Autor: Fabian Sperk, Samed Beqari

RFC 666

(LCP ..Led Control Protocol)

Introduction

The objective of LCP is to control one or an array of eurolite LED PAR-64 HCL LED spotlights. LCP, though usable directly by a user at a terminal, is designed mainly for use by programs.

This paper assumes knowledge of the following protocols described in the ARPA Internet Protocol Handbook.

The Transmission Control Protocol

Terminology

LCP

**Spot**    
The Spot defines each LED in the Lamp Table and each Spot has 3 Features.

**TCP**   
Protocol which is used for the Client-Server connection.   
   
**Feature**   
The Feature defines the colour of the Spot.

**Data connection**   
A simple connection over which data is transferred. The data transferred will be the spot number and the feature number. The path will be between a user and a server.

**Data port**   
The passive data transfer process "listens" on the data port for a connection from the active transfer process in order to open the data connection.

**error recovery**   
A procedure that allows a user to recover from certain errors such as failure of either Host system or transfer process. Is handled over TCP.    
 

# Changelog

|  |
| --- |
| **Change** |
| Changed open port from 666 to 8000 |
| Change config file. The server does not write it, he just reads it in to learn about available spots.   * Config file is called DMX\_CONFIG.txt and resides in same folder as server executable |
| Changed server feature value retention.   * Previously the server was not expected to keep track of the values that features were set to. * Now he does. |
| Changed message format   * Not all messages have the same format. * Some are of fixed-, some are of variable length. |
| Changed data representation   * Now data and feature value is also saved. * Both server and client hold this information in separate tables. |
| Changed commands completely:   * Handshake now involves the server sending information about available spots and features directly. (the client does not need to query the server for that info any more). * Request Spot Information and Request Feature Information were removed. * Set feature value and get feature value were added * Command to disconnect from server was removed. |
| Change minimum implementation specifics to match new specifics. (new set of commands) |
| Rewrote “Sequence of Commands and Replies” and “typical scenarios” sections to match the new implementation. |

Communication Model

System Overview

The below model shows how the communication in LCP may be implemented.



Server:

* The server has a TCP socket waiting for connections on Port 8000.
* The server is the bridge between one or more eurolite LED PAR-64 HCL LED spotlights and the client. The server learns about the available spotlights, by reading a config file.
* The server allows the client to request information about available spotlights and their features.
* The server does keep track of set values for the spotlight features throughout the connection. If the connection is lost, the Server will keep the spotlights in the current configuration and wait for a reconnect.

Client:

* Only one client can connect to the server at the same time.
* The client can request information about available spotlights and their features from the server.

Data Representation

Both, the server and the client hold a table of variable length, that is made up of spotStruct2\_t type elements.

* The length of the table depends on the number of available spots. (1 spot = 1 table entry)

The spotStruct2\_t type groups the following information:

* spotIndex: DMX index of spot.
* featureCount: number of available features for this spot
* featureArray: pointer to fist position of an array that is of -featureCount- size and holds information on the values of all the features of the spot.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Establishing Connection

Is handled by TCP. Connections are expected at Port 8000.

Error Recovery & Restart

There is no provision for detecting bits lost or scrambled in data transfer; this level of error control is handled by the TCP.

|  |
| --- |
| **NOTE:**   * all commands and replies are sent as plaintext strings. * all command end with an ‘\0’ (string delimiter char) |

Commands

**Handshake with Server (0x01)**

* Can be used to get the system into a known state. (all lights off/on)

**Get Feature Value (0x02)**

* With this command the client is able to learn about feature values from the server.

**Set Feature Value (0x02)**

* With this command the client tells the server to set the value of a given feature.

Replies (incl. Error Codes)

**General structure:**

* For all replies, the message starts with an echo of the command sent from the client followed by an error code.
  + Possible error-codes are:

|  |  |
| --- | --- |
| **Code** | **Type** |
| 0x01 | All Good (no error) |
| 0xEA | Command unknown |
| 0xEB | Argument unkonw |
| 0xFF | All went wrong. (in case an error occurred but none of the above error-codes applies) |

**Specific part:**

**Handshake reply**

|  |  |
| --- | --- |
| General format: | [echo] [err-code] [No Spot-Feature Pairs] [Spot0] [Features-Spot0] [..] [..] |
| Example | [0x01] [0xBF] [2] [1] [24] [25] [24]  //2 spots are available -> at indexes 1 and 25; each spots holds 24 features |

