# Automation Protocols

## For Ensemble Designs Products

AVENUE
Router Control
Source: Video Server 1
Destination: Satellite Link
ENSEMBLE DESIGNS
AVENUE™ signal integration system

Revision 3.2

ENSEMBLE
DESIGNS

*Purveyors of Fine Video Gear—Loved by Engineers Worldwide*

*Clearly, Ensemble wants to be in the broadcast equipment business. It's so rare anymore to find a company of this caliber that has not been gobbled up by a large corporation. They are privately held so they don't have to please the money people. They really put their efforts into building products and working with customers.*

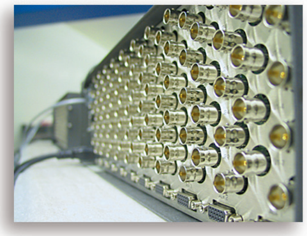
*I'm really happy with the Avenue products and Ensemble's service, and even more important my engineers are happy. We've continued to upgrade the product and add more cards. We will be rebuilding our production control room and we will use Avenue again.*

~ *Don McKay, Vice President Engineering, Oregon Public Broadcasting*

## Who is Ensemble Designs?

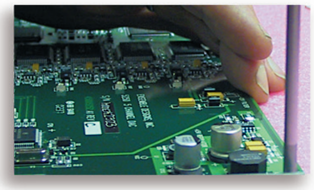
### By Engineers, For Engineers

In 1989, a former television station engineer who loved designing and building video equipment, decided to start a new company. He relished the idea of taking an existing group of equipment and adding a few special pieces in order to create an even more elegant ensemble. So, he designed and built his first product and the company was born.


Avenue frames handle 270 Mb/s, 1.5 Gb/s and 3 Gb/s signals, audio and MPEG signals. Used worldwide in broadcast, mobile, production, and post.

### Focused On What You Need

As the company has grown, more former TV station engineers have joined Ensemble Designs and this wealth of practical experience fuels the company's innovation. Everyone at the company is focused on providing the very equipment you need to complete your ensemble of video and audio gear. We offer those special pieces that tie everything together so that when combined, the whole ensemble is exactly what you need.


We're focused on processing gear– 3G/HD/SD/ASI video, audio and optical modules.

### Notably Great Service for You

We listen to you – just tell us what you need and we'll do our best to build it. We are completely focused on you and the equipment you need. Being privately held means we don't have to worry about a big board of directors or anything else that might take attention away from real business. And, you can be sure that when you call a real person will answer the phone. We love this business and we're here to stay.

Come on by and visit us. Drop in for lunch and a tour!

### Bricks and Mortar of Your Facility

The bricks and mortar of a facility include pieces like up/downconverters, audio embedders, video converters, routers, protection switches and SPGs for SD, HD and 3Gb/s. That's what we're focused on, that's all we do – we make proven and reliable signal processing and infrastructure gear for broadcasters worldwide, for you.

Shipped with care to television broadcasters and video facilities all over the world.

# Contents

# Automation Protocols

Revision 3.0

# 1 Introduction

This document describes the automation protocols supported by Ensemble Designs' router or router-related products.

## 1.1 Intended Audience

The intended audience is the developer tasked with connecting an external control device to an Ensemble Designs product. It is recommended that the reader be familiar with general configuration procedures for the applicable product by referring to its user manual.

## 1.2 Supported Protocols

Ensemble Designs routers support the following automation protocols:

- GV TEN-XL ASCII Protocol
- GV Performer ASCII Protocol
- Generic ASCII Protocol

For installations that already support one of these protocols, choosing the same protocol can minimize programming changes to the automation system when installing the Ensemble Designs router.

# 2 Conventions

This section describes the conventions used in this document.

