LivePanel Network API

1/24/12

Introduction

Starting with version 1.50a3 and forward, the BlueLite X1 Live Panel exposes a network API for remote control. This document is intended to serve as a basic introduction to the protocol. In addition to this document, please consult the sample source project at:

https://github.com/LightInMotion/lPanel

If that is where you got this document, irony noted.

Get To It

To remotely control the LivePanel you need to do two things:

1. Find it on the network
2. Tell it to do something

Finding, or 'discovery', piggy backs on an existing protocol – Art-Net.

You need to open a Datagram (UDP) socket and broadcast a 'Poll' message to it. The poll message is compatible with Art-Net packets (from LPNet.h):

```
typedef struct {
     U8 ProtoID[8];          // protocol ID = "LIMSClp"
     U16 OpCode;             // == LPNET_OPCODE_POLL
     U8 VersionH;            // 0
     U8 VersionL;            // protocol version, set to LPNET_VERSION
} LPNET_POLL;
```

However, the protocol ID is not "Art-Net", so broadcasting it to Port 0x1936 should not cause any Art-Net devices to explode, unless they are not validating packets, in which case they deserve to die… But I digress. The Live Panel will answer with a Poll Reply, sent to the address and port from which it received the poll message. Again, the packet is Art-Net compatible:

```
typedef struct {
     U8 ProtoID[8];  // protocol ID = "LIMSClp"
     U16 OpCode;     // == LPNET_OPCODE_POLLREPLY
     U8 VersionH;    // 0
     U8 VersionL;    // protocol version, set to LPNET_VERSION
```

```
    U32 Address;    // IP Address
    U16 Port;       // Port for service
    U16 Flags;      // See Flags above
    U32 Info;       // When LPNET_FLAG_INUSE, IP Address of user
} LPNET_POLLREPLY;
```

It is important to note that the packet is in Big Endian ('network order'). Once you get past all the Art-Net coexist stuff at the top (protocol ID, OpCode, version), we get the 3 bits of information we care about:

- IP Address of the LivePanel (presumed to be IPv4 since we used UDP broadcast for discovery)
- Port to talk to for LivePanel operations
- A flag telling us if the LivePanel is currently available or already in use by another application

Currently, LivePanel is single client. It will reject any other connection requests while a client application is active. Though it will drop the existing client after 5 minutes of no activity (so you'll want to periodically issue commands to keep the connection alive).

If the in-use flag is set, the info field contains the IP address of the client currently using the API so you can hunt them down and berate them, etc.

Once you have the IP Address and Port, and assuming that the in-use flag is not set, you can open a Streaming (TCP, so you will probably want to set the TCP_NODELAY option) socket to the LivePanel and get to the good stuff.

The Good Stuff

Once you have a TCP socket open to the LivePanel you can start telling it what to do. With one exception, all commands to the Live Panel consist of:

- Command
- 14 bit value
- 14 bit parameter

These are packed into a 6 byte message as (from LPNet.h):

```
0xFF,
<command>,
<valH 7 bits>,
<valL 7 bits>,
```

```
<parH 7 bits>,
<parL 7 bits>
```

Since the stream is TCP, there is no reason to add extra error checking, etc. to the packet. There is also no length. The only time that the most significant bit of a byte is set is when the 0xFF is sent, so we can use that to insure packet syncronization.

If value or parameter is unused, it should be set to 0.

So, going (yet again) from LPNet.h. 8192 is the value for full on the Grand Master and LPCMD_GM is 0. So sending:

```
0xFF,
0x0,    // LPCMD_GM
0x40,   // 8192 is 0x2000, >> 7 (drop bottom 7 bits) is 0x40
0x0,    // & 0x7F (mask for bottom 7 bits) is 0
0x0,
0x0
```

Will set the Grand Master to full. The one exception is to the packet structure above is the LPCMD_CUES command (set the cues for a chaser). In this special case, the last two bytes (param) are dropped, and replaced with a 0 terminated UTF-8 ('standard C') string. The string formatted the same as what you would enter into LivePanel itself (ex. "100 200 300" or "100,200,300");

What about getting information?

One of the big improvements in the new network based APIs over the older ActiveX automation API is that the flow of information is more two directional. In the case of the LivePanel, this is pretty straightforward.

Commands that will receive a response include GET in their name. So LPCMD_GM_GET will get the current Grandmaster value. Unlike commands, responses have no real packet structure at all, just:

- Command being responded to
- Response specific data

The format of each response is covered (yet again) in LPNet.h. In the case of LPCMD_GM_GET, we would expect to receive:

```
0x02, // LPCMD_GM_GET
0x20, // High byte of 8192 (since we set full above)
0x00  // Low byte of 8192
```

Hopefully, that is enough to get you going. But always feel free to contact us at www.limsc.co.