NetLinx Module Interface Specification

for

Extron IN1508 Switcher/Scaler

Professional Control Solutions LLC • 3804 Parchment Circle • Richmond, VA • 23233-7034
804-364-3809 • PCS_LLC@comcast.net

TABLE OF CONTENTS

Introduction 3
Overview 3
Implementation 3
Port Mapping 4
Channels 5
Levels 5
Command Control 6
Command Feedback 7
Device Notes 7
Programming Notes 7
Adding Functions to Modules 8
  Commands to the device 8
  Responses from the device 8

Date	Initials	Version	Comments
06-05-2008	DC	V1.0.0	Initial release
09-22-2009	DC	V1.1.0	Minor changes to queue handling

Introduction

This is a reference manual to describe the interface provided between an AMX NetLinx system and an Extron IN1508 Switcher/Scaler**.** The Extron IN1508 Switcher/Scaler supports an RS-232 serial protocol. The required serial communication settings are a baud rate of 9600**,** 8 data bits, 1 stop bit, no parity, and handshaking off**.** The cable for this device is FG#10-752. The wiring diagram for this cable is as follows

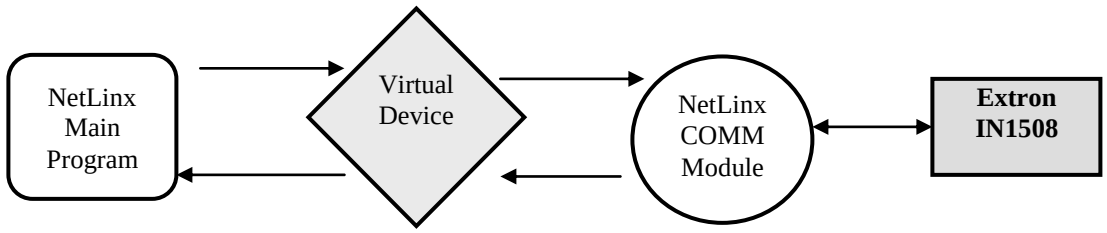
AMX NXI	AMX NI		Extron Crosspoint (Male DB9)
(Gnd) 1	(Gnd) 5	_____	(Gnd) 5
(Rx) 2	(Rx) 2	_____	(Tx) 2
(Tx) 3	(Tx) 3	_____	(Rx) 3

This module package contains only the COMM module. There are no Main or UI .axi files provided.

Overview

The COMM module translates between the standard interface described below and the switcher serial protocol. It parses the buffer for responses from the switcher, sends strings to control the switcher, and receives commands from the main program.

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



Some functionality in the device interface may not be implemented in the API interface. In cases where device functions are desired but not API-supported, the PASSTHRU command may be used to send any and all device-protocol commands to the device. See the PASSTHRU command and the Adding Functions to Modules section for more information.

Implementation

To interface to the AMX Extron IN1508 Switcher/Scaler module, the programmer must perform the following steps:

1. Define the device ID for the switcher that will be controlled.

2. Define the virtual device ID that the Extron IN1508 Switcher/Scaler COMM module will use to communicate with the main program and User Interface.
3. The NetLinx Extron_IN1508_Comm 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 switcher to be controlled, and the virtual device ID for communicating to the main program.

For an example of how to do this, see below:

```
DEFINE_DEVICE
dvSwt = 5001:1:0                // serial control
vdvSwt = 33001:1:0
dvTP = 10001:1:0
define_module 'Extron_in1508_Comm mSwtDev (vdvSwt, dvSwt)
```

Upon initialization the AMX Comm module will communicate with the switcher and information will be exchanged.

Port Mapping

This module uses multiple virtual devices in order distinguish events for one zone from another.

Virtual Device	Channels	Levels	Control	Feedback
33001:1:0	All Channels	All Levels	All Control Cmds	All Feedback Cmds

Table 1 - Port Mapping

Channels

The main program controls the switcher via channel events (NetLinx commands *pulse, on, and off*) sent to the COMM module. The channels supported by the COMM module are listed below. These channels are associated with the virtual device and is independent of the channels associated with the touch panel device.

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

Channel	Description
24	PULSE: Ramp Volume Up 1db
25	PULSE: Ramp Volume Down 1db
26	PULSE: Cycle Volume Mute
199	ON: Set Volume Mute On – used for feedback also OFF: Set Volume Mute Off
251	ON: Device is Online – used for feedback only OFF: Device is not Online

Table 2 - Virtual Device Channel Events

Levels

The main program controls the switcher via level events (NetLinx command *send_level*) sent to the COMM module. The level supported by the COMM module is listed below. This level is associated with the virtual device and is independent of the levels associated with the touch panel device.

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

Level	Description
1	Volume Level (range 0…255)

Table 3 - Virtual Device Level Events

Command Control

The main program the switcher via command events (NetLinx command *send_command*) sent to the COMM module. The commands supported by the COMM module are listed below.

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

Command	Description
SWITCH=<input>	Switch the input to the output. <input> : 0 = disconnect input channel 1..8 = input channel SWITCH=1
DEBUG?	Query the state of the debug feature. DEBUG?
DEBUG=<value>	Set the state of debugging messages in the Comm. module. **Note:** See Programming Notes section. <value> : 0 = set all messages off 1 = set all messages on DEBUG=1
PASSTHRU=<string>	Allows user the capability of sending commands directly to whatever unit is attached with minimal processing by the Duet module. User must be aware of the protocol implemented by the unit to use this command. This gives the user access to features that may not be directly supported by the module. For more information, see the "Adding Functions to Modules" section below. **Note:** The terminating characters should not be sent. The module will append them. <string> : string to send to unit PASSTHRU=2*1B (Video Mute)
SWITCH?	Query for the input channel that is currently connected to the output channel. SWITCH?
VERSION	Query for the current version number of the Duet module. VERSION?

Table 4 – Send Command Definitions

Command Feedback

The COMM module provides feedback to the main program for switcher changes via string events. The strings supported are listed below.

PLEASE NOTE: Feedback is only provided when there is a state change. If no state change resulted from the command sent in, then no feedback will be returned.

String	Description
DEBUG=<value>	Returns the state of debugging messages in the UI module and the Comm. module. <value> : 0 = all messages off 1 = set all messages on DEBUG=1
SWITCH=<input>	Reports the input channel that is currently connected to the output channel. This string is sent in response to the CL= command being successfully executed or in response to the SWITCH= command. <input> : 0 = no input channel connected 1..8 = input channel SWITCH=1
VERSION=<version>	Reports the version number of the module. <version> : x.y.z = module version number VERSION=1.0.0

Table 5 - Command Feedback Definitions

Device Notes

- The supported device is the Extron IN1508.

Programming Notes

- By default the number of inputs is 8 and the number of outputs is 1.
- This module de-queues commands to the device at a regular interval of 200 milliseconds and also de-queues when valid response is received.

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=‘ command like checksum calculation.)

Responses from the device

The module will automatically interpret replies from the device and pass these on to the application code according to the documented API. Some device replies may not be passed on to the application code.