- A number followed by the letter "H" is a hexadecimal number. For example, 12H is the hexadecimal equivalent of decimal 18.
- Command syntax in this document is described using a notation similar to BNF (Backus-Naur Form).

| | |
|---|---|
| <symbol> | A symbol is represented by the symbol name inside of arrow brackets '<' and '>'. |
| [optional] | Optional parts are shown inside of square brackets. |
| … | Items that can be repeated are indicated by an ellipse. |
| literals | Characters, numbers, and strings of text that are not part of a symbol are to be interpreted literally. They would be typed exactly as they are shown. |
| '>' | Single quotes may be used in certain cases to avoid ambiguity when a character needs to be typed literally. For example, since the arrow bracket is normally used to indicate a symbol, enclosing it in single quotes means it should be interpreted literally as the arrow bracket character. In this case it should be typed as a single character (without the quotes). |

Syntax Example:

@<Space>X:<Lvls>/<Dest>,<Src>[/<Dest>,<Src>…]<CR>

The symbols in the above example are <Space>, <Lvls>, <Dest>, <Src>, and <CR>. The meaning of symbols like these will be defined as they occur throughout this document.

The optional portion in the example (enclosed in square brackets), shows that an additional "/<Dest,<Src>" element may be present. It also indicates (with the ellipse character) that this optional element may occur repeatedly.

The following characters in the example are literals and should be typed as they appear (no quotes):

'@', 'X', ':', '/', and ','

## 3 Configuring Control Points

The presentation of router inputs and outputs to a control point, such as a control panel, web browser, automation system, etc., is configurable in Ensemble Designs products. This presentation is managed through a "Profile"—a map of the sources (inputs) and destinations (outputs) that are available to a control point.

This map, or profile, also determines the order in which the sources and destinations appear to the control point. These Profiles thereby provide the flexibility for every control point to have its own view (or a shared view) of the organization of inputs and outputs. As such, Profiles are of particular interest for installations using an automation system.

## 3.1 Understanding Profiles

Router products from Ensemble Designs use Profiles to determine how sources and destinations are presented.

A Profile provides a configured view of the sources and destinations in a router. It determines which sources and destinations are accessible and the order of these sources and destinations as seen by a controlling device. This ordering feature is of particular interest for automation control because it provides a simple means to map sources and destinations to suit the automation system's requirements.

An example will help illustrate how a Profile is used to connect an automation system to the router.

**Note:** It is assumed the reader is familiar with the methods for configuring input and output ports in the router. For help with these topics, please refer to the appropriate Ensemble Designs router manual.

Assume that we have cabled up the router so that it has 4 inputs and 2 outputs. Table 1 lists the inputs and outputs connected to the router in this example.

*Table 1*

| **Router port number** | **Src/Dest Name** |
|---|---|
| Input Port 1 | Network Feed |
| Input Port 2 | Studio 3 |
| Input Port 3 | Server 6 |
| Input Port 4 | Server 4 |
| Output Port 1 | Microwave |
| Output Port 2 | Transmitter |

Suppose we want to access these inputs and outputs from an automation system and that we want to renumber the sources and destinations to fit the automation system's numbering scheme. In Table 2 we have added a column to list the desired automation number associated with each input/output.

*Table 2*

| **Router port number** | **Src/Dest Name** | **Desired automation number** |
|---|---|---|
| Input Port 1 | Network Feed | source #7 |
| Input Port 2 | Studio 3 | source #5 |
| Input Port 3 | Server 6 | source #2 |
| Input Port 4 | Server 4 | source #3 |
| Output Port 1 | Microwave | destination #8 |
| Output Port 2 | Transmitter | destination #3 |

By rearranging the information in Table 2, we have produced a sorted list of sources and a sorted list of destinations for the automation system. The results are in Table 3 and Table 4.

