NetLinx Module Interface Specification

for a

Zandar MX4/16 Video Processor

TABLE OF CONTENTS

Revision History

Date	Initials	Comments
11-01-04	DC	Initial release
05-20-05	DC	Default baud rate changed to 57600

Introduction

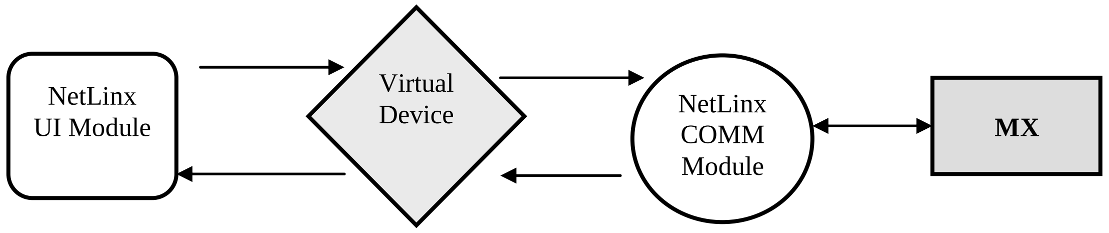
This is a reference manual to describe the interface provided between an AMX NetLinx system and a Zandar MX Video Processor. The required communication settings are a baud rate of 57600, 8 data bits, 1 stop bit, no parity, and handshaking off.

Overview

The COMM module translates between the standard interface described below and the **Zandar MX** serial protocol. It parses the buffer for responses from the MX, sends strings to control the MX, 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 UI module and Touch Panel file do not include all of the features supported by the Comm module. These files are for demonstration purposes, and additional coding will need to be done to provide a complete system.

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



Implementation

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

1. Define the device ID for the MX that will be controlled.
2. Define the virtual device ID that the MX COMM module will use to communicate with the main program and User Interface. NetLinx virtual devices start with device number 33001.
3. If a touch panel interface is desired, a touch panel file MX,REV0.tpd and module (Zandar_MX_UI.axs) have been created for testing.
4. The NetLinx MX 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 MX 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
   dvMX        = 5001:1:0      // The MX connected to the NetLinx on 1st RS-232 port
   dvMX TPl    = 10001:1:0    // The touch panel used for output

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

DEFINE_VARIABLE
//Define arrays of button channels used on your own touch panel
integer nTP_BUTTONS[]={1,2,3,4,5,6,7}

DEFINE_START       // Place define_module calls to the very end of the define_start section.
// Comm module
DEFINE_MODULE 'Zandar_MX_Comm' mdlMX_APP(dvMX, vdvMX)

