PPA Control Protocol (v2)

This document describes how to remote control the PPA ethernet versions with external media control units like e.g. Crestron® using UDP packets over LAN.

1. General Information

1.1. Timing

After powering on the unit, allow a delay of app. 10 sec until the first command can be received. Between two commands allow a delay of at least 50 ms, or wait for the response packet from the device.

1.2. UDP/IP via LAN

The PPAs can be addressed via **UDP Port 5001**. Reply messages from the PPAs to your controller will be sent back to the same port the controller used for sending.
The IP address can be configured in the Four Audio System Software.

There are two modes of IP configuration for the device:

- ✓ **DHCP (the default)** will try to obtain an IP from a local DHCP server, such as a standard IP router. If this fails, it will fall back to a Link Local IP address in the range 169.254.0.0/16, as specified in RFC5735 (https://tools.ietf.org/html/rfc5735).
- ✓ **Static IP** will assign a fixed IP address to the device. When choosing this option, please ensure that the subnet is correctly chosen and that there are no address conflicts in the network. Otherwise, the device may become unreachable via LAN.

1.3. Testing

You may test the communication with your PPA device using your PC and the Packetsender software http://packetsender.com/.

2. Communication

This section explains the PPA network application protocol in some detail. If you just need to know how to trigger a **Preset Recall** via network, you may skip right to the examples section (see 8).

All network communication between the PPA and the rest of the world is done with messages in the form of binary UDP packets. Each message consists of a 12 byte BasicHeader, followed by command specific information.

All numbers are sent in Little Endian byte order, i.e. the least significant bit (LSB) comes first. For example, the value 770 – or hexadecimal 0x0302 - is sent as 02 03.

2.1. BasicHeader

Each message needs to be preceded by a header in the following format:

Byte #	1	2	3	4	5	6	7	8	9	10	11	12
Meaning	Message Type	ProtocolId, always 0x01	Status		DeviceUniqueId				Message Sequence Number		ComponentId, leave 0xFE for Mastertone	Reserved (leave 0)

MessageType (1 byte)

This is the most important byte as it shows what the message is all about. The following values are interesting for media control devices:

Value (hex)	**Description**
0x00	*Ping*, can be used to check if the device is still answering. A Ping consists just of the BasicHeader and is not followed by any more data.
0x01	*LiveCmd,* used to set or query individual parameters such as input gain.
0x02	*DeviceData,* used to query or set general information, such as device name or static IP address.
0x04	*PresetRecall*, either recalls a preset on device, or queries which preset is currently active.

Status (2 bytes)

These two bytes determine the basic communication role of the message. The following values may be relevant for basic media control programming. Please remember that byte order is little endian, so 0x0002 is sent as “02 00”. Any message with an unknown Status must be ignored.

Value (hex)	**Description**
0x0002	*Command*, used for commands that actually change something on the device.
0x0006	*Request*, queries some parameters from the device.
0x0001	*Response*, sent from the device as reply to a command or request.
0x0009	*Error*, sent from the device if something has gone wrong.
0x0041	*Wait*, tells the receiver that the requested procedure is initialized but may take some time.

DeviceUniqueId (4 bytes)

Is an identifier that will be filled by your PPA device, allowing you to uniquely identify the device, e.g. to guarantee that you are talking to the right device in a DHCP setting, where the IP may change. In messages *to* the device, it can remain 0.

MessageSequenceNumber (2 bytes)

This is an identifier for your message. The PPA will always reply to the message referring to it with the same sequence number. You may start with an arbitrary number here, but it is crucial to use a *different number for each new message*. Otherwise, errors will result when trying to send several commands in quick succession, as the network failure recovery mechanisms will get confused.

ComponentId (1 byte)

This byte is used to address a specific component in multi-component devices (n/a for the PPA) Since PPA devices are not stacked, just use 0xFE to directly address the device connected to the network.

2.2. LiveCmd command

A live command follows a BasicHeader with MessageType 0x01 (=LiveCmd) and a Status of (0x0002). Most live commands are forwarded to the PPA DSP, setting parameters such as a gain value. The message consists of 16 bytes, structured as follows:

Byte #	13	14	15..24	25	26	27	28
Meaning	CrtFlags	OptFlags	Path	Value			

CrtFlags (1 byte)

The CrtFlags are used to choose variants of the command. If these flags aren't understood by the device, the command will be rejected. Two values are of use for a LiveCmd:

Value (hex)	**Description**
0x00	*Standard LiveCmd.* In this case, the value to be set is fully encoded in the 4 Value bytes.
0x01	*Contains String.* In this case, Value inside the LiveCmd structure refers to the length of a character string following the LiveCmd. This is used for submitting channel names.

