YAMAHA

AMX Control Module for:

MTX5-D Matrix Processor

Rev. 1

Release December 10, 2014

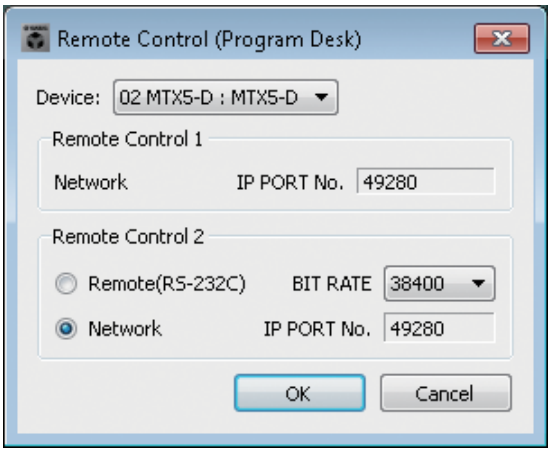
commercial audio

Introduction

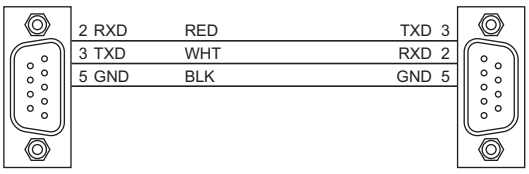
The MTX5-D can be controlled via Ethernet or RS232 using our Serial Control Protocol (SCP). The MTX5-D supports up to nine remote control connections at once.

In MTX Editor, if you go to `System > Remote Control` you will see the two types of remote connection listed. One will be a TCP/IP socket on port 49280. The other will be an RS232 port (configurable to be either 38400 baud or 115200 baud).

The RS232 pinout for the MTX5-D is Rx 2, Tx 3, Gnd 5. For an AMX system, that means pins 2 and 3 are crossed. Do not connect any other pins – it may stop the communication from flowing. The full pinout (and AMX FG number for their pre-made cables) are here:



Remote Control Connector Information
Cable FG Number: 10-756-04



9 Pin D Sub Female (AMX 41-0903 WITH AMX 41-0902 HOOD)

9 Pin D Sub Female (AMX 41-0903 WITH AMX 41-0902 HOOD)

Revision
Rev 1

Legend
GND = Ground
RXD = Receive Data
TXD = Transmit Data

Cable length
10 Feet

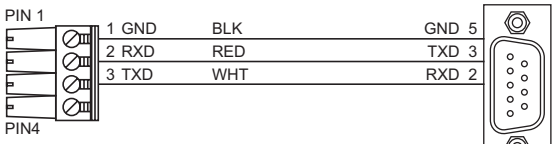
Legend
GRND = Ground
TXD = Transmit Data
RXD = Receive Data

Type of Control
RS-232

Supported AMX Devices
AXCENT3 RS232 PORT

Function Note
Customer must provide the 9 pin connector (female housing, female pins) that is supplied with the lens.
AMX doesn't currently carry this connector.

Remote Control Connector Information
Cable FG Number: 10-756-10



NETLINX Connections 4 Pin Female Phoenix (P/N AMX 41-5047)

9 Pin D Sub Female (AMX 41-0903 WITH AMX 41-0902 HOOD)

Revision
Rev 1

Legend
GND = Ground
RXD = Receive Data
TXD = Transmit Data

Connector Type
9 Pin D-Sub, Female

Type of Control
RS-232

Tips and Communication Behaviors

We highly recommend using IP control rather than RS232. Because the AMX system knows when an IP socket is dropped, the module can properly re-scan features and ensure status information is current if the connection is lost. Using RS232, if the MTX5-D is restarted or the cable is interrupted, it is more difficult to know if the connection was lost and things can fall out of sync.

Command to turn communications on: The MTX5-D will not respond on the control port until it receives a wakeup string of "`'devstatus runmode',$10`". If you are using IP, the module sends this string as soon as the IP socket opens.

