



ANALOG WAY®
Pioneer in Analog, Leader in Digital

Programmer's Guide

Axion2



1. Introduction

The Axion2 allows communication through an ASCII code protocol. It treats any character that it receives on the LAN (TCP/UDP) as a possible command but only accepts legal commands.

There is no starting/ending code needed in a command string. A command can be a single character typed on a keyboard and does not require any special character before or after it. (It is not necessary to press "ENTER" on the keyboard). A command can be preceded by a value (See chapter A-2). A command is case sensitive.

When the Axion2 receives a valid command, it will be executed. Then it will send back the status of the parameters that have changed due to this command. If the command cannot be executed (value out of range, device not ready, etc), the Axion2 will just send back the current status of the corresponding parameters. If the command is invalid, an error response will be returned. All responses end with a carriage return <CR> and a line feed <LF> signaling the end of the response character string (see chapter A-3).

2. Command Structure

- Write/Set: [[index,] ...][Value][Command]
- Read/Get: [[index,] ...][Command]
- Answer: [Command][[index,] ...][Value]

Some commands are "global" and do not require any index. Other commands might need indexes to specify which screen/element/matrix... need to be affected. Each index should be followed by a comma.



1.Command List

	Command	Answer	Description	Access	Min	MAX	Index 1	Index 2	Index 3	Value
	System	?	DEV	Device Type	Read	96	96			
#		#	Request all current commands values	ReadWrite	0	1				1 = Request >> 0 when completed
rA		rA	Axion2 Ready Status	Read	0	1				1 = Ready
rS		rS	Screen Ready Status	Read	0	1	0 = Screen1 ... 5 = Screen6			1 = Ready
rM		rM	Matrix Ready Status	Read	0	1	0 = MatrixId1 ... 15 = MatrixId16			1 = Ready



	Command	Answer	Description	Access	Min	MAX	Index 1	Index 2	Index 3	Value
Version	vF	vF	Firmware Version	Read	0	65535				
	vC	vC	Command List Version	Read	0	65535				

	Command	Answer	Description	Access	Min	MAX	Index 1	Index 2	Index 3	Value
Preset	PM	PM	Loading Mode	ReadWrite	0	1				0 = Load everything from the Memory 1 = Load a Memory
	PA	PA	Load a Preset Memory with its screen filter and configure the Preview	ReadWrite	0	90				0 = None 1 = Load Memory #1 ... 90 = Load Memory #90
	PL	PL	Load a Preset Memory using current screen filter and configure the Preview	ReadWrite	0	90				0 = None 1 = Load Memory #1 ... 90 = Load Memory #90
	SS	SS	Switch a source per Screen/EltType/EltIndex and configure the Preview	ReadWrite	0	64	0 = Screen1 ... 5 = Screen6	0 = BG Frame 1 = BG Live 2 = PIP 3 = Logo	0 = EIt 1 ... 63 = EIt 64	0 = None 1 = Source #1 ... 64 = Source #64
	SA	SA	Autocentering request of an element's source per Screen / EItType / EItIndex	ReadWrite	0	1	0 = Screen1 ... 5 = Screen6	0 = BG Frame 1 = BG Live 2 = PIP 3 = Logo	0 = EIt 1 ... 63 = EIt 64	1 = Autocentering >> 0 when completed



	Command	Answer	Description	Access	Min	MAX	Index 1	Index 2	Index 3	Value
Preset Control	SF	SF	Screen Filter (Disable / Enable Preset Load and Take per Screen)	ReadWrite	0	1	0 = Screen1 ... 5 = Screen6			1 = Enabled
	TK	TK	Take as soon as devices are ready	ReadWrite	0	1				1 = Take >> 0 when completed



2.Examples

Write command		Answer	Description
Value	Characters		
10	PL	PL10<CR><LF>	Load the Preset Memory #10
2,2,4,6	SS	SS2,2,4,6<CR><LF>	Put the source #6 into the 5th PIP of screen #3
Read command		Answer	Description
Index	Characters		
	?	DEV96<CR><LF>	Get the Device Type (96=ARC200)
0,	rS	rS0,1<CR><LF>	Get the status of screen#1 (1=Ready)

