6** - Equalizer Band 6 *(Output And Speaker EQ Only)*
- **7** - Equalizer Band 7 *(Output And Speaker EQ Only)*
- **8** - Equalizer Band 8 *(Output And Speaker EQ Only)*
- **9** - Equalizer Band 9 *(Output And Speaker EQ Only)*
- **10** - Equalizer Band 10 *(Output And Speaker EQ Only)*
- **11** - Equalizer Band 11 *(Speaker EQ Only)*
- **12** - Equalizer Band 12 *(Speaker EQ Only)*
- **13** - Equalizer Band 13 *(Speaker EQ Only)*
- **14** - Equalizer Band 14 *(Speaker EQ Only)*
- **15** - Equalizer Band 15 *(Speaker EQ Only)*

4 API Endpoints

4.1 Raw Socket API

The Primary API in the amplifier is based on a TCP Socket connection (**Port 7621**) and is **Line based**. That means every line is delimited by newline `\n`
Every line contains a single message. The API consists of 2 parts - a Command/Response interface and a Publish/Subscribe Interface.

4.2 WebSocket API

It is also possible to connect to the Websocket based API in the amplifier. The syntax of commands and replies is exactly the same between the Socket based API and the WebSocket based API - though a single websocket message might contain/return multiple lines of text - with each line containing a single message.

4.3 Examples

4.3.1 Ncat

Examples in the documentation are based on NCAT (https://nmap.org/download.html). The specific syntax is PowerShell, but it can easily be converted to Bash for Linux.

PowerShell style:

```
$> "POWER_ON" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*POWER_ON
```

Bash style:

```
$> echo "POWER_ON" | ncat 192.168.64.100 7621 --no-shutdown -i 1
*POWER_ON
```

4.3.2 Python - socketes example

```
import socket
```

```
TARGET = '192.168.64.100'
PORT = 7621

def get_all():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((TARGET, PORT))

        cmd = "GET *\n"
        s.sendall(cmd.encode())

        while True:
            reply = s.recv(64*1024)

            if reply:
                reply = reply.decode()
                print(reply)

            if not reply or f'*{cmd}' in reply:
                break


def subscribe_all():
    with socket.socket(socket.AF_INET, socket.SOCK_STREAM) as s:
        s.connect((TARGET, PORT))

        cmd = "SUBSCRIBE *\n"
        s.sendall(cmd.encode())

        for i in range(5):
            reply = s.recv(64*1024)

            if reply:
                reply = reply.decode()
                print(reply)


get_all()
subscribe_all()
```

4.3.3 Python - websockets example

```
# Requires Websockets
# to install: pip install websockets

from websockets.sync.client import connect

TARGET = '192.168.64.100'

def get_all():
    with connect(f"ws://{TARGET}/ws") as websocket:
        cmd = "GET *"
        websocket.send(cmd)

        reply = websocket.recv(timeout=0.5)
        print(reply)

        websocket.close_socket()

def subscribe_all():
    with connect(f"ws://{TARGET}/ws") as websocket:
        cmd = "SUBSCRIBE *"
        websocket.send(cmd)

        for i in range(5):
            reply = websocket.recv(timeout=0.5)
            print(reply)

        websocket.close_socket()



get_all()
subscribe_all()
```

5 Command/Response

The Command/Response interface allows for Querying/Updating the registers in the amplifier and to execute commands.

To execute a command - send a websocket message with the command followed by newline.

- If the command executes successfully the response will be an asterisk followed by the command text.

```
>> {{COMMAND}}
<< *{{COMMAND}}
```

- If the command fails the response will be an hash followed by an error description.

```
>> {{COMMAND}}
<< #{{Error Message}}
```

- If the command returns data in form of registers the response will be:

```
>> {{COMMAND}}
<< +{{RESPONSE}}
<< *{{COMMAND}}
```

5.1 Command Types

- GET (see page 27)
- SET (see page 28)
- INC (see page 28)
- SUBSCRIBE (see page 29)
- UNSUBSCRIBE (see page 30)
- POWER_ON (see page 30)
- POWER_OFF (see page 30)

5.1.1 GET

Get value of amplifier register. The command supports wildcards.

Format:

```
>> GET {{REGISTER}}
<< +{{RESPONSE(s)}}
<< *{{COMMAND}}
```

Example:

```
>> GET IN-100.NAME
<< +IN-100.NAME "Analog 1"
<< *GET IN-100.NAME
```

```
>> GET IN-*.NAME
<< +IN-100.NAME "Analog 1"
<< +IN-101.NAME "Analog 2"
<< +IN-102.NAME "Analog 3"
<< +IN-103.NAME "Analog 4"
<< +IN-200.NAME "S/PDIF 1"
<< +IN-201.NAME "S/PDIF 1R"
<< +IN-400.NAME "Noise Generator"
<< *GET IN-*.NAME
```

5.1.2 SET

Set value in amplifier register. The command does not support wildcards!

Format:

```
>> SET {{REGISTER}} {{VALUE}}
<< *{{COMMAND}}
```

Example

```
>> SET IN-100.NAME "Streamer"
<< +IN-100.NAME "Analog 1"
<< *SET IN-100.NAME "Streamer"
```

5.1.3 INC

Modifies the value in amplifier register by the amount specified in the command. The valus can be positive or negative.
The command does not support wildcards!

Format:

```
>> INC {{REGISTER}} {{VALUE}}
<< +{{REGISTER}} {{MODIFIED VALUE}}
<< *{{COMMAND}}
```

Example

```
>> INC ZONE-A.GAIN -5
<< +ZONE-A.GAIN -5.00
<< *INC ZONE-A.GAIN -5
```

5.1.4 SUBSCRIBE

Subscribe to changes in all registers and dynamics. The subscribe command does not support subscriptions to individual registers. This might change in a later release.

The register changes will stream to the websocket after subscription...

```
>> SUBSCRIBE
...
<< +IN-100.DYN.SIGNAL -49.9777
<< +IN-100.DYN.CLIP 0
<< +IN-101.DYN.SIGNAL -49.3077
<< +IN-101.DYN.CLIP 0
<< +IN-102.DYN.SIGNAL -99.7209
<< +IN-102.DYN.CLIP 0
...
<< *SUBSCRIBE
```

5.1.4.1 SUBSCRIBE <BLANK|*|REG|DYN> \<FREQ\>

REG: Register Updates Only

DYN: Dynamic updates Only

"*": All register updates - Equal to BLANK

<BLANK>: IF EMPTY - Both Dynamic and register updates

<FREQ>: Frequency of updates: 1=1 update per second, 0.5 equals 1 update every 5 seconds.

Subscribe to changes in all registers or dynamic updates. The subscribe command does not support subscriptions to individual registers. This might change in a later release.

The register changes will stream to the socket/websocket after subscription...

Example: Subscribe to All updates

```
>> SUBSCRIBE
...
<< +IN-100.DYN.SIGNAL -49.9777
<< +IN-100.DYN.CLIP 0
<< +IN-101.DYN.SIGNAL -49.3077
...
```

5.1.5 UNSUBSCRIBE

5.1.5.1 UNSUBSCRIBE <BLANK|*|REG|DYN>

REG - Register Updates Only

DYN - Dynamic updates Only

"BLANK" - IF EMPTY - Both dynamic and register updates

Unsubscribe to the previous subscription. The parameter must match a previous subscription. An unsubscribe all (Blank value) will not unsubscribe a subscription to register only updates.

```
>> UNSUBSCRIBE DYN
<< *UNSUBSCRIBE DYN
```

5.1.6 POWER_ON

TYPE: Command

Methods: POWER_ON

Example:

```
>> POWER_ON
<< *POWER_ON
```

5.1.7 POWER_OFF

TYPE: Command

Methods: POWER_OFF

Example:

```
>> POWER_OFF
<< *POWER_OFF
```

5.2 Registers

5.2.1 Base Registers

Supported Registers for General Use

Register Name	Type	Access	Note
API_VERSION (see page 31)	`String`	Get	
SYSTEM.STATUS.STATE (see page 32)	`Enum`	Get	{INIT, STANDBY, ON, FAULT}
SYSTEM.STATUS.SIGNAL_IN (see page 32)	`Enum`	Get	{OFF, NO_SIGNAL, SIGNAL, CLIP}
SYSTEM.STATUS.SIGNAL_OUT (see page 33)	`Enum`	Get	{OFF, NO_SIGNAL, SIGNAL, CLIP, FAULT}
SYSTEM.STATUS.LAN (see page 33)	`String`	Get	IP Address or Empty
SYSTEM.STATUS.WIFI (see page 33)	`String`	Get	

5.2.1.1 API_VERSION

TYPE: Register

METHODS: Get

VALUES: ENUM

INIT: Amplifier is initializing

STANDBY: Amplifier is in standby

ON: Amplifier is on

FAULT: Amplifier has Non recoverable Error

Example:

```
>> GET API_VERSION
<< +API_VERSION "1.6"
<< *GET API_VERSION
```

5.2.1.2 SYSTEM.STATUS.STATE

TYPE: Register

METHODS: Get

VALUES: ENUM

INIT: Amplifier is initializing

STANDBY: Amplifier is in standby

ON: Amplifier is on

FAULT: Amplifier has Non recoverable Error

Example:

```
>> GET SYSTEM.STATUS.STATE
<< +SYSTEM.STATUS.STATE "ON"
<< *GET SYSTEM.STATUS.STATE
```

5.2.1.3 SYSTEM.STATUS.SIGNAL_IN

TYPE: Register

METHODS: Get

VALUES: ENUM

OFF: Input(s) is Off

NO_SIGNAL: Input(s) has no signal (Below threshold)

SIGNAL: Input(s) has signal (Above threshold)

CLIP: Input(s) is clipping ADC - please decrease sensitivity

Example:

```
>> GET SYSTEM.STATUS.SIGNAL_IN
<< +SYSTEM.STATUS.SIGNAL_IN "SIGNAL"
<< *GET SYSTEM.STATUS.SIGNAL_IN
```

5.2.1.4 SYSTEM.STATUS.SIGNAL_OUT

TYPE: Register

METHODS: Get

VALUES: ENUM

> **OFF:** Output(s) is Off
>
> **NO_SIGNAL:** Output(s) has no signal (Below threshold)
>
> **SIGNAL:** Output(s) has signal (Above threshold)
>
> **CLIP:** Output(s) is clipping in amplifier - please decrease volume.
>
> **FAULT:** Output(s) has unspecified fault

Example:

```
>> GET SYSTEM.STATUS.SIGNAL_OUT
<< +SYSTEM.STATUS.SIGNAL_OUT "SIGNAL"
<< *GET SYSTEM.STATUS.SIGNAL_OUT
```

5.2.1.5 SYSTEM.STATUS.LAN

TYPE: Register

METHODS: Get

VALUES: STRING

> **IP Address:** LAN is connected and has received IP Address
>
> **EMPTY:** LAN is not connected or no IP Address received/configured

