Ethernet Protocol Description

ScreenMaster III 3216 and Montage SC-3200

Document Revision 1.1

11/22/2020

Contents

Introduction

This document describes the Ethernet interface used to communicate with the hardware buttons, joystick, T-Bar, and rotary controls present on the ScreenMaster III 3216 and the Montage SC-3200 hardware control consoles made by Vista Control Systems, Corp. (later renamed to Vista Systems and later acquired by Christie Digital Systems).

The purpose for creating this document is to help anyone who would like to retrofit one of these consoles with new software, and re-purpose them with some new software. Each of these consoles used an embedded computer with an internal Ethernet cross-over cable internally connected to an application board also inside the console. The interface uses simple UDP commands to and from the hardware application board, and the remainder of this document will describe the message formats.

The latest version of this document, along with a C# based implementation of the interface is available on GitHub at https://github.com/dsmithson/Vista.Controllers.ScreenMaster3.

For the remainder of this document, the term **host** will be used to describe the controlling PC (and/or application) and the term **keyboard** will be used to refer to the physical hardware (button boards / application board / rotary encoders / joystick system).

IP Addressing and port Scheme

As far as I'm aware, the IP address of the keyboard cannot be changed and has a fixed IP address of 192.168.1.3. Similarly, the keyboard is hard-coded to send event messages to a specific IP address, requiring the host to have an IP address of 192.168.1.2 (subnet mask is 255.255.0.0).

All messages to the keyboard are sent to UDP port 9995, and events coming from the keyboard to the host are also sent using UDP port 9995.

Message Structure

All control and update messages to and from the keyboard, respectively, are made using standard messages described within this section. Each message is contained within a single UDP packet.

Each UDP packet will contain a message header and a payload. The message header is a static number of bytes which identify the packet type and optionally provides command-specific arguments. The data payload is a variable number of bytes (depending on message type), and contains data specific to the command.

UDP Message Format:
[Message Header (28 bytes)] [Data Payload (0-N bytes)]

Message Header

Messages sent both from the keyboard to the host and from the host to the keyboard will begin with a 28-byte header. Messages may also not make use of all arguments in the message header, however the full size of the message header will always be present at the beginning of the packet. These bytes are shown in the table below:

Byte Offset	Number of Bytes	Data Type	Description
0	6	String	Static header – always set to: "3216kb"
7	4	Unsigned integer	Command
11	4	Unsigned integer	Argument 1
15	4	Unsigned integer	Argument 2
19	4	Unsigned integer	Argument 3
23	4	Unsigned integer	Argument 4

Depending on the command type supplied in the message, the full UDP message payload may contain additional data after the command header.

Sending Messages

Messages can be sent from the host to the keyboard to illuminate LEDs, set Quick Key background color (red/green), and set Quick Key text.

Command Types and Arguments

Command table below reverse-engineered from old code enumerations, and further details on commands not listed in more detail in subsequent sections are unknown.

Command Type	Command	Argument 1	Argument 2	Argument 3	Argument 4
LoaderMessage	100	START LOAD REBOOT			
SetLamps	101				
SetText1	102	Board	Switch		
SetText8	103	Board			
SetText48	104				
GetStatus	105				
ClearErrors	106				
SetDiagFlags	107	Flags			
TestModules	108				
TestRotaryEncoder	109				
LoaderStart	110		Address	Filesize	File checksum
LoadBlock	111		Block ID	Size	Block checksum
Reboot	112				

Update Lamps and LCD Background Colors

This message payload contains bitwise operators to set the on/off status of each lamp for each physical key on the console in a single message, and additionally sets the LCD background color for each of the Quick Key buttons. As this command is a full refresh of the keyboard state, the host needs to maintain an internal map of all lamp states to make partial updates.

NOTE: The incandescent lamps inside each of the keys consume a reasonable amount of power, and there is a limit to the number of lamps that can be lit at one time. As of this writing, this limitation is known but the specific number of lamps that can be lit at once is not known.

Byte Offset	Number of Bytes	Value	Notes
0	6	3216kb	Header – Static
7	4	101	Header – Command
11	16	Blank	Header – Unused
27	36	Bit-wise lamp on values (1=on/0=off)	36 bytes supports 288 unique buttons

63	48	Byte color value for each LCD button 0 = Off 1 = Green 2 = Red 3 = Alternate 4 = Green/Red 5 = Red/Green	48 bytes corresponds to the 48 quick key buttons

Update LCD Button Text – 1 Button

This command updates the text of a single LCD button. When the controlling application knows it's updating only a single button, this is the most performant update mechanism.

Byte Offset	Number of Bytes	Value	Notes
0	6	3216kb	Header – Static
7	4	102	Header – Command
11	4	Board id	Header – Arg 1
15	4	Switch id	Header – Arg 2
19	8	Blank	Header – Unused
27	18	Text for LCD Button	Up to 18 characters (3 lines * 6 characters per line)

Update LCD Button Text – 8 Buttons (1 row)

This command updates the text of a full row of 8 LCD buttons.

Byte Offset	Number of Bytes	Value	Notes
0	6	3216kb	Header – Static
7	4	103	Header – Command
11	4	Board id	Header – Arg 1
15	12	Blank	Header - Unused
27	144	Text for 8 LCD Buttons	144 Characters – 18 characters per button

Update LCD Button Text – All Buttons

This command updates the text all LCD buttons in a single message.

Byte Offset	Number of Bytes	Value	Notes
0	6	3216kb	Header – Static
7	4	103	Header – Command
11	16	Blank	Header - Unused
27	864	Text for all 48 LCD Buttons	864 Characters – 18 characters per button

Receiving Messages

Hardware events, including press/release, joystick, and T-Bar actions made physically on the keyboard are translated to Ethernet messages sent to the host PC.

Keyboard to host event messages use the same message header format as host to keyboard messages and have designated commands specifically for each event type (listed below). Unlike host to keyboard messages, however, none of the event messages contain data payloads. *Note that this may be inaccurate, as details for system and error message event types are not currently known. As such, event consumers should handle UDP packets larger than the message header size.*

Event Type	Command	Argument 1	Argument 2	Argument 3	Argument 4
Key Press / Release	0	Key Index	0 = Release 1 = Pressed		
Joystick	1	X Position	Y Position	Z Position	
T-Bar	2	Position			
Rotary Encoder	3	Rotary Index	Rotary Offset		
System	4	???	???	???	???
Error	5	???	???	???	???