When using RS232, AMX can't be sure if the product is online or offline. (Anytime the power is cycled on the MTX5-D, this string will need to be resent.) We have added a routine to automatically send the wake-up string after every minute of silent communication between the controller and MTX5-D, but if you cycle power on the MTX5-D, it may be a short while between the MTX5-D finishing boot-up and the serial port becoming available. If you prefer to send a command to the module, you can send 'WakeUp' and the module will then send the string for you.

Efficient communications model: The MTX5-D automatically sends status updates any time parameters change. For instance, if you change volume from this module, a wall controller or MTX editor, status updates are automatically generated. The same goes for presets – all status changes will be updated for you. However, you should be aware that the system will not give you a status update if you send a command that doesn't change a value. For instance, if you ask for volume at 50% and it was already at 50%, no status string will be returned.

Module Basics

This **COMM module** is an interpreter, translating a clearly understandable API to the unique command structure of the MTX5-D. For instance, to set the volume of input 5 to 50% (using scale of 0-1000):

The API accepts: **`Input:Volume=5:500`**
Converts Command to: `"'setn MTX:stx_512/60000/0/4/0/0/0 0 0 500',$0A"`
The status comes back:
`"'OK setn MTX:stx_512/60000/0/4/0/0/0 0 0 500 ',$22,'-15.57',$22,$0A"`
The API returns: **`Input:Volume=In 5:Vol 500,On 1 (On)`**

By translating command strings from raw addresses to readable strings, writing and debugging your code is dramatically simplified.

The module implements the vast majority of the controls available in the MTX5-D. "Set it and forget it" features like speaker tuning were left off, but otherwise the features set is rather complete.

The **UI module** has hard coded buttons and bargraphs for the most common features including:

- Head Amp Gain, Phantom Power and Metering
- Input and Output Volume Ramping, Target Level, Status, On/Off and Metering
- Input EQ bands 1 & 3 Ramping, Status (you can set a treble/bass EQ easily)
- Output Volume Ramping, Target Level, Status, On/Off and Metering
- Effect 1 & 2 Patch and Parameter Adjustment

Because the MTX5-D would implement more features than can be hard coded to an AMX panel's 4,000 buttons and 600 levels, we left the rest so you can implement the pieces you need. Chances are you will be updating status as your panel navigation changes, so you need to write your own UI anyway. To streamline that, the UI module has the parsing routine in a DATA_EVENT for every feature supported in the COMM module. That should reduce the amount of time required for status updates.

In general, controlling **binary values** (like an on/off or mute button) will use settings of 0, 1 or T (for toggle). The MTX does not have a native toggle command, but our COMM module tracks the status and sends the appropriate state for you. Here are some examples of how a command would be formatted:

```
HeadAmp:48V=4:0  // This turns Input 4 Phantom Power Off
HeadAmp:48V=4:1  // This turns Input 4 Phantom Power On
HeadAmp:48V=4:T  // This toggles the state of Input 4 Phantom Power
```

Most **range values** operate in a range of 0-1000 and support ramping values with + and -. The MTX does not have a native ramp command, but our COMM module tracks the status and sends the appropriate increase for you. There are some exceptions to that rule, such as the Effect Patch which only has 5 values, and some others were "ramping" commands didn't seem appropriate. Those will be noted in the protocol.

```
Output:Volume=12:500  // This sets Output 12 to 50% (scale is 0-1000)
Output:Volume=12:+    // This ramps Output 12 Volume up a step
Output:Volume=12:-    // This ramps Output 12 Volume up a step
```

The **initial status** of all features tracked by the module will be scanned when the control port comes online. For IP sockets, this works very well because this means we scan as soon as we connect. The RS232 port will scan as soon as the RS232 port comes online. So long as the MTX is powered up and ready, this works fine. If you turn the MTX off, we will need to rescan. To do this, you can use an `'Init=1'` SEND_COMMAND to the COMM module and the system will resynchronize all values.