Example:

```
>> GET SYSTEM.STATUS.LAN
<< +SYSTEM.STATUS.LAN "192.168.64.100"
<< *GET SYSTEM.STATUS.LAN
```

5.2.1.6 SYSTEM.STATUS.WIFI

TYPE: Register

METHODS: Get

VALUES: STRING

> **IP Address:** WIFI is connected and has received IP Address
>
> **EMPTY:** WIFI is not connected or no IP Address received/configured

Example:

```
>> GET SYSTEM.STATUS.WIFI
<< +SYSTEM.STATUS.WIFI "192.168.4.1"
<< *GET SYSTEM.STATUS.WIFI
```

5.2.2 Device Information Registers

Register Name	Type	Access	Note
SYSTEM.DEVICE.SWID (see page 34)	`Integer`	Get	
SYSTEM.DEVICE.HWID (see page 35)	`Integer`	Get	
SYSTEM.DEVICE.VENDOR_NAME (see page 35)	`String`	Get	
SYSTEM.DEVICE.MODEL_NAME (see page 35)	`String`	Get	
SYSTEM.DEVICE.SERIAL (see page 36)	`String`	Get	
SYSTEM.DEVICE.FIRMWARE (see page 36)	`String`	Get	
SYSTEM.DEVICE.FIRMWARE_DATE (see page 36)	`String`	Get	
SYSTEM.DEVICE.MAC (see page 37)	`String`	Get	
SYSTEM.DEVICE.WIFI_MAC (see page 37)	`String`	Get	

5.2.2.1 SYSTEM.DEVICE.SWID

TYPE: Register

METHODS: Get

VALUES: INTEGER

Example:

```
>> GET SYSTEM.DEVICE.SWID
```

```
<< +SYSTEM.DEVICE.SWID {{ api_swid }}
<< *GET SYSTEM.DEVICE.SWID
```

5.2.2.2 SYSTEM.DEVICE.HWID

TYPE: Register

METHODS: Get

VALUES: INTEGER

Example:

```
>> GET SYSTEM.DEVICE.HWID
<< +SYSTEM.DEVICE.HWID 4
<< *GET SYSTEM.DEVICE.HWID
```

5.2.2.3 SYSTEM.DEVICE.VENDOR_NAME

TYPE: Register

METHODS: Get

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SYSTEM.DEVICE.VENDOR_NAME
<< +SYSTEM.DEVICE.VENDOR_NAME {{ api_vendor_name }}
<< *GET SYSTEM.DEVICE.VENDOR_NAME
```

5.2.2.4 SYSTEM.DEVICE.MODEL_NAME

TYPE: Register

METHODS: Get

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SYSTEM.DEVICE.MODEL_NAME
<< +SYSTEM.DEVICE.MODEL_NAME {{ api_model_name }}
<< *GET SYSTEM.DEVICE.MODEL_NAME
```

5.2.2.5 SYSTEM.DEVICE.SERIAL

TYPE: Register

METHODS: Get

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SYSTEM.DEVICE.SERIAL
<< +SYSTEM.DEVICE.SERIAL "2122023201X00031"
<< *GET SYSTEM.DEVICE.SERIAL
```

5.2.2.6 SYSTEM.DEVICE.FIRMWARE

TYPE: Register

METHODS: Get

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SYSTEM.DEVICE.FIRMWARE
<< +SYSTEM.DEVICE.FIRMWARE "1.0.0"
<< *GET SYSTEM.DEVICE.FIRMWARE
```

5.2.2.7 SYSTEM.DEVICE.FIRMWARE_DATE

TYPE: Register

METHODS: Get

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SYSTEM.DEVICE.FIRMWARE_DATE
<< +SYSTEM.DEVICE.FIRMWARE_DATE "Nov  5 2021 07:51:56"
<< *GET SYSTEM.DEVICE.FIRMWARE_DATE
```

5.2.2.8 SYSTEM.DEVICE.MAC

TYPE: Register

METHODS: Get

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SYSTEM.DEVICE.MAC
<< +SYSTEM.DEVICE.MAC "C4:5B:BE:31:42:F3"
<< *GET SYSTEM.DEVICE.MAC
```

5.2.2.9 SYSTEM.DEVICE.WIFI_MAC

TYPE: Register

METHODS: Get

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SYSTEM.DEVICE.WIFI_MAC
<< +SYSTEM.DEVICE.WIFI_MAC "C4:5B:BE:31:42:F0"
<< *GET SYSTEM.DEVICE.WIFI_MAC
```

5.2.3 System Information Registers

Register Name	Type	Access	Note
SETUP.SYSTEM.DEVICE_NAME (see page 38)	`String[32]`	Get, Set	
SETUP.SYSTEM.VENUE_NAME (see page 39)	`String[32]`	Get, Set	
SETUP.SYSTEM.CUSTOMER_NAME (see page 39)	`String[32]`	Get, Set	

Register Name	Type	Access	Note
SETUP.SYSTEM.ASSET_TAG (see page 40)	`String[32]`	Get, Set	
SETUP.SYSTEM.INSTALLER_NAME (see page 40)	`String[32]`	Get, Set	
SETUP.SYSTEM.CONTACT_INFO (see page 41)	`String[32]`	Get, Set	
SETUP.SYSTEM.INSTALL_DATE (see page 41)	`String[32]`	Get, Set	
SETUP.SYSTEM.INSTALL_NOTES (see page 41)	`String[512]`	Get, Set	
SETUP.SYSTEM.LOCATING (see page 42)	`Boolean`	Get, Set	
SETUP.SYSTEM.CUSTOM1 (see page 42)	`String[8192]`	Get, Set	
SETUP.SYSTEM.CUSTOM2 (see page 43)	`String[8192]`	Get, Set	
SETUP.SYSTEM.CUSTOM3 (see page 43)	`String[8192]`	Get, Set	

5.2.3.1 SETUP.SYSTEM.DEVICE_NAME

TYPE: Register

METHODS: Get, Set

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SETUP.SYSTEM.DEVICE_NAME
```

```
<< +SETUP.SYSTEM.DEVICE_NAME {{ api_device_name }}
<< *GET SETUP.SYSTEM.DEVICE_NAME

>  SET SETUP.SYSTEM.DEVICE_NAME "MyFXamp"
<< *SET SETUP.SYSTEM.DEVICE_NAME {{ api_device_name_new }}

>  GET SETUP.SYSTEM.DEVICE_NAME
<< +SETUP.SYSTEM.DEVICE_NAME {{ api_device_name_new }}
<< *GET SETUP.SYSTEM.DEVICE_NAME
```

5.2.3.2 SETUP.SYSTEM.VENUE_NAME

TYPE: Register

METHODS: Get, Set

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SETUP.SYSTEM.VENUE_NAME
<< +SETUP.SYSTEM.VENUE_NAME ""
<< *GET SETUP.SYSTEM.VENUE_NAME

>> SET SETUP.SYSTEM.VENUE_NAME "THouse"
<< *SET SETUP.SYSTEM.VENUE_NAME "THouse"

>> GET SETUP.SYSTEM.VENUE_NAME
<< +SETUP.SYSTEM.VENUE_NAME "THouse"
<< *GET SETUP.SYSTEM.VENUE_NAME
```

5.2.3.3 SETUP.SYSTEM.CUSTOMER_NAME

TYPE: Register

METHODS: Get, Set

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SETUP.SYSTEM.CUSTOMER_NAME
<< +SETUP.SYSTEM.CUSTOMER_NAME ""
<< *GET SETUP.SYSTEM.CUSTOMER_NAME

>> SET SETUP.SYSTEM.CUSTOMER_NAME "R. Rock"
<< *SET SETUP.SYSTEM.CUSTOMER_NAME "R. Rock"
```

```
>> GET SETUP.SYSTEM.CUSTOMER_NAME
<< +SETUP.SYSTEM.CUSTOMER_NAME "R. Rock"
<< *GET SETUP.SYSTEM.CUSTOMER_NAME
```

5.2.3.4 SETUP.SYSTEM.ASSET_TAG

TYPE: Register

METHODS: Get, Set

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SETUP.SYSTEM.ASSET_TAG
<< +SETUP.SYSTEM.ASSET_TAG ""
<< *GET SETUP.SYSTEM.ASSET_TAG

>> SET SETUP.SYSTEM.ASSET_TAG "XZ233WV"
<< *SET SETUP.SYSTEM.ASSET_TAG "XZ233WV"

>> GET SETUP.SYSTEM.ASSET_TAG
<< +SETUP.SYSTEM.ASSET_TAG "XZ233WV"
<< *GET SETUP.SYSTEM.ASSET_TAG
```

5.2.3.5 SETUP.SYSTEM.INSTALLER_NAME

TYPE: Register

METHODS: Get, Set

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SETUP.SYSTEM.INSTALLER_NAME
<< +SETUP.SYSTEM.INSTALLER_NAME ""
<< *GET SETUP.SYSTEM.INSTALLER_NAME

>> SET SETUP.SYSTEM.INSTALLER_NAME "AV.X"
<< *SET SETUP.SYSTEM.INSTALLER_NAME "AV.X"

>> GET SETUP.SYSTEM.INSTALLER_NAME
<< +SETUP.SYSTEM.INSTALLER_NAME "AV.X"
<< *GET SETUP.SYSTEM.INSTALLER_NAME
```

5.2.3.6 SETUP.SYSTEM.CONTACT_INFO

TYPE: Register

METHODS: Get, Set

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET SETUP.SYSTEM.CONTACT_INFO
<< +SETUP.SYSTEM.CONTACT_INFO ""
<< *GET SETUP.SYSTEM.CONTACT_INFO

>> SET SETUP.SYSTEM.CONTACT_INFO "555-9753"
<< *SET SETUP.SYSTEM.CONTACT_INFO "555-9753"

>> GET SETUP.SYSTEM.CONTACT_INFO
<< +SETUP.SYSTEM.CONTACT_INFO "555-9753"
<< *GET SETUP.SYSTEM.CONTACT_INFO
```

5.2.3.7 SETUP.SYSTEM.INSTALL_DATE

TYPE: Register

METHODS: Get, Set

VALUES: STRING

Max Length 64 chars

Example:

```
>> GET SETUP.SYSTEM.INSTALL_DATE
<< +SETUP.SYSTEM.INSTALL_DATE ""
<< *GET SETUP.SYSTEM.INSTALL_DATE

>> SET SETUP.SYSTEM.INSTALL_DATE "01-01-2021"
<< *SET SETUP.SYSTEM.INSTALL_DATE "01-01-2021"

>> GET SETUP.SYSTEM.INSTALL_DATE
<< +SETUP.SYSTEM.INSTALL_DATE "01-01-2021"
<< *GET SETUP.SYSTEM.INSTALL_DATE
```

5.2.3.8 SETUP.SYSTEM.INSTALL_NOTES

TYPE: Register

METHODS: Get, Set

VALUES: STRING

Max Length 256 chars

Example:

```
>> GET SETUP.SYSTEM.INSTALL_NOTES
<< +SETUP.SYSTEM.INSTALL_NOTES ""
<< *GET SETUP.SYSTEM.INSTALL_NOTES

>> SET SETUP.SYSTEM.INSTALL_NOTES "Nice"
<< *SET SETUP.SYSTEM.INSTALL_NOTES "Nice"

>> GET SETUP.SYSTEM.INSTALL_NOTES
<< +SETUP.SYSTEM.INSTALL_NOTES "Nice"
<< *GET SETUP.SYSTEM.INSTALL_NOTES
```

5.2.3.9 SETUP.SYSTEM.LOCATING

TYPE: Register

METHODS: Get, Set

VALUES: BOOLEAN

Example:

```
>> GET SETUP.SYSTEM.LOCATING
<< +SETUP.SYSTEM.LOCATING 0
<< *GET SETUP.SYSTEM.LOCATING

>> SET SETUP.SYSTEM.LOCATING 1
<< *SET SETUP.SYSTEM.LOCATING 1

>> GET SETUP.SYSTEM.LOCATING
<< +SETUP.SYSTEM.LOCATING 1
<< *GET SETUP.SYSTEM.LOCATING
```

5.2.3.10 SETUP.SYSTEM.CUSTOM1

TYPE: Register

METHODS: Get, Set

VALUES: STRING

Max Length 8192 chars

Example:

```
>> GET SETUP.SYSTEM.CUSTOM1
<< +SETUP.SYSTEM.CUSTOM1 ""
<< *GET SETUP.SYSTEM.LOCATING

>> SET SETUP.SYSTEM.CUSTOM1 "Custom"
<< *SET SETUP.SYSTEM.CUSTOM1 "Custom"

>> GET SETUP.SYSTEM.CUSTOM1
<< +SETUP.SYSTEM.CUSTOM1 "Custom"
<< *GET SETUP.SYSTEM.CUSTOM1
```

5.2.3.11 SETUP.SYSTEM.CUSTOM2

TYPE: Register

METHODS: Get, Set

VALUES: STRING

Max Length 8192 chars

Example:

```
>> GET SETUP.SYSTEM.CUSTOM2
<< +SETUP.SYSTEM.CUSTOM2 ""
<< *GET SETUP.SYSTEM.LOCATING

>> SET SETUP.SYSTEM.CUSTOM2 "Custom"
<< *SET SETUP.SYSTEM.CUSTOM2 "Custom"

>> GET SETUP.SYSTEM.CUSTOM2
<< +SETUP.SYSTEM.CUSTOM2 "Custom"
<< *GET SETUP.SYSTEM.CUSTOM2
```

5.2.3.12 SETUP.SYSTEM.CUSTOM3

TYPE: Register

METHODS: Get, Set

VALUES: STRING

Max Length 8192 chars

Example:

```
>> GET SETUP.SYSTEM.CUSTOM3
<< +SETUP.SYSTEM.CUSTOM3 ""
<< *GET SETUP.SYSTEM.LOCATING

>> SET SETUP.SYSTEM.CUSTOM3 "Custom"
```

```
<< *SET SETUP.SYSTEM.CUSTOM3 "Custom"

>> GET SETUP.SYSTEM.CUSTOM3
<< +SETUP.SYSTEM.CUSTOM3 "Custom"
<< *GET SETUP.SYSTEM.CUSTOM3
```

5.2.4 Input Registers

Register Name	Type	Access	Unit	Range
IN.COUNT (see page 44)	`Integer`	Get		*IID* in [1, .COUNT]
IN-{IID}.NAME (see page 45)	`String[32]`	Get, Set		
IN-{IID}.SENS (see page 45)	`Enum`	Get, Set		{14DBU, 4DBU, -10DBV, MIC}
IN-{IID}.GAIN (see page 46)	`Float`	Get, Set	dB	[-15.0, 15.0] [-48, 0] for Generator
IN-{IID}.STEREO (see page 46)	`Boolean`	Get, Set		
IN-{IID}.HPF_ENABLE (see page 47)	`Boolean`	Get, Set		
IN-{IID}.DYN.SIGNAL (see page 47)	`Float`	Get, Set	dB	[-144, 20.0]
IN-{IID}.DYN.CLIP (see page 48)	`Boolean`	Get, Set		

5.2.4.1 IN.COUNT

TYPE: Register

METHODS: Get

VALUES: INTEGER

Example:

```
>> GET IN.COUNT
<< +IN.COUNT 7
<< *GET IN.COUNT
```

5.2.4.2 IN-{IID}.NAME

TYPE: Register

METHODS: Get, Set

PARAMS:

{IID}: See {IID} Input Channels (see page 20)

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET IN-100.NAME
<< +IN-100.NAME "ANALOG 1"
<< *GET IN-100.NAME

>> SET IN-100.NAME "CD Player"
<< *SET IN-100.NAME "CD Player"

>> GET IN-100.NAME
<< +IN-100.NAME "CD Player"
<< *GET IN-100.NAME
```

5.2.4.3 IN-{IID}.SENS

TYPE: Register

METHODS: Get, Set

PARAMS:

{IID}: See {IID} Input Channels (see page 20)

VALUES: Enumeration

14DBU: 14 DBU Sensitivity - Max input (ADC Clip) +24 DBU

4DBU: 4 DBU Sensitivity - Max input (ADC Clip) +14 DBU

-10DBV: -10 dBV Sensitivity - Max input (ADC Clip) +4 DBU

MIC: Max sensitivity for Microphone

Example:

```
>> GET IN-100.SENS
```

```
<< +IN-100.SENS "4DBU"
<< *GET IN-100.SENS

>> SET IN-100.SENS "-10DBV"
<< *SET IN-100.SENS "-10DBV"

>> GET IN-100.SENS
<< +IN-100.SENS "-10DBV"
<< *GET IN-100.SENS
```

5.2.4.4 IN-{IID}.GAIN

TYPE: Register

METHODS: Get, Set

PARAMS:

  {IID}: See {IID} Input Channels (see page 20)

VALUES: FLOAT

  Gain in dB. Range [-15.0 - 15.0], [-48, 0] for Generator

Example:

```
>> GET IN-100.GAIN
<< +IN-100.GAIN 0.000
<< *GET IN-100.GAIN

>> SET IN-100.GAIN -4.0
<< *SET IN-100.GAIN -4.0

>> GET IN-100.GAIN
<< +IN-100.GAIN -4.000
<< *GET IN-100.GAIN
```

5.2.4.5 IN-{IID}.STEREO

TYPE: Register

METHODS: Get, Set

PARAMS:

  {IID}: See {IID} Input Channels (see page 20)

VALUES: BOOLEAN

NOTES:

  Only valid for *PRIMARY* channels: 100, 102, 200. Error if other channel or generator

Example:

```
>> GET IN-100.STEREO
<< +IN-100.STEREO 0
<< *GET IN-100.STEREO

>> SET IN-100.STEREO 1
<< *SET IN-100.STEREO 1

>> GET IN-100.STEREO
<< +IN-100.STEREO 1
<< *GET IN-100.STEREO
```

5.2.4.6 IN-{IID}.HPF_ENABLE

TYPE: Register

METHODS: Get, Set

PARAMS:

{IID}: See {IID} Input Channels (see page 20)

VALUES: BOOLEAN

NOTES:

Only valid for Analog channels: 100-199

Example:

```
>> GET IN-100.HPF_ENABLE
<< +IN-100.HPF_ENABLE 0
<< *GET IN-100.HPF_ENABLE

>> SET IN-100.HPF_ENABLE 1
<< *SET IN-100.HPF_ENABLE 1

>> GET IN-100.HPF_ENABLE
<< +IN-100.HPF_ENABLE 1
<< *GET IN-100.HPF_ENABLE
```

5.2.4.7 IN-{IID}.DYN.SIGNAL

TYPE: Subscription Only

PARAMS:

{IID}: See {IID} Input Channels (see page 20)

VALUES: FLOAT

Signal level in dB. Range [-144 - 20]. -144 if no signal

NOTES:

Updated every 50 ms.

Example:

```
>> SUBSCRIBE
<< *SUBSCRIBE
...
<< +IN-100.DYN.SIGNAL -73.4993
<< +IN-101.DYN.SIGNAL -72.8205
<< +IN-102.DYN.SIGNAL -101.728
<< +IN-103.DYN.SIGNAL -98.6826
<< +IN-200.DYN.SIGNAL -144
<< +IN-201.DYN.SIGNAL -144
```

5.2.4.8 IN-{IID}.DYN.CLIP

TYPE: Subscription Only

PARAMS:

{IID}: See {IID} Input Channels (see page 20)

VALUES: BOOLEAN

Signal Clip. True when ADC is clipping

NOTES:

Updated every 50 ms.

Example:

```
>> SUBSCRIBE
<< *SUBSCRIBE
...
<< +IN-100.DYN.CLIP 0
<< +IN-101.DYN.CLIP 0
<< +IN-102.DYN.CLIP 0
<< +IN-103.DYN.CLIP 0
<< +IN-200.DYN.CLIP 0
```

5.2.5 Input Eq Registrers

Register Name	Type	Access	Unit	Range
IN.EQ.COUNT (see page 49)	`Integer`	Get		*EID* in [1, .COUNT]

Register Name	Type	Access	Unit	Range
IN-{IID}.EQ.BYPASS (see page 49)	`Boolean`	Get, Set		
IN-{IID}.EQ-{EID}.TYPE (see page 50)	`Enum`	Get, Set		{PARAMETRIC, LOWPASS_12, HIGHPASS_12, LOW_SHELF_Q, HIGH_SHELF_Q}
IN-{IID}.EQ-{EID}.GAIN (see page 51)	`Float`	Get, Set	dB	[-15, 15]
IN-{IID}.EQ-{EID}.FREQ (see page 51)	`Float`	Get, Set	Hz	[20, 20000]
IN-{IID}.EQ-{EID}.Q (see page 52)	`Float`	Get, Set		[0.4, 30]
IN-{IID}.EQ-{EID}.BYPASS (see page 52)	`Boolean`	Get, Set		

5.2.5.1 IN.EQ.COUNT

TYPE: Register

METHODS: Get

VALUES: INTEGER

NOTES:

Gets number of Input EQ Bands.

Example:

```
>> GET IN.EQ.COUNT
<< +IN.EQ.COUNT 5
<< *GET IN.EQ.COUNT
```

5.2.5.2 IN-{IID}.EQ.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

{IID}: See {IID} Input Channels (see page 20)

VALUES: BOOLEAN

NOTES:

Enable/Disable EQ for channel {IID}. Only valid for Analog inputs: 100-199.