*Table 3*

| Desired automation Src number | Src Name | Router port number |
|---|---|---|
| 1 | - | - |
| 2 | Server 6 | Input Port 3 |
| 3 | Server 4 | Input Port 4 |
| 4 | - | - |
| 5 | Studio 3 | Input Port 2 |
| 6 | - | - |
| 7 | Network Feed | Input Port 1 |

*Table 4*

| Desired automation Dest number | Dest name | Router port number |
|---|---|---|
| 1 | - | - |
| 2 | - | - |
| 3 | Transmitter | Output Port 2 |
| 4 | - | - |
| 5 | - | - |
| 6 | - | - |
| 7 | - | - |
| 8 | Microwave | Output Port 1 |

**Note:** In this example, we are assuming a protocol that numbers sources and destinations beginning with the number 1. Some protocols begin numbering at zero, such as the Grass Valley TEN-XL ASCII Protocol and the Generic ASCII protocol.

We can now use these lists to create a Profile in the router that numbers the sources and destinations to match the desired automation numbering scheme.

By inserting spaces into the Profile's source or destination lists, we can obtain the desired automation numbering. Then we assign this Profile to the automation device that will be controlling the router. The Profile to accomplish the numbering scheme for this example is illustrated in Figure 1.

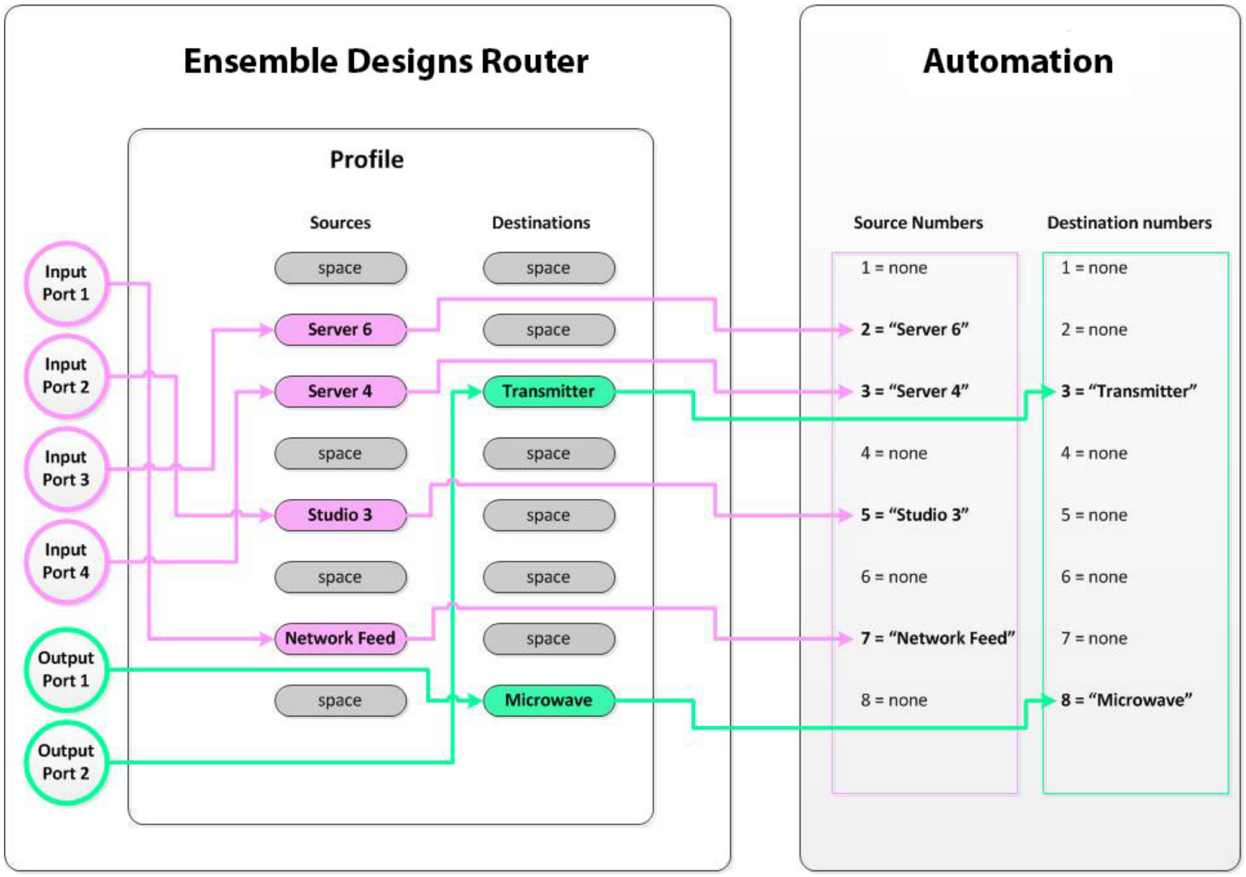


*Figure 1*

To change the number for a particular source or destination, simply edit the Profile and move the source or destination to its new position until the order produces the numbering scheme required by the automation system. For example, if Transmitter needs to be changed to destination 1, simply place it at the top of the destination list in the Profile.

**Note:** For detailed instructions on how to create and edit Profiles, refer to the appropriate Ensemble Designs router manual.

# 4 GV TEN-XL ASCII Protocol

The Grass Valley TEN-XL ASCII Protocol is a serial protocol used by TEN-XL Routing Switchers. For installations that already support this protocol, this is a good choice to minimize programming changes to the automation system.

The implementation of this protocol has been adapted as appropriate to reflect the features in Ensemble Designs routers.

## 4.1 Protocol Requirements

- Sources and destinations are numbered starting at 0.
- All commands begin with the ASCII STX character (02H).
- Receipt of STX in the middle of a command discards all collected bytes and begins a new command.
- Errors of any kind, such as missing parameters, unexpected characters, out of range values, etc., result in discarding the current command.
