


Stardraw Forums

Home » All Stardraw Products » Stardraw Control 2010

# Controlling Epson Projectors through Ethernet TCP/IP

<http://forums.stardraw.com/Topic3157.aspx>

 By ctms - 2/7/2008 3:31:15 AM

Hi,


We find in the attached "pdf" file that the Epson projectors that have a TCP/IP network port can be controlled through Ethernet from a computer through its software included in their package & it says that we can control it by AV control system. The same RS-232 commands that we use to control it through computer COM port by StarDraw Control can be also used to control it through TCP/IP control. May you help us in writing the required program to do it. Actually, we now need to control our new Epson projector EMP-6100 by stardraw through network directly without using any additional devices like GlobalCache or Extron IPLink.

We also need to know how to read the status of the projector (ON/OFF, source selected,....etc) & apply it on a button to show its status on the control interface.

Regards

Mohammad M.Kamel

AUC, CTMS

 By James Heliker - 2/7/2008 7:32:57 PM

Mohammad,

Does the Epson projector have a web interface? Meaning, can you login to it when it is on the network and get information about it?

If so, you'd probably find the easiest way is to emulate the web interface with some C# scripting inside of Stardraw Control.

The basic structure would be to isolate each section of the web interface, for example, if they are using PHP or CGI scripting, you'd find the long links for each function that the web interface performs, and copy them all to a text file, with a reference to what each link actually does.

Then you'd need to build a little script that formed an HTTP request, that is setup from a number of parameters in your script, just like pushing buttons in a web browser.


Then the return stream from the request would be used to get data back into Stardraw Control, and translated into something that is actually useful to the user, like a green light on the GUI, when the projector is on, for example.

Its not an extremely simple task, but easier than getting into the base protocol of the device and opening up streams of data to and fro.. 🤖

If you need help, either someone here on the forum can guide you specifically through each step, or feel free to

contact a Certified Programmer. I have a great deal of experience in this area, and would be glad to help. Email me at jheliker AT linkline DOT com.

Hope that helps,


 By ctms - 2/11/2008 6:35:40 AM

Hi,

There is no web interface for this projector. There is only a software that come included in its package. Kindly find attached file for the Epson EMP-6100 user manual & the software Epson Monitor V4.0.

I need to know if we can use the software to know the orders that sent from computer to the projector -the software can control the ON/OFF & source selection through network- to simulate them in the StarDraw control interface????

or the serial command of the projector can be sent through network to control it????

 By pclark - 2/11/2008 12:56:16 PM

**ctms (2/11/2008)**

Hi,

There is no web interface for this projector. There is only a software that come included in its package. Kindly find attached file for the Epson EMP-6100 user manual & the software Epson Monitor V4.0.

I need to know if we can use the software to know the orders that sent from computer to the projector -the software can control the ON/OFF & source selection through network- to simulate them in the StarDraw control interface????

or the serial command of the projector can be sent through network to control it????

Hi Mohammad

Thanks for attaching the Epson Communication Guide to assist with your enquiry.

[James - thanks for the suggestion of using the web interface; many product are now incorporating this method of control as an alternative]

A driver can certainly be written for this device, using either serial or TCP control.

The Communication Guide is a little ambiguous or vague in places:

1. it states that the same serial commands can be sent via a TCP connection, but does not state which TCP port to open: it refers to the "ESC/VP.net protocol manual": I've searched the Epson website, and searched Google, but am unable to find this.
2. I did find web references to TCP ports 3620 and 3629 so they may be worth trying.
3. it is not clear whether a response from the projector includes the CR line terminator and then the colon, or whether just the colon indicates the end of response.
4. the "colon from project" is also included in the sent command examples: as this is sent TO the projector it can't be FROM the projector - it must either be a typo error or may mean that the command should be prefixed with a colon.
5. it does not state whether the "step parameter" functions (eg. VOL, BRIGHT, CONTRAST) can be queried using a get command, eg. VOL?

In any case, these issues can be mostly ignored in order to provide you with a basic example.

To create a TCP driver for this device you would first create a new product, and add a port named "Ethernet" of type "TCP Input Port". Click Edit to open the script editor.