Command/String Index Values

Input Order	
1-24	Mono Inputs 1-24
25-28	Stereo Inputs 1L-2R
29-30	SD Card Player
31-34	Effect Return 1L-2R

Input Channel Numbers	
1-24	Mono Input Channels 1-24
25-28	Stereo Input Channels 1L-2R
29-30	SD Card Player L/R
31-34	Effect Return 1L-2R

Effect Patches	
1	Reverb Hall
2	Reverb Stage
3	Karaoke Echo
4	Vocal Echo

EQ Types	
1	Parametric EQ
2	Low Shelf, 6dB/Oct
3	Low Shelf, 12dB/Oct
4	High Shelf 6dB/Oct
5	High Shelf 12dB/Oct
6	HPF (High Pass Filter)
7	LPF (Low Pass Filter)

Audio Source	
0	Disconnect
1-8	Analog Mic/Line 1-8
9-12	Analog Stereo In 1L-2R
13-14	SD Card Player
15-30	YDIF 1-16
31-46	Dante 1-16
47-62	Card Slot 1-16

Meter Groups (to Start & Stop)	
1	Analog Input Meters (Mic/Line 1-8, St In 1L-2R)
2	Input Channels, Post EQ (Inputs 1-30)
3	Input Channels, Post On (Inputs 1-30)
4	Effect Returns (Effect 1L-2R)
5	Matrix (Zone) Outputs (Zone 1-16)
6	Output Channels (Out 1-16)

Levels on vdvMeters	
1-12	Mic/Line 1-8, St In 1L-2R
21-50	Post-EQ meters for Input 1-30
51-80	Post-On meters for Input 1-30
81-84	Post-On meters for Effect Returns 1L-2R
91-106	Post-On meters for Matrix (Zone) Outputs
111-126	Post-On meters for Outputs 1-16

Set-Up/Debug Commands

Debug=<0-4>
Debug?

This sets the amount of debug info sent to dvDebug.

Debug Values	
0	No Debug Messages
1	Errors Only
2	Show API Exchanges
3	Show Raw Protocol Values
4	Show Parsing Process Messages

Debug:dvDebug=<D:P:S>
Debug:dvDebug?

By default, we send debug messages to 0:1:0 so they show up in the telnet window when you type `msg on`. If you prefer, you can send them to a virtual device to log them or to relay messages from multiple masters to a single one so they are displayed in one place.

Debug:Name=<UniqueName>
Debug:Name?

Debug strings from the module will usually come up as `YamahaMTX5D(D:P:S)` where D:P:S is replaced with the vdvCom address of this module. If you want to use a different name for debug purposes, you can assign a new name with this command. You can shorten the name or create more distinct names when multiple MTX5-D units are present such as `MTX5D-MtgRms` or `MTX5D-BallRms`.

Set:IP=<IP Address>:49280:IP_TCP
Set:RS232=Delay=<1/10 sec>:HSOFF:XOFF:SET BAUD <38400|115200>,N,8,1 485 DISABLE

Sets parameters for the module to connect to the MTX5-D.

For IP, you will want to send in the IP address and other IP settings as shown, then turn on channel 250 on vdvCom. Whenever channel 250 is on, the module tries to keep the IP socket connected. (If the connection is lost, the module will automatically attempt to re-establish the link.) If the channel is off, the module will disconnect and wait for the module to re-link.

For RS232, you effectively send the serial port initialization strings. In older processors, sometimes the init strings needed to be delayed from the online event. You can type in an appropriate delay in this module and it will take care of it for you.

Init=<0|1>

This command launches a routine to sends the control port “WakeUp” string and scans the current status of all tracked status. The routine uses a timeline to churn through all initial status and takes several minutes to complete. Sending a value of 1 cancels an existing routine and starts over. Sending a value of 0 terminates a routine in progress.

Passthru=<STRING TO SEND>

Anything after the equal sign is sent to the serial port. A Line Feed is added at the end for you.