// Touch panel module
DEFINE_MODULE 'Zandar_MX_UI' mdlMX_APP(vdvMX, dvTPArray, nTPButtons, nTuneButtons)
```

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

Command Interface

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

On commands that use multiple arguments, the delimiter will be the colon (:).

NOTE: Commands proceeded by a star (*) do not cause any immediate changes to the output of the MX4/16. They store values in the comm module that are used with other commands. For example, the BORDERFORMAT command sets the color that is turned on or off by the BORDER command.

Command	Description
BORDER=<value>	Turn the borders around the pip windows On or Off. <value>: 0 = Border OFF 1 = Border ON BORDER=1 Turns on borders with the previously defined color.
BORDER?	Request for current border color. Value returned from table listed in the BORDERFORMAT command. BORDER?
CAPTION=<value>	Turn the caption On or Off. <value>: 0 = Caption OFF 1 = Caption ON CAPTION=1 Turns on previously defined caption.
CAPTION?	Request for the current CAPTION setting. CAPTION?
CLOCK=<value>	Turn the CLOCK On or Off. <value>: 0 = CLOCK OFF 1 = CLOCK ON CLOCK=1 Turns on previously defined clock settings.
CLOCK?	Request for the current CLOCK setting. CLOCK?
COLORBAR=<value>	Turn the Colorbar screen On or Off. <value>: 0 = COLORBAR OFF 1 = COLORBAR ON COLORBAR=1 Turns Colorbar screen.
COLORBAR?	Request for the current Colorbar setting. COLORBAR?

DEBUG=<value>	Set the debug state <value>: 0 = OFF 1 = ON DEBUG=1
DEBUG?	Request the debug state. DEBUG?
DISPLAY:<value>	Set the Display pattern. (MX16 Only according to Zandar documentation) ID Layout Channels <value>: 1 4x4 1-16 2 12+1 1-13 3 12+1 2-14 4 12+1 3-15 5 12+1 4-16 6 3x3 1-9 7 3x3 4-12 8 3x3 7-15 9 2x2 1-4 10 2x2 5-8 11 2x2 9-12 12 2x2 13-16 13 PRST1 1-10 14 PRST2 1-8 15 PRST3 1-6 16 PRST4 1-1 For MX-4, ID 1 shows a 2x2 layout DISPLAY:9
DISPLAY?	Request for current Display pattern. (See table above). DISPLAY?
FADE=<value>	Set the FADE value. <value>: 0 = Normal Image 1 = Faded to Black FADE=1
FADE?	Request for current FADE setting. FADE?
FREEZE=<value>:<pip>	Freeze mode for a specific window. <value>: 0 = Unfreeze 1 = Freeze <pip>: 1-16 = Pip window to freeze/unfreeze FREEZE=1:4

FULLSCREEN=<pip>	Shifts output to fullscreen for a specific pip window. <pip>: 1-16 = Pip window to enlarge. FULLSCREEN=3
PASSTHRU=<string>	Allows user the capability of sending commands directly to the MX 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=ECHO_ON
PIPTEXT=<value>:<pip>[:<text>:<fgcolor>:<bgcolor>:<pos>]	Set/Display individual pip caption. Data in braces ([]) is only used when <value> is 1. <value>: 0 = Remove pip caption from screen. 1 = Show pip caption. <pip>: 1-16 = Pip window for this operation. <text>: 8 character caption. Alphanumeric characters and +-./_, <fgcolor>: Hex value for foreground color. (See table in BORDERFORMAT command). <bgcolor>: Hex value for background color. (See table in BORDERFORMAT command). <pos>: 0 = Top of window. 1 = Bottom of window. PIPTEXT=0:
TALLY=<value>:<tid>:<pip>	Sets/removes a special border around a specific pip window. <value>: 0 = Turn Tally OFF 1 = Turn Tally ON <tid>: 1 or 2 (two tally windows supported) <pip>: 1-16 = pip window to highlight with tally. TALLY=1:1:4 Note: Turning on a specific tally ID will cause the previous pip window's tally using that same tally ID to OFF.

*BORDERFORMAT=<value>	Sets the BORDER color for all PIP screens. <value>: 1B = Yellow 29 = Cyan 37 = Green 45 = Magenta 54 = Red 62 = Blue 77 = Orange 00-0F = GrayScale F0 = Half-tone video behind labels NOTE: Other hex values may be used but values may be outside of the color range allowed. BORDERFORMAT=62 Sets border colors to Blue
*CAPTIONFORMAT=<text>:<fgcolor>:<bgcolor>:,<xpos>:<ypos>	Set Caption parameters. <text>: 13 character caption. Alphanumeric characters and +-./_, <fgcolor>: Hex value – see Border color values. <bgcolor>: Hex value – see Border color values. <xpos>: 0-999* – Horizontal text position <ypos>: 0-999* – Vertical text position CAPTION=Caption_text *Note: extreme values may cause wraparound.
*CLOCKFORMAT=<style>:<pos>	Sets the format for the clock display. <style>: 0 = time 1 = date 2 = time & date <pos>: 0 = top left 1 = top right 2 = bottom left 3 = bottom right CLOCKFORMAT=2:3

OUTPUTFORMAT=<scan>: <format>:<xpos>:<ypos>	Set up the output video display. <scan>: 0 = Underscan 1 = Overscan <format>: 0 = RGB 1 = YCrCB 2 = Comp/S-Video <xpos>: -9 to +9* <ypos>: -9 to +9 OUTPUTFORMAT=0:2:0:0 *Note: only use even numbers to retain color fidelity.
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 **MX** changes via string events. The following strings are reported as the result of a specific request (commands ending in ?) or as the result of the value being changed.

Other commands will echo back a response when the command is accepted. For example, the successful completion of an OUTPUTFORMAT command will return OUTPUTFORMAT=OK.

String	Description

`BORDER=<value>`	`Reports the border status.`
	
	`<value>:  0 = Border OFF`
	`          1 = Border ON`
	
	`BORDER=1  Turns on borders with the previously defined color.`

`CAPTION=<value>`	`Reports the caption status.`
	
	`<value>:  0 = Caption OFF`
	`          1 = Caption ON`
	
	`CAPTION=1`

`CLOCK=<value>`	`Reports the status of the clock display.`
	
	`<value>:  0 = CLOCK OFF`
	`          1 = CLOCK ON`
	
	`CLOCK=1`

`COLORBAR=<value>`	`Reports the colorbar status.`
	
	`<value>:  0 = COLORBAR OFF`
	`          1 = COLORBAR ON`
	
	`COLORBAR=1`

`DEBUG=<value>`	`Reports debug status.`
	
	`<value>: 0 = OFF`
	`         1 = ON`
	
	`DEBUG=1`

DISPLAY:<value>	Reports the Display pattern. (MX16 Only according to Zandar documentation) ID Layout Channels <value>: 1 4x4 1-16 2 12+1 1-13 3 12+1 2-14 4 12+1 3-15 5 12+1 4-16 6 3x3 1-9 7 3x3 4-12 8 3x3 7-15 9 2x2 1-4 10 2x2 5-8 11 2x2 9-12 12 2x2 13-16 13 PRST1 1-10 14 PRST2 1-8 15 PRST3 1-6 16 PRST4 1-12 For MX-4, ID 1 shows a 2x2 layout DISPLAY:9
FADE=<value>	Reports the FADE value. <value>: 0 = Normal Image 1 = Faded to Black FADE=1
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 MX devices. Some of the MX commands have been omitted and may be executed using the PASSTHRU command.

Programming Notes

The COMM module imposes synchronous operation i.e. the COMM will not send a command to the MX 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 (100 milliseconds).

The Request commands return the state that is currently stored in the COMM module. They do NOT send a new Request to the device. 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.