**Get feature value reply:**

|  |  |
| --- | --- |
| General format: | [echo] [err-code] [Spot-Index] [Feature-Index] [Feature-Value] |
| Example | [0x02] [0xBF] [1] [3] [50]  //Feature 3 of spot at index 1 has a value of 50 |

**Set feature value reply:**

|  |  |
| --- | --- |
| General format: | [echo] [err-code] |
| Example | [0x01] [0xBF]  //requested feature was set succefully |

Declarative Specification

Minimum Implementation

In order to make the LCP workable without needless error messages, the following minimum implementation is required for all servers:

* Commands/Replies:
* Perform Handshake + allow for initialization of features (all on, all off, leave as is)
* Get requested feature value
* Set requested feature value

Connection Handling

Is handled by TCP. Connections are expected at Port 8000.

In case of a lost communication (Link terminated unexpectedly), the server waits for a reconnection from the client.

Commands & Replies

**Handshake with Server (0x01)**

* Can be used to get the system into a known state. (all lights off/on)

|  |  |
| --- | --- |
| Syntax | [0x01] [0x01, 0x02, 0x03] |
| Parameters | 0x01 … turn all lights off  0x02 … turn all lights on  0x03 … do nothing, just reply with highest spot index |
| Server Reply | [0x01] [err-code] [No Spot-Feature Pairs] [Spot0] [Features-Spot0] [..] [..] |

**Get feature value (0x02)**

* With this command the client is able to learn about feature values from the server.

|  |  |
| --- | --- |
| Syntax | [0x02] [spot-index] [feature-index] |
| Parameters | [spot-index] ..index of spot that we want to access  [feature-index] .. index of feature that we want to know the value of |
| Server Reply | [0x02] [err-code] [Spot-Index] [Feature-Index] [Feature-Value] |

**Set feature value (0x03)**

* With this command the client tells the server to set the value of a given feature.

|  |  |
| --- | --- |
| Syntax | [0x03] [spot-index] [feature-index] [value] |
| Parameters | [spot-index] ..index of spot that we want to access  [feature-index] .. index of feature that we want to set the value for  [value] ..value we would like to set the feature to |
| Server Reply | [0x03] [err-code] |

Sequence of Commands and Replies

A typical sequence of commands would be for the client to perform a handshake with the server, leaving all lights off. During the handshake procedure the client learns about the available spots and their features. He is then able to request information on the value of a given feature of any available spot, or set it to a value of his liking.

Typical Scenarios

Scenario 1 (client connects + turns all lights of; he then lights up main red channel)

The following may be a typical scenario. A client connects to the server issuing him to turn all lights off.

After he has learned about the available spots during the handshake procedure, he now knows that there is 1 spot available.

He goes on to turn on the main red channel.

 ’---->’ represents commands from Client to Server, and ’<----’ represents replies from Server to Client.

|  |  |
| --- | --- |
| What’s happening | Commands |
| Handshake      Set feature 1 of spot 1 to 255  (main brightness)     Set feature 3 of spot 1 to 255  (main red channel) | ----> 0x01 0x01 (turn all lights off)  <---- 0xF1 0xBF 0x01 0x01 018 (ACK 1 spot available; spotIndex = 1; 24 features available)    ----> 0x03 0x01 0x01 0xFF  <---- 0x03 0xBF    ----> 0x03 0x01 0x03 0xFF  <---- 0x03 0xBF |

Scenario 2 (Connection is lost; relearn feature values)

If the connection is lost (link in TCP/IP Path failed). The server leaves the lights in the current state and retains the feature value information. A client can reconnect and learn about the values by using the -get feature value- command.

|  |
| --- |
| NOTE:  Before running scenario 2, it is recommended that you run scenario 1 and terminate the client.  In this way, scenario 2 will play out exactly as described. |

|  |  |
| --- | --- |
| What’s happening | Commands |
| Handshake      get value of feature 1 of spot 1  (main brightness)     get value of feature 3 of spot 1  (main red channel) | ----> 0x01 0x03 (leave lights as is)  <---- 0xF1 0xBF 0x01 0x01 018 (ACK 1 spot available; spotIndex = 1; 24 features available)    ----> 0x02 0x01 0x01  <---- 0x02 0xBF 0x01 0x01 0xFF (spot 1; feature 1 has value of 255)  ----> 0x03 0x01 0x03 0xFF  <---- 0x02 0xBF 0x01 0x03 0xFF (spot 1; feature 3 has value of 200) |