OptFlags (1 byte)

The OptFlags are used to communicate additional hints to the device. If these flags aren't understood by the device, they'll just be ignored. Two values are of use for a LiveCmd:

Value (hex)	**Description**
0x00	*Execute Immediately.* The LiveCmd will be executed immediately by the device.

0x01	*Delay Recompute.* The execution of the command is delayed until another LiveCmd without this flag is set. This is useful e.g. to avoid transitional noise when changing several EQ settings at once.

Path

The Path specifies which type of value is set by the command. It may theoretically consist of up to 5 Levels, which in turn consist of a "level type" and a "level position". For PPA, only the first 3 Levels are needed so far. All unused Levels are padded with zeroes.

Here are some examples to illustrate the structure of Path, an overview of allowed values for Level Type will be given further below.

Pos[0]	**Type[0]**	**Pos[1]**	**Type[1]**	**Pos[2]**	**Type[2]**	**Pos[3]...**	**Meaning**
00	04 (Gain)	05 (input no., zero based!)	01 (Input)	00	00	00 ...	Gain on Input 6
00	0a (Delay)	00 (output no.)	02 (Output)	00	00	00 ...	Delay on Output 1
00	04 (Gain)	02 (EQ no.)	03 (Eq)	01 (input no.)	01 (Input)	00 ...	Gain of 3rd EQ on Input 2

Value

The field Value contains the value to be set. In most cases, this is a numerical value, given in a certain unit.

Value	**LevelType**	**... found where?**	**Value Type**	**Conversion / Hint**	**Unit**
Input	0x01	*Top level*	-	-	-
Output	0x02	*Top level*, Input	-	-	-
Eq	0x03	Input, Output	-	-	-
Gain	0x04	Input, Output, Eq	number	10 * g[dB] + 800	dB
Eqtype	0x05	Eq	enum	0=LP6, 1=LP12, 2=HP6, 3 =HP12, 4=Bell, 5=LS6, 6=LS12, 7=HS6, 8=HS12, 9=AP6, 10=AP12	-
Quality	0x07	Eq	number	400 * q	-

Active	0x08	Eq	bool	0 = false, 1 = true	-
Mute	0x09	Input, Output	bool	0 = false, 1 = true	-
Delay	0x0a	Input, Output	number	48000 * x	s
Phase Inversion	0x0b	Input, Output	bool	0=false, 1=true	-

2.3. DeviceData request

A DeviceData query follows a BasicHeader with a MessageType of 0x02 (=DeviceData) and a Status of 0x0006 (=request). In order to request the standard DeviceData structure described below, just fill the DeviceData command with 4 zero bytes:

Byte #	13	14	15	16
Meaning	CrtFlags (leave 0)	OptFlags (leave 0)	Reserved (leave 0)	Reserved (leave 0)

As an answer, you will receive the BasicHeader acknowledgement, followed by the following structure:

Byte #	**Meaning**
13	CrtFlags (must be 0x00)
14	OptFlags (ignore these)
15..16	Device type id (1=Mastertone rev 1, 21=Mastertone rev2)
17	Subnet Prefix length (in case of static IP)
18	Diagnostic state (should be 0 if all is fine)
19..22	Firmware Version
23..24	Serial no.
25..28	Reserved
29..32	Gateway IP
33..36	Static IP (set to 0 when DHCP is active)
37..40	Hardware Features (bitfield)
41	Start preset Id (note that this refers to the preset Id, not position!)

42..47	Reserved
48..79	Device Name (Latin 1 encoded)
80	Vendor ID
81..82	Reserved

2.4. PresetRecall command

A preset recall command follows a BasicHeader with a MessageType of 0x04 (=PresetRecall) and a Status of 0x0002 (=command). The message consists of 4 bytes, structured as follows:

Byte #	13	14	15	16
Meaning	CrtFlags	OptFlags (leave 0)	Index/Position	Reserved (leave 0)

CrtFlags (1 byte)

The CrtFlags are used to chose variants of the command. If these flags aren't understood by the device, the command will be rejected. The following values are of general use:

Value (hex)	**Description**
0x00	*Recall by Preset Index.* The number given in byte 15 is the internal index of the Preset. The index of a preset may be obtained by requesting the preset list beforehand (this requires a block transfer and is not described in this document).
0x02	*Recall by Preset Position.* The number given in byte 15 is simply the position of the Preset in the device preset list. Recommended for use in media control devices.

Index/Position (1 byte)

Specifies the preset to be recalled. The method of addressing the preset is determined by the value of CrtFlags. Note that Position is zero-based, i.e. the first preset has Position 0, the second Position 1, and so on.

2.5. Acknowledgement message

An acknowledgement is sent whenever a command has been successfully executed. It consists of the BasicHeader with the MessageType and MessageSequenceNumber of the corresponding command, and Status set to 0x0001.

