NetLinx Module Interface Specification

for a

Christie Digital
LW400 Projector

Also works with LX501/LW401/LWU421/
LX601i/LW551i/LWU501i

TABLE OF CONTENTS

Revision History

Date	Initials	Comments
3-14-2010	DC	Initial release
6-4-2013	DC	IP control – additional models – other minor updates

Introduction

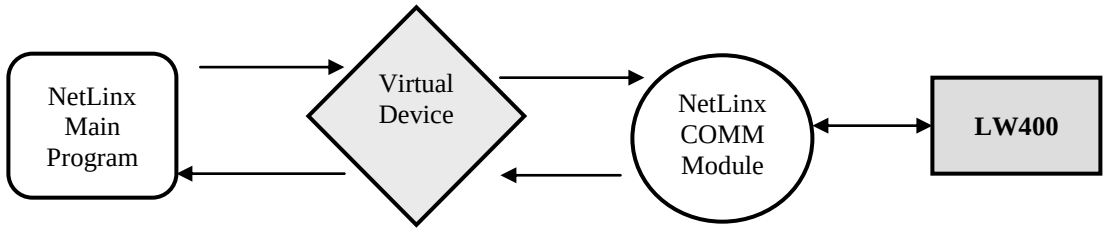
This is a reference manual to describe the interface provided between an AMX NetLinx system and a Christie Digital LW400 projector. The required communication settings are a baud rate of 19200, 8 data bits, 1 stop bit, no parity, and handshaking off.

Overview

The COMM module translates between the standard interface described below and the **LW400** serial protocol. It parses the buffer for responses from the LW400, sends strings to control the LW400, and receives commands from the UI module or telnet sessions.

A User Interface (UI) module is also provided. This module uses the standard interface described below and parses the string responses for feedback.

The following diagram gives a graphical view of the interface between the interface code and the NetLinx module.



Implementation

To interface to the LW400 module, the programmer must perform the following steps:

1. Define the device ID for the LW400 that will be controlled.
2. Define the virtual device ID that the LW400 COMM module will use to communicate with the main program and User Interface. NetLinx virtual devices start with device number 33001.
3. The NetLinx LW400 module must be included in the program with a DEFINE_MODULE command. This command starts execution of the module and passes in the following key information: the device ID of the LW400 to be controlled, and the virtual device ID for communicating to the main program.

An example of how to do this is shown below.

```
DEFINE_DEVICE
   dvLW400          = 5001:1:0     // The LW400 connected to the NetLinx on 1st RS-232 port
  OR
      dvLW400       = 0:3:0        // The P_SERIES connected to the NetLinx via IP

   dvLW400 TPl      = 10001:1:0    // The touch panel used for output

   vdvLW400         = 33001:1:0    // The virtual device use for communication between the
                                   // Comm module interface and User_Interface (UI) module interface

DEFINE_START      // Place define_module calls to the very end of the define_start section.
// Comm module
DEFINE_MODULE 'Christie Digital_LW400_Comm' mdlLW400_APP(vdvLW400, dvLW400)
```

Upon initialization the AMX Comm module will communicate with the Christie Digital LW400 and information will be exchanged.

Channels

The channels supported by the COMM module are listed below. These channels are associated with the virtual device(s) and are independent of the channels associated with the touch panel device.

Note: An '*' indicates an extension to the standard API.

Channel	Description
251	This channel is used for feedback only. ON: Device is online. OFF: Device is offline.
255	This channel is used for feedback only. ON: POWER is ON. OFF: POWER is OFF

Levels

The Main Program controls the video projector via level events (NetLinx command *send_level*) sent to the COMM module. The levels supported by the COMM module are listed below. These levels are

associated with the virtual device(s) and are independent of the levels associated with the touch panel device.

Level	Description
1	Lamp-Hours used

Command Interface

The UI module controls the LW400 via command events (NetLinx command *send_command*) sent to the COMM module. The commands supported by the COMM module are listed below.

NOTE: AMX has not defined a standard API for LW400 devices.

