NetLinx Module Interface Specification

for an

Extron AVT100 Tuner

TABLE OF CONTENTS

LIST OF TABLES

Revision History

Date	Initials	Comments
07/18/05	DC	Initial Release

Introduction

The Extron AVT100 is a TV tuner. The AMX module communicates to the Extron AVT100 using the following RS232 specifications:

Rate = 9600 bps
Data bits = 8
Stop bit = 1
Parity = none
Handshaking = off

For the NXI integrated controller, the communication cable is a DB-9 Female.

AMX	Extron AVT100
1 GND	5 GND
2 RXD	3 TXD
3 TXD	2 RXD

For NI systems, the standard AMX programming cable (crossover) should be used.

The communication module is called by adding the following line of code:
DEFINE_MODULE Extron_AVT100_Comm' label(virtual_device_name, real_device_name).

The UI module is called by adding the following line of code:
DEFINE_MODULE 'Extron_AVT100_UI' label (virtual_device_name, touch_panel_array, touch_button_array, display_array)

Overview

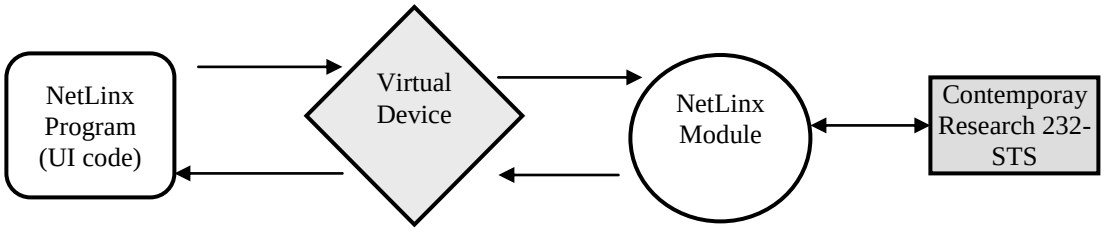
This document will define the common NetLinx module interface for a Extron AVT100 TV tuner.

The Extron AVT100 may be serially controlled using the Extron_AVT100_Comm.tko module included. This module requires a single serial port connection from a NetLinx serial port to the Extron AVT100 being controlled. For installations containing more than one Extron AVT100, multiple instantiations of the module with multiple physical RS-232 ports may be used. The communication module implements the actual Extron AVT100 protocol for communicating to the unit but exposes a more simplified, NetLinx-friendly protocol to the programmer.

The user interface module, called Extron_AVT100_UI.axs, contains the simplified protocol. It is through this user interface module the programmer takes control of the device by sending commands to the communication module, which in turn translates the commands into device specific protocol and sends them to the physical device. There is no direct connection between the user interface module and the physical device.

The programmer will interface to the NetLinx module through a virtual device. This virtual device is defined by the programmer and is used to control the communication between the user interface module and the communication module.

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



Command Interface

The interface code will control the Extron AVT100 TV tuner via command events (NetLinx command *send_command)*. These commands will be sent to the module to affect control. Below are the commands supported.

* Indicates extended commands beyond the generic API.

Command	Description
*AUDIO_MODE=<mode>	Switch to the desired audio mode. <mode> = 0 : MONO/MONO 1 : STEREO AUDIO_MODE=1
*AUDIO_MODE?	Query for the state of the audio mode. AUDIO_MODE?
*AUDIO_MUTE=<value>	Turn audio on or off. <value> = 0 : Off 1 : On AUDIO_MUTE=1
*AUDIO_MUTE?	Query for the current audio mute state. AUDIO_MUTE?
DEBUG=<value>	Enable/disable the debug mode. Debug is used to view the strings being sent out by the user interface module and the communication module and to view the incoming string from the device to the communication module and from the communication module to the user interface module. To view the messages, open a TELNET session and turn messaging on; then send the debug command as follows: <value> = 1 : (turn debug on) 0 : (turn debug off) DEBUG=1
DEBUG?	Query for the state of the telnet debugging messages. DEBUG?

PASSTHRU=<string>	Allow user the capability of sending commands directly to whatever unit is attached 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. A carriage return should be included for the command to be sent out. <string> : string to send to unit **Note:** The delimiter 'carriage return' is appended by the Comm module. Refer to the "Adding Functions to Modules" section for additional information. PASSTHRU=THIS IS A COMMAND PASSTHRU=RESET
TUNE=<value>	Increments, decrements, or sets the tuner channel. <value>: + = increment tuner - = decrement tuner n = channel, range: 1-127 P = previous channel TUNE=+ TUNE=92
TUNE?	Query the current tuner channel. TUNE?
*TUNER_MODE=<mode>	Switch to the desired tuner mode. (see programming notes) <mode> = 1 : IRC 2 : HRC 3 : STANDARD TUNER_MODE=3
*TUNER_MODE?	Query for the state of the tuner mode. TUNER_MODE?
VERSION?	Query for the current version number of the NetLinx module. VERSION?
*VIDEO_MUTE=<value>	Turn video on or off. <value> = 0 : Off 1 : On VIDEO_MUTE=1

*VIDEO_MUTE?	Query for the current video mute state. VIDEO_MUTE?

Table 1 – Send Command Definitions

String Feedback

The NetLinx module will provide feedback to the interface code for the Extron AVT100 TV tuner changes via string events. Below are the strings supported.

String	Description
AUDIO_MODE=<mode>	Feedback on the state of audio mode. <mode> = 0 : MONO/MONO 1 : STEREO AUDIO_MODE=1
AUDIO_MUTE=<value>	Feedback on the state of audio mute. <value> = 0 : Off 1 : On AUDIO_MUTE=1
DEBUG=<value>	Feedback on the current setting of the debug messages. <value> = 0 : Debug messages are off 1 : Debug messages are on DEBUG=1
ERROR=<msg>	Feedback on errors that occur in the Comm module and in the device. <msg> : string ERROR=Invalid command.
TUNE=<value>	Feedback on the tuner channel. <value>: n = channel, range: 1-127 TUNE=92
TUNER_MODE=<mode>	Feedback on the tuner mode. (see programming notes) <mode> = 0 : BROADCAST 1 : IRC 2 : HRC 3 : STANDARD TUNER_MODE=3

VERSION=<value>	Report the current version number of the NetLinx module. <value> : current version number in xx.yy format VERSION=1.42
VIDEO_MUTE=<value>	Feedback on the state of video mute. <value> = 0 : Off 1 : On VIDEO_MUTE=1

Table 2 - String Feedback Definitions

Device Notes

1. The highest channel number that can be tuned is 125. (This module only supports NTSC).
2. The device baud rate must be set to 9600 in order for the Comm module to work.

Programming Notes

The AVT100 has a dip-switch that determines the base tuner mode. If it is set to Broadcast, the "TUNER_MODE?" command will return a 0 and all TUNER_MODE= commands will be ignored, and will return a status of 0.

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_string 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.