2.6. Wait message

A wait message is sent by the device to acknowledge that it has received a command, but processing may take more than 500 ms. The message may be ignored by the receiver, displayed to the user, or used to extend a resend timeout.

The message consists of a BasicHeader with the MessageType and MessageSequenceNumber of the corresponding command, and Status set to 0x0041. After the header, the expected maximum time to wait is indicated by the following structure:

Byte #	13	14	15	16
Meaning	CrtFlags (should be 0)	OptFlags (ignore these)	TimeToWait (in 1/100 s)	

2.7. Error message

If for some reason the device is not able to comply to a message sent to it, it will answer with an error message. The error message consists of a BasicHeader with the Status field set to 0x0009; the fields MessageType and MessageSequenceNumber correspond to those of the message the error refers to. After the BasicHeader, the following structure informs you about details of the error:

Byte #	13	14	15	16
Meaning	ErrorCode		Reserved	Reserved

The ErrorCode takes on one of the following values:

Value (hex)	**Description**
0x0001	*Bad Request.* The message sent to it is not understood by the device.
0x0002	*Unknown Resource.* E.g. trying to recall a preset that doesn't exist.
0x0003	*Busy.* E.g. trying to access the device from two controls at once.
0x0004	*Out of Resource.* E.g. trying to upload a new preset to a device that has no free preset slots left.
0x0005	*Internal.* This code should not occur in normal operation. If it happens repeatedly, it hints at an inconsistent device state.

2.8. Example: Recall the 3rd preset

Preset positions are zero-based, so recalling the 3rd presets means means Position = 02.

To achieve this, simply send the following packet to the device via UDP port 5001. The fields that may vary for your purpose are highlighted in the table.

Byte #	1	2	3	4	5	6	7	8	9	10	11	12
Value (hex)	04	01	02	00	00	00	00	00	ee	01	fe	00
Explanation	Message Type: PresetRecall	x	Status: Command		DeviceUniqueId: leave empty				MessageSequence Number: Use a new one for each command!		ComponentId, always 0xfe for Mastertone	

Byte #	13	14	15	16
Value (hex)	02	00	02	00
Explanation	CrtFlags: Recall by Position	OptFlags	Position: 2	Reserved

Now, you should expect something like the following response from the device, sent to the UDP port from which you sent the command and from device port 5001:

Byte #	1	2	3	4	5	6	7	8	9	10	11	12
Value (hex)	04	01	01	00	6a	00	02	00	ee	01	00	00
Explanation	PresetRecall	x	Status: Response (Success)		DeviceUniqueId: Identifies the device				The same sequence number as in your command			

Here, Status is probably the most interesting field. Only status 0x0001 means that the device has fully executed the command. 0x0009 (Error) signals an error, and 0x0041 (Wait) just means that the device may take some time to process your request. As a PresetRecall may take some time, it is common to receive a Wait message before the final acknowledgement.

2.9. Example: Set gain on output 4 to -10 dB

To change an individual value, we need to send a *LiveCmd.* Again, we highlight values that will most probably vary if you want to perform a similar command.

Byte #	1	2	3	4	5	6	7	8	9	10	11	12

Value (hex)	01	01	02	00	00	00	00	00	ef	01	fe	00
Explanation	Message Type: LiveCmd	x	Status: Command		DeviceUniqueId: leave empty				MessageSequence Number: Use a new one for each command!		ComponentId, always 0xfe for Mastertone	

Byte #	13	14	15	16	17	18	19..24	25..28
Value (hex)	00	00	04	00	02	03	00 00 00 00 00 00	bc 02 00 00
Explanation	CrtFlags (standard value)	OptFlags (apply immediately)	Gain	-	Output	Pos 4	Zero pad levels 3..5	-10 [dB] * 10 + 800 = 700, or 0x2bc in hex

2. After sending this, the following response from the device will indicate that all is fine:

Value (hex)	01	01	01	00	6a	00	02	00	ef	01	00	00
Explanation	LiveCmd	x	Status: Response (Success)		DeviceUniqueId: Identifies the device				The same sequence number as in your command			

Again, evaluate the sequence number to be sure that this is actually the answer to your command, and evaluate the status to be sure that everything's okay. Status 0x0001 means that the device has fully executed the command. 0x0009 (Error) signals an error, 0x0041 (Wait) asks you for a little patience. As a simple Gain command usually won't take much time, you'll most probably receive an immediate acknowledgement. However, we consider it good style to always be prepared for an intermediate Wait message (at the very least, ignore it instead of taking it for an acknowledgement or error!), as the policy for sending Wait message may change in future implementations.

Table 1 document revision history

2021-03-29	V1.0	doc. created	RT