Example:

```
>> GET IN-100.EQ.BYPASS
<< +IN-100.EQ.BYPASS 0
<< *GET IN-100.EQ.BYPASS

>> SET IN-100.EQ.BYPASS 1
<< *SET IN-100.EQ.BYPASS 1

>> GET IN-100.EQ.BYPASS
<< +IN-100.EQ.BYPASS 1
<< *GET IN-100.EQ.BYPASS
```

5.2.5.3 IN-{IID}.EQ-{EID}.TYPE

TYPE: Register

METHODS: Get, Set

PARAMS:

{IID}: See {IID} Input Channels (see page 20)

{EID}: See {EID} Equalizer Bands (see page 23)

VALUES: ENUM

NOTES:

Equalizer band type. Only valid for Analog inputs: 100-199

Example:

```
>> GET IN-100.EQ-1.TYPE
<< +IN-100.EQ-1.TYPE PARAMETRIC
<< *GET IN-100.EQ-1.TYPE

>> SET IN-100.EQ-1.TYPE NOTCH
<< *SET IN-100.EQ-1.TYPE NOTCH

>> GET IN-100.EQ-1.TYPE
<< +IN-100.EQ-1.TYPE NOTCH
<< *GET IN-100.EQ-1.TYPE
```

5.2.5.4 IN-{IID}.EQ-{EID}.GAIN

TYPE: Register

METHODS: Get, Set

PARAMS:

{IID}: See {IID} Input Channels (see page 20)

{EID}: See {EID} Equalizer Bands (see page 23)

VALUES: FLOAT

Gain in dB. Range [-80, 0].

NOTES: Equalizer band type. Only valid for Analog inputs: 100-199

Example:

```
>> GET IN-100.EQ-1.GAIN
<< +IN-100.EQ-1.GAIN 0.0
<< *GET IN-100.EQ-1.GAIN

>> SET IN-100.EQ-1.GAIN 1.0
<< *SET IN-100.EQ-1.GAIN 1.0

>> GET IN-100.EQ-1.GAIN
<< +IN-100.EQ-1.GAIN 1.0
<< *GET IN-100.EQ-1.GAIN
```

5.2.5.5 IN-{IID}.EQ-{EID}.FREQ

TYPE: Register

METHODS: Get, Set

PARAMS:

{IID}: See {IID} Input Channels (see page 20)

{EID}: See {EID} Equalizer Bands (see page 23)

VALUES: FLOAT

Frequency in Hz. Range [20, 20000]

NOTES:

Equalizer band type. Only valid for Analog inputs: 100-199

Example:

```
>> GET IN-100.EQ-1.FREQ
<< +IN-100.EQ-1.FREQ 100
<< *GET IN-100.EQ-1.FREQ
```

```
>> SET IN-100.EQ-1.FREQ 200
<< *SET IN-100.EQ-1.FREQ 200

>> GET IN-100.EQ-1.FREQ
<< +IN-100.EQ-1.FREQ 200.0
<< *GET IN-100.EQ-1.FREQ
```

5.2.5.6 IN-{IID}.EQ-{EID}.Q

TYPE: Register

METHODS: Get, Set

PARAMS:

{IID}: See {IID} Input Channels (see page 20)

{EID}: See {EID} Equalizer Bands (see page 23)

VALUES: FLOAT

NOTES:

Equalizer band type. Only valid for Analog inputs: 100-199

Example:

```
>> GET IN-100.EQ-1.Q
<< +IN-100.EQ-1.Q 0.7
<< *GET IN-100.EQ-1.Q

>> SET IN-100.EQ-1.Q 1.5
<< *SET IN-100.EQ-1.Q 1.5

>> GET IN-100.EQ-1.Q
<< +IN-100.EQ-1.Q 1.5
<< *GET IN-100.EQ-1.Q
```

5.2.5.7 IN-{IID}.EQ-{EID}.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

{IID}: See {IID} Input Channels (see page 20)

{EID}: See {EID} Equalizer Bands (see page 23)

VALUES: BOOLEAN

NOTES:

Bypass the equalizer band. Only valid for Analog channels: 100-199

Example:

```
>> GET IN-100.EQ-1.BYPASS
<< +IN-100.EQ-1.BYPASS 0
<< *GET IN-100.EQ-1.BYPASS

>> SET IN-100.EQ-1.BYPASS 1
<< *SET IN-100.EQ-1.BYPASS 1

>> GET IN-100.EQ-1.BYPASS
<< +IN-100.EQ-1.BYPASS 1
<< *GET IN-100.EQ-1.BYPASS
```

5.2.6 Zone Registers

Register Name	Type	Access	Unit	Range
ZONE.COUNT (see page 54)	`Integer`	Get, Set		*ZID* in [1, .COUNT]
ZONE-{ZID}.NAME (see page 54)	`String[32]`	Get, Set		
ZONE-{ZID}.PRIMARY_SRC (see page 55)	`Integer`	Get, Set		Valid *IID*
ZONE-{ZID}.PRIORITY_SRC (see page 55)	`Integer`	Get, Set		Valid *IID*
ZONE-{ZID}.GAIN (see page 56)	`Float`	Get, Set	dB	[GAIN_MIN, GAIN_MAX]
ZONE-{ZID}.GAIN_MIN (see page 56)	`Float`	Get, Set	dB	[-80, GAIN_MAX]
ZONE-{ZID}.GAIN_MAX (see page 57)	`Float`	Get, Set	dB	[GAIN_MIN, 0]
ZONE-{ZID}.STEREO (see page 57)	`Boolean`	Get, Set		
ZONE-{ZID}.GPIO_VC (see page 58)	`Integer`	Get, Set		Valid *VID*, *0 for OFF*

Register Name	Type	Access	Unit	Range
ZONE-{ZID}.MUTE (see page 58)	`Boolean`	Get, Set		
ZONE-{ZID}.MUTE_ENABLE (see page 59)	`Boolean`	Get, Set		
ZONE-{ZID}.SRC-{SID}.ENABLED (see page 59)	`Boolean`	Get, Set		
ZONE-{ZID}.DYN.SIGNAL (see page 60)	`Float`	Subscribe	dB	[-144, 20.0]

5.2.6.1 ZONE.COUNT

TYPE: Register

METHODS: Get

VALUES: INTEGER

Example:

```
>> GET ZONE.COUNT
<< +ZONE.COUNT 2
<< *GET ZONE.COUNT
```

5.2.6.2 ZONE-{ZID}.NAME

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET ZONE-A.NAME
<< +ZONE-A.NAME "ZONE A"
<< *GET ZONE-A.NAME
```

```
>> SET ZONE-A.NAME "Bar"
<< *SET ZONE-A.NAME "Bar"

>> GET ZONE-A.NAME
<< +ZONE-A.NAME "Bar"
<< *GET ZONE-A.NAME
```

5.2.6.3 ZONE-{ZID}.PRIMARY_SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: Input Source ID.

See {SID} Input Source (see page 21)

Example:

```
>> GET ZONE-A.PRIMARY_SRC
<< +ZONE-A.PRIMARY_SRC 100
<< *GET ZONE-A.PRIMARY_SRC

>> SET ZONE-A.PRIMARY_SRC 100
<< *SET ZONE-A.PRIMARY_SRC 100

>> GET ZONE-A.STEREO
<< +ZONE-A.PRIMARY_SRC 100
<< *GET ZONE-A.PRIMARY_SRC 100
```

5.2.6.4 ZONE-{ZID}.PRIORITY_SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: Input Source ID.

See {SID} Input Source (see page 21)

Example:

```
>> GET ZONE-A.PRIORITY_SRC
<< +ZONE-A.PRIORITY_SRC 100
<< *GET ZONE-A.PRIORITY_SRC
```

```
>> SET ZONE-A.PRIORITY_SRC 100
<< *SET ZONE-A.PRIORITY_SRC 100

>> GET ZONE-A.STEREO
<< +ZONE-A.PRIORITY_SRC 100
<< *GET ZONE-A.PRIORITY_SRC 100
```

5.2.6.5 ZONE-{ZID}.GAIN

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Gain in dB. Range [ZONE-{ZID}.GAIN_MIN (see page 56) - ZONE-{ZID}.GAIN_MAX (see page 57)].

Default [-80, 0]

NOTES:

Read-Only if ZONE-{ZID}.GPIO_VC (see page 58) is set on zone

Example:

```
>> GET ZONE-A.GAIN
<< +ZONE-A.GAIN -40.00
<< *GET ZONE-A.GAIN

>> SET ZONE-A.GAIN -20.0
<< *SET ZONE-A.GAIN -20.0

>> GET ZONE-A.GAIN
<< +ZONE-A.GAIN -20.000
<< *GET ZONE-A.GAIN
```

5.2.6.6 ZONE-{ZID}.GAIN_MIN

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Minimum Gain in dB. Range [-80.0 - ZONE-{ZID}.GAIN_MAX (see page 57)]

Example:

```
>> GET ZONE-A.GAIN_MIN
<< +ZONE-A.GAIN_MIN -40.00
<< *GET ZONE-A.GAIN_MIN

>> SET ZONE-A.GAIN_MIN -20.0
<< *SET ZONE-A.GAIN_MIN -20.0

>> GET ZONE-A.GAIN_MIN
<< +ZONE-A.GAIN_MIN -20.000
<< *GET ZONE-A.GAIN_MIN
```

5.2.6.7 ZONE-{ZID}.GAIN_MAX

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Gain in dB. Range [ZONE-{ZID}.GAIN_MIN (see page 56) - 0.0]

Example:

```
>> GET ZONE-A.GAIN_MAX
<< +ZONE-A.GAIN_MAX -40.00
<< *GET ZONE-A.GAIN_MAX

>> SET ZONE-A.GAIN_MAX -20.0
<< *SET ZONE-A.GAIN_MAX -20.0

>> GET ZONE-A.GAIN_MAX
<< +ZONE-A.GAIN_MAX -20.000
<< *GET ZONE-A.GAIN_MAX
```

5.2.6.8 ZONE-{ZID}.STEREO

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: BOOLEAN

NOTES:

Only valid for *PRIMARY* zones: 'A' and 'C'. Error if *secondary* zone

Example:

```
>> GET ZONE-A.STEREO
<< +ZONE-A.STEREO 0
<< *GET ZONE-A.STEREO

>> SET ZONE-A.STEREO 1
<< *SET ZONE-A.STEREO 1

>> GET ZONE-A.STEREO
<< +ZONE-A.STEREO 1
<< *GET ZONE-A.STEREO
```

5.2.6.9 ZONE-{ZID}.GPIO_VC

TYPE: Register

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: INTEGER

See {VID} Volume Controls (see page 23)

NOTES:

The register will not check if the GPIO pin is configured for Volume Control - which is required for the External Volume control to work.

Example:

```
>> GET ZONE-A.GPIO_VC
<< +ZONE-A.GPIO_VC 0
<< *GET ZONE-A.GPIO_VC

>> SET ZONE-A.GPIO_VC 1
<< *SET ZONE-A.GPIO_VC 1

>> GET ZONE-A.GPIO_VC
<< +ZONE-A.GPIO_VC 1
<< *GET ZONE-A.GPIO_VC
```

5.2.6.10 ZONE-{ZID}.MUTE

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: BOOLEAN

Example:

```
>> GET ZONE-A.MUTE
<< +ZONE-A.MUTE 0
<< *GET ZONE-A.MUTE

>> SET ZONE-A.MUTE 1
<< *SET ZONE-A.MUTE 1

>> GET ZONE-A.MUTE
<< +ZONE-A.MUTE 1
<< *GET ZONE-A.MUTE
```

5.2.6.11 ZONE-{ZID}.MUTE_ENABLE

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: BOOLEAN

Example:

```
>> GET ZONE-A.MUTE_ENABLE
<< +ZONE-A.MUTE_ENABLE 0
<< *GET ZONE-A.MUTE_ENABLE

>> SET OUT-1.MUTE_ENABLE 1
<< *SET ZONE-A.MUTE_ENABLE 1

>> GET OUT-1.MUTE_ENABLE
<< +ZONE-A.MUTE_ENABLE 1
<< *GET ZONE-A.MUTE_ENABLE
```

5.2.6.12 ZONE-{ZID}.SRC-{SID}.ENABLED

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

{SID}: See {SID} Input Source (see page 21)

VALUES: BOOLEAN

NOTES:

Limits the selectable inputs for the Primary SRC.
If an InputID is set to disabled it cannot be selected as a Primary Src

Example:

```
>> GET ZONE-A.SRC-100.ENABLED
<< +ZONE-A.SRC-100.ENABLED 1
<< *GET ZONE-A.SRC-100.ENABLED

>> SET ZONE-A.SRC-100.ENABLED 0
<< *SET ZONE-A.SRC-100.ENABLED 0

>> GET ZONE-A.STEREO
<< +ZONE-A.SRC-100.ENABLED 0
<< *GET ZONE-A.SRC-100.ENABLED 100
```

5.2.6.13 ZONE-{ZID}.DYN.SIGNAL

TYPE: Subscription Only

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Signal level in dB. Range [-144 - 20]. -144 if no signal

NOTES: Updated every 50 ms.

Example:

```
>> SUBSCRIBE
<< *SUBSCRIBE
...
<< ZONE-A.DYN.SIGNAL -100.358
<< ZONE-B.DYN.SIGNAL -99.9367
```

5.2.7 Zone Ducker Registers

Register Name	Type	Access	Unit	Range	
ZONE-{ZID}.DUCK.MODE (see page 61)		Enum			{OFF, DUCKER, OVERRIDE}

Register Name	Type	Access	Unit	Range	
ZONE-{ZID}.DUCK.AUTO (see page 62)	`Boolean`	Get, Set			
ZONE-{ZID}.DUCK.THRESHOLD (see page 62)	`Float`	Get, Set	dB	[-80, 0]	
ZONE-{ZID}.DUCK.DEPTH (see page 63)	`Float`	Get, Set	Sec	[-144, 0]	
ZONE-{ZID}.DUCK.ATTACK (see page 63)	`Float`	Get, Set	Sec	[0.001, 0.2]	
ZONE-{ZID}.DUCK.RELEASE (see page 64)	`Float`	Get, Set	Sec	[0.010, 10.0]	
ZONE-{ZID}.DUCK.HOLD (see page 64)	`Float`	Get, Set		[0, 10]	
ZONE-{ZID}.DUCK.OVERRIDE_GAIN (see page 65)	`Float`	Get, Set	dB	[-60, 15]	
ZONE-{ZID}.DUCK.OVERRIDE_GAIN_ENABLE (see page 65)	`Boolean`	Get, Set			

5.2.7.1 ZONE-{ZID}.DUCK.MODE

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: ENUM

OFF: Ducker is Off

DUCKER: Ducking Mode

OVERRIDE: Input Override Mode

Example:

```
>> GET ZONE-A.DUCK.MODE
<< +ZONE-A.DUCK.MODE "OFF"
<< *GET ZONE-A.DUCK.MODE

>> SET ZONE-A.DUCK.MODE OVERRIDE
<< *SET ZONE-A.DUCK.MODE 1

>> GET ZONE-A.DUCK.MODE
<< +ZONE-A.DUCK.MODE "OVERRIDE"
<< *GET ZONE-A.DUCK.MODE
```

5.2.7.2 ZONE-{ZID}.DUCK.AUTO

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: BOOLEAN

Example:

```
>> GET ZONE-A.DUCK.AUTO
<< +ZONE-A.DUCK.AUTO 1
<< *GET ZONE-A.DUCK.AUTO

>> SET ZONE-A.DUCK.AUTO 0
<< *SET ZONE-A.DUCK.AUTO 0

>> GET ZONE-A.DUCK.0
<< +ZONE-A.DUCK.AUTO 0
<< *GET ZONE-A.DUCK.AUTO
```

5.2.7.3 ZONE-{ZID}.DUCK.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Threshold in dB. Range [xx, xx]

Example:

```
>> GET ZONE-A.DUCK.THRESHOLD
<< +ZONE-A.DUCK.THRESHOLD -10
<< *GET ZONE-A.DUCK.THRESHOLD

>> SET ZONE-A.DUCK.THRESHOLD -5
<< *SET ZONE-A.DUCK.THRESHOLD -5

>> GET ZONE-A.DUCK.THRESHOLD
<< +ZONE-A.DUCK.THRESHOLD -5
<< *GET ZONE-A.DUCK.THRESHOLD
```

5.2.7.4 ZONE-{ZID}.DUCK.DEPTH

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Depth in dB. Range [xx, xx]

Example:

```
>> GET ZONE-A.DUCK.DEPTH
<< +ZONE-A.DUCK.DEPTH -10
<< *GET ZONE-A.DUCK.DEPTH

>> SET ZONE-A.DUCK.DEPTH -5
<< *SET ZONE-A.DUCK.DEPTH -5

>> GET ZONE-A.DUCK.DEPTH
<< +ZONE-A.DUCK.DEPTH -5
<< *GET ZONE-A.DUCK.DEPTH
```

5.2.7.5 ZONE-{ZID}.DUCK.ATTACK

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Attack time in seconds. Range [xx, xx]

Example:

```
>> GET ZONE-A.DUCK.ATTACK
<< +ZONE-A.DUCK.ATTACK 0.050
<< *GET ZONE-A.DUCK.ATTACK

>> SET ZONE-A.DUCK.ATTACK 0.1
<< *SET ZONE-A.DUCK.ATTACK 0.1

>> GET ZONE-A.DUCK.ATTACK
<< +ZONE-A.DUCK.ATTACK 0.100
<< *GET ZONE-A.DUCK.ATTACK
```

5.2.7.6 ZONE-{ZID}.DUCK.RELEASE

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Release in seconds. Range [xx, xx]

Example:

```
>> GET ZONE-A.DUCK.RELEASE
<< +ZONE-A.DUCK.RELEASE 0.500
<< *GET ZONE-A.DUCK.RELEASE

>> SET ZONE-A.DUCK.RELEASE 1.0
<< *SET ZONE-A.DUCK.RELEASE 1.0

>> GET ZONE-A.DUCK.RELEASE
<< +ZONE-A.DUCK.RELEASE 1.000
<< *GET ZONE-A.DUCK.RELEASE
```

5.2.7.7 ZONE-{ZID}.DUCK.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Hold in seconds. Range [xx, xx]

Example:

```
>> GET ZONE-A.DUCK.HOLD
<< +ZONE-A.DUCK.HOLD 0.000
<< *GET ZONE-A.DUCK.HOLD

>> SET ZONE-A.DUCK.HOLD 1.0
<< *SET ZONE-A.DUCK.HOLD 1.0

>> GET ZONE-A.DUCK.HOLD
<< +ZONE-A.DUCK.HOLD 1.000
<< *GET ZONE-A.DUCK.HOLD
```

5.2.7.8 ZONE-{ZID}.DUCK.OVERRIDE_GAIN

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Gain in dB. Range[xx, xx]

Example:

```
>> GET ZONE-A.DUCK.OVERRIDE_GAIN
<< +ZONE-A.DUCK.OVERRIDE_GAIN 0.0
<< *GET ZONE-A.DUCK.OVERRIDE_GAIN

>> SET ZONE-A.DUCK.OVERRIDE_GAIN -20
<< *SET ZONE-A.DUCK.OVERRIDE_GAIN -20

>> GET ZONE-A.DUCK.OVERRIDE_GAIN
<< +ZONE-A.DUCK.OVERRIDE_GAIN -20.0
<< *GET ZONE-A.DUCK.OVERRIDE_GAIN
```

5.2.7.9 ZONE-{ZID}.DUCK.OVERRIDE_GAIN_ENABLE

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: BOOLEAN

Example:

```
>> GET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE
<< +ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE 0
<< *GET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE

>> SET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE 1
<< *SET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE 1

>> GET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE
<< +ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE 1
<< *GET ZONE-A.DUCK.OVERRIDE_GAIN_ENABLE
```

5.2.8 Zone Compressor Registers

Register Name	Type	Access	Unit	Range
ZONE-{ZID}.COMPRESSOR.AUTO (see page 67)	`Boolean`	Get, Set		
ZONE-{ZID}.COMPRESSOR.THRESHOLD (see page 67)	`Float`	Get, Set	dB	[-40, 20]
ZONE-{ZID}.COMPRESSOR.ATTACK (see page 68)	`Float`	Get, Set	Sec	[0.0003, 0.050]
ZONE-{ZID}.COMPRESSOR.RELEASE (see page 68)	`Float`	Get, Set	Sec	[0.001, 1.0]
ZONE-{ZID}.COMPRESSOR.HOLD (see page 69)	`Float`	Get, Set	Sec	[0, 1]
ZONE-{ZID}.COMPRESSOR.RATIO (see page 69)	`Float`	Get, Set		[1, 50]
ZONE-{ZID}.COMPRESSOR.KNEE (see page 70)	`Float`	Get, Set	dB	[0, 12]
ZONE-{ZID}.COMPRESSOR.BYPASS (see page 70)	`Boolean`	Get, Set		

5.2.8.1 ZONE-{ZID}.COMPRESSOR.AUTO

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: BOOLEAN

NOTES: Use automatic parameters for Attack, Release and Ratio based on crossover frequency

Example:

```
>> GET ZONE-A.COMPRESSOR.AUTO
<< +ZONE-A.COMPRESSOR.AUTO 1
<< *GET ZONE-A.COMPRESSOR.AUTO

>> SET ZONE-A.COMPRESSOR.AUTO 0
<< *SET ZONE-A.COMPRESSOR.AUTO 0

>> GET ZONE-A.COMPRESSOR.AUTO
<< +ZONE-A.COMPRESSOR.AUTO 0
<< *GET ZONE-A.COMPRESSOR.AUTO
```

5.2.8.2 ZONE-{ZID}.COMPRESSOR.THRESHOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Threshold for compressor in dBFS. Range [-40, 20]

Example:

```
>> GET ZONE-A.COMPRESSOR.THRESHOLD
<< +ZONE-A.COMPRESSOR.THRESHOLD 0.000
<< *GET ZONE-A.COMPRESSOR.THRESHOLD

>> SET ZONE-A.COMPRESSOR.THRESHOLD -10
<< *SET ZONE-A.COMPRESSOR.THRESHOLD -10

>> GET ZONE-A.COMPRESSOR.THRESHOLD
<< +ZONE-A.COMPRESSOR.THRESHOLD -10.000
<< *GET ZONE-A.COMPRESSOR.THRESHOLD
```

5.2.8.3 ZONE-{ZID}.COMPRESSOR.ATTACK

TYPE: Register

METHODS: Get, Set

PARAMS:

 {ZID}: See {ZID} Zones (see page 21)

PATH:

VALUES: FLOAT

 Attack Time for compressor in seconds. Range [0.0003, 0.050]

Example:

```
>> GET ZONE-A.COMPRESSOR.ATTACK
<< +ZONE-A.COMPRESSOR.ATTACK 0.045
<< *GET ZONE-A.COMPRESSOR.ATTACK

>> SET ZONE-A.COMPRESSOR.ATTACK 0.1
<< *SET ZONE-A.COMPRESSOR.ATTACK 0.1

>> GET ZONE-A.COMPRESSOR.ATTACK
<< +ZONE-A.COMPRESSOR.ATTACK 0.100
<< *GET ZONE-A.COMPRESSOR.ATTACK
```

5.2.8.4 ZONE-{ZID}.COMPRESSOR.RELEASE

TYPE: Register

METHODS: Get, Set

PARAMS:

 {ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

 Release Time for compressor in seconds. Range [0.001, 1.0]

Example:

```
>> GET ZONE-A.COMPRESSOR.RELEASE
<< +ZONE-A.COMPRESSOR.RELEASE 0.750
<< *GET ZONE-A.COMPRESSOR.RELEASE

>> SET ZONE-A.COMPRESSOR.RELEASE 0.8
<< *SET ZONE-A.COMPRESSOR.RELEASE 0.8

>> GET ZONE-A.COMPRESSOR.RELEASE
<< +ZONE-A.COMPRESSOR.RELEASE 0.800
<< *GET ZONE-A.COMPRESSOR.RELEASE
```

5.2.8.5 ZONE-{ZID}.COMPRESSOR.RATIO

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Ratio for compressor. Range [1, 50]

Example:

```
>> GET ZONE-A.COMPRESSOR.RATIO
<< +ZONE-A.COMPRESSOR.RATIO 10.000
<< *GET ZONE-A.COMPRESSOR.RATIO

>> SET ZONE-A.COMPRESSOR.RATIO 12
<< *SET ZONE-A.COMPRESSOR.RATIO 12

>> GET ZONE-A.COMPRESSOR.RATIO
<< +ZONE-A.COMPRESSOR.RATIO 12.000
<< *GET ZONE-A.COMPRESSOR.RATIO
```

5.2.8.6 ZONE-{ZID}.COMPRESSOR.HOLD

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Hold for compressor in seconds. Range [0.0, 1.0]

Example:

```
>> GET ZONE-A.COMPRESSOR.HOLD
<< +ZONE-A.COMPRESSOR.HOLD 0.000
<< *GET ZONE-A.COMPRESSOR.HOLD

>> SET ZONE-A.COMPRESSOR.HOLD 0.1
<< *SET ZONE-A.COMPRESSOR.HOLD 0.1

>> GET ZONE-A.COMPRESSOR.HOLD
<< +ZONE-A.COMPRESSOR.HOLD 0.100
<< *GET ZONE-A.COMPRESSOR.HOLD
```

5.2.8.7 ZONE-{ZID}.COMPRESSOR.KNEE

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: FLOAT

Knee for compressor. Range [1, 12]

Example:

```
>> GET ZONE-A.COMPRESSOR.KNEE
<< +ZONE-A.COMPRESSOR.KNEE 4.000
<< *GET ZONE-A.COMPRESSOR.KNEE

>> SET ZONE-A.COMPRESSOR.KNEE 5
<< *SET ZONE-A.COMPRESSOR.KNEE 5

>> GET ZONE-A.COMPRESSOR.KNEE
<< +ZONE-A.COMPRESSOR.KNEE 5.000
<< *GET ZONE-A.COMPRESSOR.KNEE
```

5.2.8.8 ZONE-{ZID}.COMPRESSOR.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

{ZID}: See {ZID} Zones (see page 21)

VALUES: BOOLEAN

Bypass compressor. Set to 0 to enable compressor, 1 to disable.

Example:

```
>> GET ZONE-A.COMPRESSOR.BYPASS
<< +ZONE-A.COMPRESSOR.BYPASS 1
<< *GET ZONE-A.COMPRESSOR.BYPASS

>> SET ZONE-A.COMPRESSOR.BYPASS 0
<< *SET ZONE-A.COMPRESSOR.THRESHOLD 0

>> GET ZONE-A.COMPRESSOR.BYPASS
<< +ZONE-A.COMPRESSOR.BYPASS 0
<< *GET ZONE-A.COMPRESSOR.BYPASS
```

5.2.9 Output Registers

Register Name	Type	Access	Unit	Range
OUTPUT.COUNT (see page 72)	`Integer`	Get		*OID* in [1, .COUNT]
OUT-{OID}.NAME (see page 72)	`String[32]`	Get, Set		
OUT-{OID}.GAIN (see page 75)	`Float`	Get, Set	dB	[-30.0, 15.0]
OUT-{OID}.MUTE (see page 76)	`Boolean`	Get, Set		
OUT-{OID}.SRC (see page 72)	`String[1]`	Get, Set		*ZID*
OUT-{OID}.SRC_CHANNEL (see page 73)	`Enum`	Get, Set		{L, R, S}
OUT-{OID}.POLARITY (see page 74)	`Integer`	Get, Set		{-1, 1}
OUT-{OID}.OUTPUT_MODE (see page 74)	`Enum`	Get, Set		{OFF, 8R, 70V, 100V, BTL}
OUT-{OID}.OUTPUT_HIGHPASS (see page 75)	`Float`	Get, Set	Hz	{0, [20-1000]}
OUT-{OID}.DYN.SIGNAL (see page 76)	`Float`	Subscribe	dB	[-144, 20.0]
OUT-{OID}.DYN.CLIP (see page 77)	`Boolean`	Subscribe		

5.2.9.1 OUTPUT.COUNT

TYPE: Register

METHODS: Get

VALUES: INTEGER

Example:

```
>> GET OUT.COUNT
<< +OUT.COUNT 2
<< *GET OUT.COUNT
```

5.2.9.2 OUT-{OID}.NAME

TYPE: Register

METHODS: Get, Set

PARAMS:

{OID}: See {OID} Output Channels (see page 21)

VALUES: STRING

Max Length 32 chars

Example:

```
>> GET OUT-1.NAME
<< +OUT-1.NAME "Output CH 1"
<< *GET OUT-1.NAME

>> SET OUT-1.NAME "Left Speaker"
<< *SET OUT-1.NAME "Left Speaker"

>> GET OUT-1.NAME
<< +OUT-1.NAME "Left Speaker"
<< *GET OUT-1.NAME
```

5.2.9.3 OUT-{OID}.SRC

TYPE: Register

METHODS: Get, Set

PARAMS:

{OID}: See {OID} Output Channels (see page 21)

VALUES: STRING

See {ZID} Zones (see page 21)

NOTES: If source zone is stereo it is still possible to select Zone-B but as the value is 'invalid' as Zone-B is undefined when Zone-A is stereo (And links Zone-B) no sound will be playing. If source zone is stereo is is nessesary to set subchannel Source to play Left Channel, Right Channel or Sum of both channels.

Example:

```
>> GET OUT-1.SRC
<< +OUT-1.SRC "A"
<< *GET OUT-1.SRC

>> SET OUT-1.SRC B
<< *SET OUT-1.SRC B

>> GET OUT-1.SRC
<< +OUT-1.SRC "B"
<< *GET OUT-1.SRC
```

5.2.9.4 OUT-{OID}.SRC_CHANNEL

TYPE: Register

METHODS: Get, Set

PARAMS:

{OID}: See {OID} Output Channels (see page 21)

VALUES: ENUM

L - Left Channel Only

R - Right Channel Only

S - For Sum of Left and Right Channels

NOTES: If source zone is stereo it is necessary to set subchannel Source to play Left Channel, Right Channel or Sum of both channels.

Example:

```
>> GET OUT-1.SRC_CHANNEL
<< +OUT-1.SRC_CHANNEL "S"
<< *GET OUT-1.SRC_CHANNEL

>> SET OUT-1.SRC_CHANNEL L
<< *SET OUT-1.SRC_CHANNEL L

>> GET OUT-1.SRC
<< +OUT-1.SRC_CHANNEL "L"
<< *GET OUT-1.SRC_CHANNEL
```

5.2.9.5 OUT-{OID}.POLARITY

TYPE: Register

METHODS: Get, Set

PARAMS:

{OID}: See {OID} Output Channels (see page 21)

VALUES: INTEGER

1 - Normal Polarity
-1 - Reversed Polarity

Example:

```
>> GET OUT-1.POLARITY
<< +OUT-1.POLARITY 1
<< *GET OUT-1.POLARITY

>> SET OUT-1.POLARITY -1
<< *SET OUT-1.POLARITY -1

>> GET OUT-1.SRC
<< +OUT-1.POLARITY -1
<< *GET OUT-1.POLARITY
```

5.2.9.6 OUT-{OID}.OUTPUT_MODE

TYPE: Register

METHODS: Get, Set

PARAMS:

{OID}: See {OID} Output Channels (see page 21)

VALUES: ENUM

OFF - Output is Off
8R - Output is LowZ
70V - Output is HiZ 70 Volt
100V - Output is HiZ 100 Volt
BTL - Output is Bridged - *(Not supported for all models)*

Example:

```
>> GET OUT-1.OUTPUT_MODE
<< +OUT-1.OUTPUT_MODE "8R"
<< *GET OUT-1.OUTPUT_MODE

>> SET OUT-1.OUTPUT_MODE "100V"
<< *SET OUT-1.OUTPUT_MODE "100V"
```

```
>> GET OUT-1.SRC
<< +OUT-1.OUTPUT_MODE "100V"
<< *GET OUT-1.OUTPUT_MODE
```

5.2.9.7 OUT-{OID}.OUTPUT_HIGHPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

{OID}: See {OID} Output Channels (see page 21)

VALUES: FLOAT

Frequency in Hz. Range [20, 1000]

Example:

```
>> GET OUT-1.OUTPUT_HIGHPASS
<< +OUT-1.OUTPUT_HIGHPASS 100.000
<< *GET OUT-1.OUTPUT_HIGHPASS

>> SET OUT-1.OUTPUT_HIGHPASS 80
<< *SET OUT-1.OUTPUT_HIGHPASS 80

>> GET OUT-1.SRC
<< +OUT-1.OUTPUT_HIGHPASS 80.000
<< *GET OUT-1.OUTPUT_HIGHPASS
```

5.2.9.8 OUT-{OID}.GAIN

TYPE: Register

METHODS: Get, Set

PARAMS:

{OID}: See {OID} Output Channels (see page 21)

VALUES: FLOAT

Gain in dB. Range [-30.0 - 15.0]

Example:

```
>> GET OUT-1.GAIN
<< +OUT-1.GAIN 0
<< *GET OUT-1.GAIN

>> SET OUT-1.GAIN 1
<< *SET OUT-1.GAIN 1.0
```

```
>> GET OUT-1.GAIN
<< +OUT-1.GAIN 1.0
<< *GET OUT-1.GAIN
```

5.2.9.9 OUT-{OID}.MUTE

TYPE: Register

METHODS: Get, Set

PARAMS:

 {OID}: See {OID} Output Channels (see page 21)

VALUES: BOOLEAN

Example:

```
>> GET OUT-1.MUTE
<< +OUT-1.MUTE 0
<< *GET OUT-1.MUTE

>> SET OUT-1.MUTE 1
<< *SET OUT-1.MUTE 1

>> GET OUT-1.MUTE
<< +OUT-1.MUTE 1
<< *GET OUT-1.MUTE
```

5.2.9.10 OUT-{OID}.DYN.SIGNAL

TYPE: Subscription Only

PARAMS:

 {OID}: See {OID} Output Channels (see page 21)

VALUES: FLOAT

 Signal level in dB. Range [-144 - 20]. -144 if no signal

NOTES: Updated every 50 ms.

Example:

```
>> SUBSCRIBE
<< *SUBSCRIBE
...
<< +OUT-1.DYN.SIGNAL -73.4993
<< +OUT-2.DYN.SIGNAL -72.8205
```

5.2.9.11 OUT-{OID}.DYN.CLIP

TYPE: Subscription Only

PARAMS:

{OID}: See {OID} Output Channels (see page 21)

VALUES: BOOLEAN

Signal Clip. True when DAC is clipping.

NOTES: Updated every 50 ms.

Example:

```
>> SUBSCRIBE
<< *SUBSCRIBE
...
<< +OUT-1.DYN.CLIP 0
<< +OUT-2.DYN.CLIP 0
```

5.2.10 Output Delay Registers

Register Name	Type	Access	Unit	Range
OUT-{OID}.DELAY.TIME (see page 77)	`Float`	Get, Set	Sec	[0.0, 0.1]
OUT-{OID}.DELAY.BYPASS (see page 78)	`Boolean`	Get, Set		

5.2.10.1 OUT-{OID}.DELAY.TIME

TYPE: Register

METHODS: Get, Set

PARAMS:

{OID}: See {OID} Output Channels (see page 21)

VALUES: FLOAT

Time in seconds. Range [0.0, 0.1]

Example:

```
>> GET OUT-1.DELAY.TIME
<< +OUT-1.DELAY.TIME 0.00000
<< *GET OUT-1.DELAY.TIME

>> SET OUT-1.DELAY.TIME 0.01
<< *SET OUT-1.DELAY.TIME 0.01

>> GET OUT-1.DELAY.TIME
<< +OUT-1.DELAY.TIME 0.01000
<< *GET OUT-1.DELAY.TIME
```

5.2.10.2 OUT-{OID}.DELAY.BYPASS

TYPE: Register

METHODS: Get, Set

PARAMS:

{OID}: See {OID} Output Channels (see page 21)

VALUES: BOOLEAN

Example:

```
>> GET OUT-1.DELAY.BYPASS
<< +OUT-1.DELAY.BYPASS 1
<< *GET OUT-1.DELAY.BYPASS

>> SET OUT-1.DELAY.BYPASS 0
<< *SET OUT-1.DELAY.BYPASS 0

>> GET OUT-1.DELAY.BYPASS
<< +OUT-1.DELAY.BYPASS 0
<< *GET OUT-1.DELAY.BYPASS
```

5.2.11 Generator Registers

Register Name	Type	Access	Unit	Range
GENERATOR.TYPE (see page 79)	`Enum`	Get, Set		{PINK, SINE}
GENERATOR.SINE.FREQ (see page 79)	`Float`	Get, Set	Hz	[20, 20000]

Register Name	Type	Access	Unit	Range
GENERATOR.PINK.LPF_ENABLE (see page 80)	`Boolean`	Get, Set		
GENERATOR.PINK.LPF_FREQ (see page 80)	`Float`	Get, Set	Hz	[20, 20000]
GENERATOR.PINK.HPF_ENABLE (see page 81)	`Boolean`	Get, Set		
GENERATOR.PINK.HPF_FREQ (see page 81)	`Float`	Get, Set	Hz	[20, 20000]

5.2.11.1 GENERATOR.TYPE

TYPE: Register

METHODS: Get, Set

VALUES: ENUM

PINK Pink Noise Generator

SINE Sine Generator

Example:

```
>> GET GENERATOR.TYPE
<< +GENERATOR.TYPE "PINK"
<< *GET GENERATOR.TYPE

>> SET GENERATOR.TYPE PINK
<< *SET GENERATOR.TYPE PINK

>> GET GENERATOR.TYPE
<< +GENERATOR.TYPE "PINK"
<< *GET GENERATOR.TYPE
```

5.2.11.2 GENERATOR.SINE.FREQ

TYPE: Register

METHODS: Get, Set

VALUES: FLOAT

Frequency in Hz. Range [20, 20000]

Example:

```
>> GET GENERATOR.SINE.FREQ
<< +GENERATOR.SINE.FREQ 1000.0
<< *GET GENERATOR.SINE.FREQ

>> SET GENERATOR.SINE.FREQ 1200
<< *SET GENERATOR.SINE.FREQ 1200

>> GET GENERATOR.SINE.FREQ
<< +GENERATOR.SINE.FREQ 1200.0
<< *GET GENERATOR.SINE.FREQ
```

5.2.11.3 GENERATOR.PINK.LPF_ENABLE

TYPE: Register

METHODS: Get, Set

VALUES: BOOLEAN

Example:

```
>> GET GENERATOR.PINK.LPF_ENABLE
<< +GENERATOR.PINK.LPF_ENABLE 0
<< *GET GENERATOR.PINK.LPF_ENABLE

>> SET GENERATOR.PINK.LPF_ENABLE 1
<< *SET GENERATOR.PINK.LPF_ENABLE 1

>> GET GENERATOR.PINK.LPF_ENABLE
<< +GENERATOR.PINK.LPF_ENABLE 1
<< *GET GENERATOR.PINK.LPF_ENABLE
```

5.2.11.4 GENERATOR.PINK.LPF_FREQ

TYPE: Register

METHODS: Get, Set

VALUES: FLOAT

Frequency in Hz. Range [20, 20000]

Example:

```
>> GET GENERATOR.PINK.LPF_FREQ
<< +GENERATOR.PINK.LPF_FREQ 100.0
<< *GET GENERATOR.PINK.LPF_FREQ
```

```
>> SET GENERATOR.PINK.LPF_FREQ 1000
<< *SET GENERATOR.PINK.LPF_FREQ 1000

>> GET GENERATOR.PINK.LPF_FREQ
<< +GENERATOR.PINK.LPF_FREQ 1000.0
<< *GET GENERATOR.PINK.LPF_FREQ
```

5.2.11.5 GENERATOR.PINK.HPF_ENABLE

TYPE: Register

METHODS: Get, Set

VALUES: BOOLEAN

Example:

```
>> GET GENERATOR.PINK.HPF_ENABLE
<< +GENERATOR.PINK.HPF_ENABLE 0
<< *GET GENERATOR.PINK.HPF_ENABLE

>> SET GENERATOR.PINK.HPF_ENABLE 1
<< *SET GENERATOR.PINK.HPF_ENABLE 1

>> GET GENERATOR.PINK.HPF_ENABLE
<< +GENERATOR.PINK.HPF_ENABLE 1
<< *GET GENERATOR.PINK.HPF_ENABLE
```

5.2.11.6 GENERATOR.PINK.HPF_FREQ

TYPE: Register

METHODS: Get, Set

VALUES: FLOAT

Frequency in Hz. Range [20, 20000]

Example:

```
>> GET GENERATOR.PINK.HPF_FREQ
<< +GENERATOR.PINK.HPF_FREQ 100.0
<< *GET GENERATOR.PINK.HPF_FREQ

>> SET GENERATOR.PINK.HPF_FREQ 1000
<< *SET GENERATOR.PINK.HPF_FREQ 1000

>> GET GENERATOR.PINK.HPF_FREQ
<< +GENERATOR.PINK.HPF_FREQ 1000.0
<< *GET GENERATOR.PINK.HPF_FREQ
```

5.2.12 Advanced

Please contact your manufacturer - for help integrating the advanced API's described below.

5.2.12.1 Mix Registers

Register Name	Type	Access	Unit	Range
`MIX.COUNT`	`Integer`	Get		
`MIX-{MID}.NAME`	`String`	Get, Set		
`MIX-{MID}.GAIN-{SID}`	`Float`	Get, Set	dB	[-80, 0]

5.2.12.2 Output Speaker Preset

Register Name	Type	Access	Unit	Range
`OUT-{OID}.PRESET.NAME`	`String`	Get		
`OUT-{OID}.PRESET.ID`	`String`	Get		
`OUT-{OID}.PRESET.LOCKED`	`Boolean`	Get		
`OUT-{OID}.PRESET.CUSTOMIZED`	`Boolean`	Get		
`OUT-{OID}.POLARITY.PROTECTED`	`Boolean`	Get		
`OUT-{OID}.OUTPUT_MODE.PROTECTED`	`Boolean`	Get		
`OUT-{OID}.SPEAKER_DELAY.PROTECTED`	`Boolean`	Get		
`OUT-{OID}.LIMITER.PROTECTED`	`Boolean`	Get		
`OUT-{OID}.SPEAKER_EQ.PROTECTED`	`Boolean`	Get		
`OUT-{OID}.XR.PROTECTED`	`Boolean`	Get		

Register Name	Type	Access	Unit	Range
`OUT-{OID}.FIR.PROTECTED`	`Boolean`	Get		

5.2.12.3 Output Speaker Delay Registers

Register Name	Type	Access	Unit	Range
`OUT-{OID}.SPEAKER_DELAY.TIME`	`Float`	Get, Set	Sec	[0.0, 0.01]
`OUT-{OID}.SPEAKER_DELAY.BYPASS`	`Boolean`	Get, Set		

5.2.12.4 Output Peak Limiter Registers

Register Name	Type	Access	Unit	Range
`OUT-{OID}.PEAK_LIMITER.BYPASS`	`Boolean`	Get, Set		
`OUT-{OID}.PEAK_LIMITER.AUTO`	`Boolean`	Get, Set		
`OUT-{OID}.PEAK_LIMITER.THRESHOLD`	`Float`	Get, Set	Vpeak	[1, 200]
`OUT-{OID}.PEAK_LIMITER.ATTACK`	`Float`	Get, Set	Sec	[0.0003, 0.100]

Register Name	Type	Access	Unit	Range
`OUT-{OID}.PEAK_LIMITER.RELEASE`	`Float`	Get, Set	Sec	[0.004, 2.0]
`OUT-{OID}.PEAK_LIMITER.HOLD`	`Float`	Get, Set	Sec	[0, 1.0]
`OUT-{OID}.PEAK_LIMITER.KNEE`	`Float`	Get, Set	dB	[0, 6.0]

5.2.12.5 Output RMS Limiter Registers

Register Name	Type	Access	Unit	Range
`OUT-{OID}.RMS_LIMITER.BYPASS`	`Boolean`	Get, Set		
`OUT-{OID}.RMS_LIMITER.THRESHOLD`	`Float`	Get, Set	Vpeak	[1, 150]
`OUT-{OID}.RMS_LIMITER.ATTACK`	`Float`	Get, Set	Sec	[0.010, 30]
`OUT-{OID}.RMS_LIMITER.RELEASE`	`Float`	Get, Set	Sec	[0.010, 30]
`OUT-{OID}.RMS_LIMITER.HOLD`	`Float`	Get, Set	Sec	[0, 1.0]
`OUT-{OID}.RMS_LIMITER.KNEE`	`Float`	Get, Set	dB	[0, 6.0]

5.2.12.6 Output Clip Limiter Registrers

Register Name	Type	Access	Unit	Range
`OUT-{OID}.CLIP_LIMITER.BYPASS`	`Boolean`	Get, Set		
`OUT-{OID}.CLIP_LIMITER.MODE`	`Enum`	Get, Set		{NORMAL, FAST}

5.2.12.7 Output Eq Registrers

Register Name	Type	Access	Unit	Range
`OUT.EQ.COUNT`	`Integer`	Get		*EID* in [1, .COUNT]
`OUT-{OID}.EQ.BYPASS`	`Boolean`	Get, Set		
`OUT-{OID}.EQ-{EID}.TYPE`	`Enum`	Get, Set		{PARAMETRIC, LOWPASS_6, HIGHPASS_6, LOWPASS_12, HIGHPASS_12, LOW_SHELF, LOW_SHELF_Q, LOW_SHELF_6, LOW_SHELF_12, HIGH_SHELF, HIGH_SHELF_Q, HIGH_SHELF_6, HIGH_SHELF_12, BANDPASS, NOTCH, ALLPASS_1, ALLPASS_2}
`OUT-{OID}.EQ-{EID}.GAIN`	`Float`	Get, Set	dB	[-15, 15]
`OUT-{OID}.EQ-{EID}.FREQ`	`Float`	Get, Set	Hz	[20, 20000]

Register Name	Type	Access	Unit	Range
`OUT-{OID}.EQ-{EID}.Q`	`Float`	Get, Set		[0.4, 30]
`OUT-{OID}.EQ-{EID}.BYPASS`	`Boolean`	Get, Set		

5.2.12.8 Output SpeakerEq Registers

Register Name	Type	Access	Unit	Range
`OUT.SPEAKER_EQ.COUNT`	`Integer`	Get		*SID* in [1, .COUNT]
`OUT-{OID}.SPEAKER_EQ.BYPASS`	`Boolean`	Get, Set		
`OUT-{OID}.SPEAKER_EQ-{EID}.TYPE`	`Enum`	Get, Set		{PARAMETRIC, LOWPASS_6, HIGHPASS_6, LOWPASS_12, HIGHPASS_12, LOW_SHELF_6, LOW_SHELF_12, HIGH_SHELF, HIGH_SHELF_Q, HIGH_SHELF_6, HIGH_SHELF_12, BANDPASS, NOTCH, ALLPASS_1, ALLPASS_2}
`OUT-{OID}.SPEAKER_EQ-{EID}.GAIN`	`Float`	Get, Set	dB	[-15, 15]
`OUT-{OID}.SPEAKER_EQ-{EID}.FREQ`	`Float`	Get, Set	Hz	[20, 20000]
`OUT-{OID}.SPEAKER_EQ-{EID}.Q`	`Float`	Get, Set		[0.4, 30]

Register Name	Type	Access	Unit	Range
`OUT-{OID}.SPEAKER_EQ-{EID}.BYPASS`	`Boolean`	Get, Set		

5.2.12.9 Output Crossover Registers

Register Name	Type	Access	Unit	Range
`OUT-{OID}.XR.BYPASS`	`Boolean`	Get, Set		
`OUT-{OID}.XR.GAIN`	`Float`	Get, Set	dB	[-15, 15]
`OUT-{OID}.XR.LOWPASS_TYPE`	`Enum`	Get, Set		{OFF, BUT6, BUT12, BUT18, BUT24, BUT48, BES12, BES24, BES48, LR12, LR24, LR36, LR48}
`OUT-{OID}.XR.LOWPASS_FREQUENCY`	`Float`	Get, Set	Hz	[20, 20000]
`OUT-{OID}.XR.HIGHPASS_TYPE`	`Enum`	Get, Set		{OFF, BUT6, BUT12, BUT18, BUT24, BUT48, BES12, BES24, BES48, LR12, LR24, LR36, LR48}
`OUT-{OID}.XR.HIGHPASS_FREQUENCY`	`Float`	Get, Set	Hz	[20, 20000]

5.2.12.10 Output FIR

Register Name	**Type**	**Access**	**Unit**	**Range**
`OUT-{OID}.FIR.BYPASS`	`Boolean`	Get, Set		
`OUT-{OID}.FIR.TAPS`	`Integer`	Get	dB	[0, 512]

5.2.12.11 Output Routing Registers

Register Name	**Type**	**Access**	**Unit**	**Range**
`ROUT-{RID}.SRC`	`Integer`	Get		
`ROUT-{RID}.SRC_CHANNEL`	`String`	Get, Set		[`S`, `L`, `R`]
`ROUT-{RID}.GAIN`	`Float`	Get, Set	dB	[-80, 0]

5.2.12.12 Analog Volume Control Registers

Register Name	**Type**	**Access**	**Unit**	**Range**
`VC.COUNT`	`Integer`	Get		*VID* in range [1, VC.Count]
`VC-{VID}.NAME`	`String`	Get		
`VC-{VID}.VALUE`	`Float`	Get	Percent	[0, 100]

5.2.12.13 Power Management Registers

Register Name	Type	Access	Unit	Range
`SETUP.POWER.POWER_ON`	`Enum`	Get, Set		{AUDIO, AUDIO_ECO, TRIGGER, TRIGGER_ECO,
				NETWORK, AUDIO_DSP}
`SETUP.POWER.MUTE_TIME`	`Integer`	Get, Set	Sec	[0, 3600]
`SETUP.POWER.STANDBY_TIME`	`Integer`	Get, Set	Sec	[0, 3600]

5.2.12.14 GPIO Registers

Register Name	Type	Access	Unit	Range
`SETUP.GPIO.PIN2`	`Enum`	Get, Set		{OFF, STANDBY_NO, STANDBY_NC, MUTE_NO, MUTE_NC}
`SETUP.GPIO.PIN4`	`Enum`	Get, Set		{OFF, VOLUME_CONTROL}
`SETUP.GPIO.PIN5`	`Enum`	Get, Set		{OFF, VOLUME_CONTROL}
`SETUP.GPIO.PIN6`	`Enum`	Get, Set		{OFF, VOLUME_CONTROL, TRIGGER_12V_IN}
`SETUP.GPIO.PIN7`	`Enum`	Get, Set		{OFF, VOLUME_CONTROL, TRIGGER_12V_OUT}

Register Name	Type	Access	Unit	Range
`SETUP.GPIO.PIN8`	`Enum`	Get, Set		{VCC_3V3}

5.2.12.15 LAN Registers

Register Name	Type	Access	Unit	Range
`SETUP.LAN.NETWORK_MODE`	`Enum`	Get		{STATIC, DHCP}
`SETUP.LAN.IP`	`String`	Get		
`SETUP.LAN.MASK`	`String`	Get		
`SETUP.LAN.GATEWAY`	`String`	Get		
`SETUP.LAN.DNS1`	`String`	Get		
`SETUP.LAN.DNS2`	`String`	Get		

5.2.12.16 WiFi Registers

Register Name	Type	Access	Unit	Range
`SETUP.WIFI.ENABLE`	`Boolean`	Get		
`SETUP.WIFI.DISABLE_LAN_CONNECTED`	`Boolean`	Get		
`SETUP.WIFI.DISABLE_AFTER`	`Float`	Get	Sec	[0, 3600]

Register Name	Type	Access	Unit	Range
`SETUP.WIFI.MODE`	`Enum`	Get		{AP, STA}
`SETUP.WIFI.AP_SSID`	`String`	Get		
`SETUP.WIFI.AP_PASS`	`String`	Get		
`SETUP.WIFI.STA_SSID`	`String`	Get		
`SETUP.WIFI.STA_PASS`	`String`	Get		

5.2.12.17 Security Registers

Register Name	Type	Access	Unit	Range
`SYSTEM.SECURITY.PASSWORD_ENABLE`	`Boolean`	Get, Set		
`SYSTEM.SECURITY.PASSWORD_HASH`	`String`	Get, Set		

5.2.12.18 Dante Registers

Register Name	Type	Access	Unit	Range
`SYSTEM.DANTE.SOFTWARE_VERSION`	`String`	Get, Set		

Register Name	Type	Access	Unit	Range
`SYSTEM.DANTE.FIRMWARE_VERSION`	`String`	Get, Set		
`SYSTEM.DANTE.IP`	`String`	Get, Set		
`SYSTEM.DANTE.MAC`	`String`	Get, Set		
`SYSTEM.DANTE.LINK_SPEED`	`Float`	Get, Set		
`SYSTEM.DANTE.AES67_ENABLED`	`String`	Get, Set		
`SYSTEM.DANTE.DEVICE_NAME`	`String`	Get, Set		
`SYSTEM.DANTE.ENCODING`	`String`	Get, Set		
`SYSTEM.DANTE.SAMPLE_RATE`	`Float`	Get, Set		
`SYSTEM.DANTE.CLOCK_STATE`	`String`	Get, Set		
`SYSTEM.DANTE.MUTE_STATE`	`String`	Get, Set		