Command	Description
ADDRESS=<value>	Sets the IP address for the projector <value>: valid IP address ADDRESS=192.168.1.200
ASPECT=<value>	Sets the current aspect ratio. <value>: 1 = Normal (All modes) 2 = Wide (All modes) ASPECT=2: Turn aspect mode to Wide.
ASPECT?	Request for current aspect ratio. ASPECT?
AUTO=<value>	Auto-adjust PC image <value>: 1 = Execute Auto-Adjust
DEBUG=<state>	Set the debug state <state> : 0 OFF 1 ON DEBUG=1
DEBUG?	Request the debug state. DEBUG?
INPUT=<value>	Selects the input source. <value> : 1 = HDMI 2 = RGB1 3 = RGB2 4 = VIDEO 5 = S-VIDEO 6 = COMPONENT INPUT=3
INPUT?	Request the current input setting. INPUT?

PASSTHRU=<string>	Allows user the capability of sending commands directly to the LW400 without processing by the NetLinx module. User must be aware of the protocol implemented by the unit to use this command. This gives the user access to features which may not be directly supported by the module. See "Adding Functions to Modules" section at the end of this document for more information. The COMM module **automatically adds the required checksum to the command string.** <string> : string to send to unit PASSTHRU=RVER
POWER=<state>	Sets the power state. <state> : 0 = off 1 = on POWER=0
POWER?	Request the current power setting. POWER?
REINIT=<value>	Reinitialize communications with the projector <value> : 1 = re-establish IP communication REINIT=1
VERSION?	Request the current version number of the NetLinx module. VERSION?

Table 1 – Send Command Definitions

String Feedback

The NetLinx COMM module provides feedback to the User Interface module for **LW400** changes via string events. The strings supported are listed below.

String	Description
ASPECT=<value>	Reports the current aspect ratio. <value>: 1 = Normal (All modes) 2 = Wide (All modes) ASPECT=2:
INPUT=<value>	Reports the input source. <value> : 1 = HDMI 2 = RGB1 3 = RGB2 4 = VIDEO 5 = S-VIDEO 6 = COMPONENT INPUT=3
LAMPTIME=<value>	Reports lamp usage in hours. (Updated every 5 minutes). Also available as level 1.
POWER=<value>	Reports the power status. <value> : 0 power is off 1 power is on 2 cooldown
VERSION=<version>	Reports the current version number of the NetLinx module. <value> : current version number in xx.yy format VERSION=1.06

Table 2 - String Feedback Definitions

Device Notes

Commands implemented by this interface include those that are commonly used and shared between various LW400 devices. Several of the LW400 commands have been omitted and may be executed using the PASSTHRU command.

A periodic poll will return the power state and the total number of hours used by the device. Should the LW400 shift into sleep mode, it will be automatically shifted back into power-on mode.

Programming Notes

The COMM module imposes synchronous operation i.e. the COMM will not send a command to the LW400 until it has gotten a response or a timeout from the previous command.

A queue is implemented so that the commands from can be stacked. Commands are pulled off the queue (1) when a reply is received or, (2) after the reply timer times out (200 milliseconds). The device is polled every 10 seconds to return the power state and the input selected.

The Request commands return the state that is currently stored in the COMM module. They do NOT send a new Request to the device. The COMM module states are updated either by the periodic poll or by the enforced feedback. When a state changes, the UI is automatically informed.

Adding Functions to Modules

Commands to the device

This module supplies a mechanism to allow additional device features to be added to software using the module. This is the PASSTHRU command, which allows protocol strings to be passed through the module. The device-specific protocol must be known in order to use this feature.

As an example, suppose that a module for a projector has not implemented the 'white balance adjustment' feature. The command that the projector protocol requires is 03H, 10H, 05H, 14H, followed by a checksum. The documentation for the PASSTHRU command specifies that the module will automatically generate the checksum. In this case, the following string should be sent from the UI code to implement 'white balance adjustment'.

```
send_command vdvDevice,'"PASSTHRU=',$03,$10,$05,$14"
```

The reason to use PASSTHRU instead of sending a protocol string directly to the device port is that the device may require command queuing, calculation of checksums, or other internal processing, which would not be done if the string was sent directly. Because of this, it is best to filter all communication TO

the device through the module. (The module documentation will indicate any processing that will be automatically done to the PASSTHRU string like checksum calculation.)

Additional Feedback from the device

The module documentation indicates what feedback is provided. If additional feedback is required, a CREATE_BUFFER for the device must be implemented in the user code to process the strings from the device manually. Note that the module will still be processing the response strings independently and sending the interpreted feedback up to the user code.