Set:RampStep:Attack=<0-1000> // default 20
Set:RampStep:CompensationLevel=<0-1000> // default 250
Set:RampStep:Decay=<0-1000> // default 20
Set:RampStep:DigitalGain=<0-1000> // default 20
Set:RampStep:Effect=<0-1000> // default 20
Set:RampStep:EQ:Freq=<0-1000> // default 20
Set:RampStep:EQ:Gain=<0-1000> // default 20
Set:RampStep:EQ:Q=<0-1000> // default 20
Set:RampStep:Gain =<0-1000> // default 20
Set:RampStep:Hold=<0-1000> // default 20
Set:RampStep:Knee=<0-1000> // default 20
Set:RampStep:Range=<0-1000> // default 20
Set:RampStep:Ratio=<0-1000> // default 20
Set:RampStep:Release=<0-1000> // default 20
Set:RampStep:ResponseTime=<0-1000> // default 20
Set:RampStep:Threshold=<0-1000> // default 20
Set:RampStep:Volume=<0-1000> // default 20

There are no ramping commands in the raw MTX5-D protocol. This COMM module tracks the status of each parameter and sends a discrete command to raise or lower the value by these increments. If you want larger steps for these commands, you could send this in when the module comes online.

Set:StatusSyncInterval =<msec> // default 250

When the control port comes online, the module begins a timeline to scan the status of all tracked features on the MTX5-D. This adjusts the polling interval of this timeline, in case your processor requires more headroom to run other events.

SCPMode:Encoding=<UTF|ASCII>
SCPMode:KeepAlive=<Interval in msec>
SCPMode:Resolution=1000
SCPMode:Value=<1:Normalized|0:Raw>

These are used by the module to set preferred protocol values. AMX Programmers should not change these values – that could cause the module to stop working properly.

DevInfo?

DevInfo:DeviceID=<String>
DevInfo:DeviceName=<String>
DevInfo:ParameterSetVersion=<String>
DevInfo:ProductName=<String>
DevInfo:SerialNumber=<String>
DevInfo:FirmwareVersion=<String>

DevStatus:Clock?

DevStatus:Clock=<0 (unlocked)|1 (locked)|2 (not scanned)>,<44.1kHz|48kHz>

Send Commands & Status Strings

Each command is listed below with the (indented) status command that will be yielded from the command. In most events, the status message will update multiple parameters at once – the commands and returns are grouped below for easier documentation.

To query the status, put a "?" in place of the '=' sign of any command leave the value field off. The module will query the status and return a string for you.

Effect:Patch=<Proc 1-2>:<1-4|+|->
Effect:Param=<Proc 1-2>:<0-1000|+|->
    Effect=Proc <1-2>:Patch <1-4>,Param <0-1000>

HeadAmp:Gain=<In 1-8>:<0-1000|+|->
HeadAmp:48V=<In 1-8>:<0-1|T>
    HeadAmp=In <1-8>:Phantom <0 Off|1 On>,Gain <0-1000>

Input:AGC:ResponseTime=<In #>:<0-1000|+|->
Input:AGC:CompensationLevel=<In #>:<0-1000|+|->
Input:AGC:NoiseGate=<In #>:<0-1|T>
Input:AGC:On=<In #>:<0-1|T>
    Input:AGC=In <1-8>: ResponseTime <0-1000>,Level <0-1000>,
    NoiseGate <0 Off|1 On>,On <0 Off|1 On>

Input:Comp:Threshold=<In #>:<0-1000|+|->
Input:Comp:Ratio=<In #>:<0-1000|+|->
Input:Comp:Knee=<In #>:<0-1000|+|->
Input:Comp:Attack=<In #>:<0-1000|+|->
Input:Comp:Release=<In #>:<0-1000|+|->
Input:Comp:Gain=<In #>:<0-1000|+|->
Input:Comp:On=<In #>:<0-1|T>
    Input:Comp=In <1-16|25-28>:Threshold <0-1000>,Ratio <0-1000>,Knee <0-1000>,Attack <0-1000>,
    Release <0-1000>,Gain <0-1000>,On <0 Off|1 On>

Input:EQ:BandBypass=<In #>,<Band 1-3>:<0-1|T>
Input:EQ:Freq=<In #>,<Band 1-3>:<0-1000|+|->
Input:EQ:Gain=<In #>,<Band 1-3>:<0-1000|+|->
Input:EQ:On=<In #>:<0-1|T>
Input:EQ:Q=<In #>,<Band 1-3>:<0-1000|+|->
Input:EQ:Type=<In #>,<Band 1|3>:<1-7>
    Input:EQ=In <1-16|25-30>,Band <1-3>:Gain <0-1000>,Freq <0-1000>,Q <0-1000>,
    Type <1-7>, BandBypass <1 (BYPASS)|0 (ENGAGE)>