- Parsing will not tolerate any additional characters within a command. For example, spaces embedded within a command will cause the command to be discarded.
- Audio crosspoints are ignored on switching requests. In the Command Response message, the audio source is always reported as '0' (30H).
- The Power Supply status returned in the Command Response message is always reported as '3' (33H), which means okay.
- The protocol requires 2 digits to specify a destination. This 2-digit number represents a decimal number. For example, a destination that is specified with a high digit of '1' and a low digit of '2' represents the decimal number 12.

## 4.2 Commands

This section lists the commands supported in the TEN-XL ASCII protocol.

### 4.2.1 Status Request Command

Description: This command requests the current source selected on the specified destination.

Syntax: <STX><Dest High><Dest Low><ENQ>

| | |
|---|---|
| <STX> | Start character (02H) |
| <Dest High> | High byte of destination number.<br>Valid range is '0' to '9' (30H to 39H) |
| <Dest Low> | Low byte of destination number.<br>Valid range is '0' to '9' (30H to 39H) |
| <ENQ> | Enquiry character (05H) |

Example: Request the status for destination number 13. Assume that the video source assigned to this destination is 7.

| | | |
|---|---|---|
| Request: | In Hex | 02 31 33 05 |
| | In ASCII | <STX>13<ENQ> |
| Response: | In Hex | 37 30 33 |
| | In ASCII | 703 |

### 4.2.2 Crosspoint Select Command

| | |
|---|---|
| Description: | This command requests switching the specified video source to the specified destination. Note that the audio crosspoint specified will have no effect; however, it must be a legal ASCII decimal digit from 0 to 9. |
| Syntax: | <STX><Dest High><Dest Low><Video Src><Audio Src> |

| | |
|---|---|
| <STX> | Start character (02H) |
| <Dest High> | High byte of destination number. Valid range is '0' to '9' (30H to 39H) |
| <Dest Low> | Low byte of destination number. Valid range is '0' to '9' (30H to 39H) |
| <Video Src> | Video source number. Valid range is '0' to '9' (30H to 39H) |
| <Audio Src> | Audio source number. Valid range is '0' to '9' (30H to 39H) |

| | | |
|---|---|---|
| Example: | Request switching video source 4 to destination 0. In this example, we specify an arbitrary audio source number of 9 which will be ignored. | |
| Request: | In Hex | 02 30 30 34 39 |
| | In ASCII | <STX>0049 |
| Response: | In Hex | 34 30 33 |
| | In ASCII | 403 |

Notes: If the video source requested is not valid for the specified destination, the response will return the current video source that is assigned to that destination.