In the classes constructor you should configure the TCP port and line terminator, eg:

```
public MyClass( ProductInstance productInstance, Port port ) : base( productInstance,
port ) {

    // set Projector Control Port
    this.IPPort = 3620; // or possibly 3629 for ESC/VP.net
    this.NewLine = "\r";
}
```

Note that I have assumed TCP port 3620 only from some brief web searches:

<https://www.securetrust.com/resources/portsearch/app/epson/>

<https://www.securetrust.com/resources/portsearch/app/vp.net/>

Setting the NewLine property to "\r" indicates that all WriteLine should be appended with CR (carriage return) and OnLineIn occurs when a CR is detected.

To send commands to the projector, create the methods you require, mark them as [Controllable] and use the built-in WriteLine method to send the command string, eg:

```
[Controllable] public void Power() { WriteLine( "PWR ON" ); }
[Controllable] public void PowerOff() { WriteLine( "PWR OFF" ); }
[Controllable] public void MuteOn() { WriteLine( "MUTE ON" ); }
[Controllable] public void MuteOff() { WriteLine( "MUTE OFF" ); }
[Controllable] public void SelectInput1() { WriteLine( "SOURCE 10" ); }
[Controllable] public void SelectInput2() { WriteLine( "SOURCE 20" ); }
[Controllable] public void SelectVideo() { WriteLine( "SOURCE 40" ); }
```

Processing responses from the projector will be more involved, as it is not clear how the response is terminated.

Instead of using OnLineIn (which will only be called after CR is received) we can use OnByteIn and accumulate each byte until we see the colon character; then process the accumulated string, eg:

```
private ArrayList list = new ArrayList();

protected override void OnByteIn( byte data ) {

    base.OnByteIn( data );

    // see if terminator character received
    if( data == (byte)':' ) {

        // get the bytes in the current message
        byte[] message = list.ToArray(typeof(byte)) as byte[];
        list.Clear();

        // convert to a string and process the response
        ProcessReponse( System.Text.Encoding.ASCII.GetString( message ) );
    }
    else
        list.Add(data);
}
```

The ArrayList holds the bytes received until a colon ( : ) is detected - the list of bytes is then converted to a string and passed to a ProcessReponse method, which might look something like this:

```
[Controllable] public event EventHandler PowerOn;
[Controllable] public event EventHandler PowerStandby;
```

```
[Controllable] public event EventHandler Input1Selected;
[Controllable] public event EventHandler Input2Selected;
[Controllable] public event EventHandler VideoSelected;

private void ProcessReponse( string response ) {

    string[] responseParts = response.Split( new char[] { '=' } );

    switch( responseParts[0] ) {

        case "":
            // response from NULL command
            Logger.Error("NULL");
            break;

        case "ERR\r":
            // invalid command
            Logger.Error("ERR");
            break;

        case "PWR": {
            if( responseParts[1] == "01" )
                // power is on
                if( PowerOn != null )
                    PowerOn( this, EventArgs.Empty );
            else
                // power is in standby or other mode
                if( PowerStandby != null )
                    PowerStandby( this, EventArgs.Empty );
        }
        break;

        case "SOURCE": {
            // source has changed
            if( responseParts[1] == "10" )
                if( Input1Selected != null )
```

```
        Input1Selected( this, EventArgs.Empty );
    else if( responseParts[1] == "20" )
        if( Input2Selected != null )
            Input2Selected( this, EventArgs.Empty );
    else if( responseParts[1] == "40" )
        if( VideoSelected != null )
            VideoSelected( this, EventArgs.Empty );
    else
        Logger.Debug("Unknown SOURCE={0}", responseParts[1]);
    }
    break;

default:
    Logger.Debug("Unknown response: {0}", response);
    break;

}
}
```

The ProcessReponse method splits the response string into parts wherever an equal sign appears in the string. The switch/case statement simply processes each type of response.

In the "PWR" example, according to the protocol specification, "01" is the return code for On; in this case the PowerOn event is fired, otherwise the PowerStandby event is fired.

You can make your choice of events as broad or precise as you need, as long as the data in the response is detailed enough to accurately determine which event to fire or property to set.

Because I've made some assumptions due to the lack of clarity in the protocol document, I wouldn't be surprised if the code above does nothing at all!

However, I trust that this gives you a good idea of how to proceed, but please post any other questions you have, and anything more you find about how the project protocol actually works.

Thanks