Input:EQ:On=<In #>:<0-1|T>
    Input:EQ:On=In <1-16|25-30>:On <0 Off|1 On>

Input:FBS:Dynamic:Clear=<In #>
Input:FBS:Fixed:On=<In #>:<0-1|T>
Input:FBS:Dynamic:On=<In #>:<0-1|T>
    Input:FBS=In <1-8>:FixedOn <0 Off|1 On>:DynamicOn <0 Off|1 On>

Input:Gate:Attack=<In #>:<0-1000|+|->
Input:Gate:Decay=<In #>:<0-1000|+|->
Input:Gate:Hold=<In #>:<0-1000|+|->
Input:Gate:On=<In #>:<0-1|T>
Input:Gate:Range=<In #>:<0-1000|+|->
Input:Gate:Threshold=<In #>:<0-1000|+|->
Input:Gate=In <1-16>:Threshold <0-1000>,Range <0-1000>,Attack <0-1000>,Decay <0-1000>,
Hold <0-1000>, On <0 Off|1 On>

Input:HPF:Freq=<In #>:<0-1000>
Input:HPF:On=<In #>:<0-1|T>
Input:HPF=In <1-16>:Freq <0-1000>,On <0 Off|1 On>

Input:Volume=<In #>:<0-1000|+|->
Input:On=<In #>:<0-1|T>
Input=In <1-34>:Vol <0-1000>,On <0 Off|1 On>

Input:Patch=<In #>:<Audio Source #>
Input=In <1-30>:Input <0-62>

Input:Phase=<In #>:<0-1|T>
Input:Phase=In <1-16>:<1 (INVERTED)|0 (NORMAL)>

InputDCA:Volume=<In DCA #>:<0-1000|+|->
InputDCA:Mute=<In DCA #>:<0-1|T>
InputDCA =In <1-8>:Vol <0-1000>,Mute <0 (UNMUTED)|1 (MUTED)>

MatrixOutput:Volume=<MatrixOut #>:<0-1000|+|->
MatrixOutput:On=<MatrixOut DCA #>:<0-1|T>
MatrixOutput=Out <1-16>:Vol <0-1000>,On <0 Off|1 On>

MatrixSend:Volume=<In #>,<Send #>:<0-1000|+|->
MatrixSend:On=<In #>,<Send #>:<0-1|T>
MatrixSend=In <1-34>,Send <0-16>:Vol <0-1000>,On <0 Off|1 On>

Meters:Start=<Meter Group #>,<Level|Hold>,<Refresh Interval (msec)>
Meters:Stop-<Meter Group #>
No status message – meters are sent as levels to vdvMeter. See Command/String Index Values.

Output:EQ:Freq=<Out #>,<Band 1-4>:<0-1000|+|->
Output:EQ:Gain=<Out #>,<Band 1-4>:<0-1000|+|->
Output:EQ:Q=<Out #>,<Band 1-4>:<0-1000|+|->
Output:EQ:Type=<Out #>,<Band 1-4>:<1-7>
Output:EQ:BandBypass=<Out #>,<Band 1-4>:<1-7>
Output:EQ=Out <1-16>,Band <1-4>:Gain <0-1000>, Freq <0-1000>, Q <0-1000>,
Type<1-7>, BandBypass <1 (BYPASS)|0 (ENGAGE)>

Output:EQ:On=<Out #>:<0-1|T>
Output:EQ:On=Out <1-16>:<0-1|T>

Output:Volume=<Out #>:<0-1000|+|->
Output:On=<Out #>:<0-1|T>
Output=Out <1-16>:Vol <0-1000>,On <0 Off|1 On>

OutputDCA:Volume=<Out #>:<0-1000|+|->
OutputDCA:Mute=<Out #>:<0-1|T>
OutputDCA =Out <1-8>:Vol <0-1000>,Mute <0 (UNMUTED)|1 (MUTED)>

Preset=<1-50>
Preset=<1-50>