If the destination number specified is invalid, the command is invalid and there is no response.

### 4.2.3 Command Response Message

Description: This is the response to all valid GV TEN-XL commands sent to the router.

Syntax: <Video Src><Audio Src><PS OK>

| | |
|---|---|
| <Video Src> | Video source number assigned to requested destination. Valid range is '0' to '9' (30H to 39H). |
| <Audio Src> | Audio source number. Not used. Always reported as '0' (30H). |
| <PS OK> | Power supply okay. Not used. Always reported as okay: '3' (33H). |

Example: Router receives a Status Request command for destination number 1. Assume that video source 6 is assigned to this destination.

| | | |
|---|---|---|
| Request: | In Hex | 02 30 31 05 |
| | In ASCII | <STX>01<ENQ> |
| Response: | In Hex | 36 30 33 |
| | In ASCII | 603 |

Notes: The Audio Source number and the Power Supply portions of the response are required by the protocol, but the values returned are to be ignored.

# 5 GV Performer ASCII Protocol

The Grass Valley Performer ASCII protocol is a serial protocol used by the Performer™ 10 x 1 Routing Switcher. For installations that already support this protocol, this is a good choice to minimize programming changes to the automation system.

The implementation of this protocol has been adapted as appropriate to reflect the features in Ensemble Designs routers.

## 5.1 Protocol Requirements

- Sources and destinations are numbered starting at 1.
- The Ensemble Designs router may support source and destination numbers beyond the limit specified in the original protocol. Any valid source or destination in the Ensemble Designs router can be specified in this implementation of the protocol.
- The maximum byte count for Cmd Data is 251 bytes (FBH).
- Multiple commands may be sent within a single message.
- All digits in messages are ASCII hex digits. For example, a message length specified with a high digit of '1' and a low digit of '2' represents the hexadecimal number 12H, or decimal 18.
- ASCII hex digits are not case specific. The Ensemble Designs router will accept upper or lower case ASCII hex digits.
- All messages begin with a carriage return character (0DH).
- All line feeds (0AH), spaces (20H), and nulls (00H) are ignored within messages, and are not included in the message length calculation.
- The Performer Switch 1 Address bytes in messages are ignored; however, they must be legal ASCII hex digits from '0' to 'F'.
- Errors of any kind, such as missing parameters, unexpected characters, out of range values, etc., result in discarding the remaining portion of the message at the point of the error.
- The P, C, R, and Q commands are not supported and will be ignored.
- No response is sent from the Ensemble Designs router because none of the supported commands require a response.

## 5.2 Message Structure

This section describes the message structure supported in the GV Performer ASCII Protocol.

A message consists of a Message Header followed by Command Data (Cmd Data). The Message Header consists of 5 bytes of information (described below). The Command Data portion can vary in size depending on which commands and how many commands are included within it.

The general message structure is as follows:

Syntax: <CR><Adr High><Adr Low><Len High><Len Low><Cmd Data>

| | |
|---|---|
| <CR> | Carriage return (0DH).<br>Part of the Message Header. |
| <Adr High> | High byte of Performer Switch 1 Address.<br>Part of the Message Header.<br>This is ignored, but must be valid ASCII hex digit.<br>(E.g. ‘0’ to ‘9’ or ‘A’ to ‘F’). |
| <Adr Low> | Low byte of Performer Switch 1 Address.<br>Part of the Message Header.<br>This is ignored, but must be valid ASCII hex digit.<br>(E.g. ‘0’ to ‘9’ or ‘A’ to ‘F’). |
| <Len High> | High byte of byte count for Cmd Data.<br>Part of the Message Header.<br>Must be valid ASCII hex digit.<br>(E.g. ‘0’ to ‘9’ or ‘A’ to ‘F’). |
| <Len Low> | Low byte of byte count for Cmd Data.<br>Part of the Message Header.<br>Must be valid ASCII hex digit.<br>(E.g. ‘0’ to ‘9’ or ‘A’ to ‘F’). |
| <Cmd Data> | Zero or more commands consisting of valid ASCII characters. |

## 5.3 Commands

This section describes the commands in the GV Performer ASCII Protocol that are supported by Ensemble Designs routers. Commands in the original protocol that are not supported by Ensemble Designs routers will simply be ignored.

The commands described here fit into the <Cmd Data> field of the message described in Section 7.2, "Message Structure," in this document.

Note that multiple commands may be sent in a single message up to a maximum of 251 bytes.

### 5.3.1 Data Preset Command

Description: This command presets a crosspoint for later execution. The source is not actually switched to the destination until a subsequent **Take** command or **All Take** command is received by the router.

Syntax:

Message Header:
<CR><Adr High><Adr Low><Len High><Len Low>

Cmd Data:
D<Lvl High><Lvl Low><Src High><Src Low><Dest High><Dest Low>

| | |
|---|---|
| D | ASCII character 'D' identifies this as Data Preset command. |
| <Lvl High> | High byte of Level. See note below. |
| <Lvl Low> | Low byte of Level. See note below. |
| <Src High> | High byte of Source number.<br>Any valid ASCII hex digit. (E.g. '0' to '9' or 'A' to 'F'). |
| <Src Low> | Low byte of Source number.<br>Any valid ASCII hex digit. (E.g. '0' to '9' or 'A' to 'F'). |
| <Dest High> | High byte of Destination number.<br>Any valid ASCII hex digit. (E.g. '0' to '9' or 'A' to 'F'). |
| <Dest Low> | Low byte of Destination number.<br>Any valid ASCII hex digit. (E.g. '0' to '9' or 'A' to 'F'). |

Note: The only Levels supported by Ensemble Designs routers are "All Levels" ("00" or 3030H) and "Video Level" ("01" or 3031H). These 2 Level values are treated by the router as equivalent to one another. Any other Level value will cause the command to be ignored.

Example: Request a preset of Source 14 (0EH) on Destination 2.
The Address bytes are ignored, so the example specifies an arbitrary value of "00" (3030H). For the Level bytes, the example specifies "01" (3031H). There are 7 bytes of Cmd Data in this example (specified as "07" or 3037H).

| | | |
|---|---|---|
| Request: | In Hex | 0D 30 30 30 37 44 30 31 30 45 30 32 |
| | In ASCII | <CR> 00 07 D 01 0E 02 (spaces for readability only) |

Response: No serial response. The router remembers the preset source for the specified destination.

### 5.3.2 Take Command

Description: This command requests a Take of a previously set preset source to the specified destination.

Syntax:

Message Header:
<CR><Adr High><Adr Low><Len High><Len Low>

Cmd Data:
T<Dest High><Dest Low>

| | |
|---|---|
| T | ASCII character 'T' identifies this as Take cmd. |
| <Dest High> | High byte of Destination number.<br>Any valid ASCII hex digit. (E.g. '0' to '9' or 'A' to 'F'). |
| <Dest Low> | Low byte of Destination number.<br>Any valid ASCII hex digit. (E.g. '0' to '9' or 'A' to 'F'). |

Example: Request a Take of the previous set preset source to Destination 4.

Request:

| | | |
|---|---|---|
| In Hex | 0D 30 30 30 33 54 30 34 | |
| In ASCII | <CR> 00 03 T 04 | (spaces for readability only) |

Response: No serial response. If a preset source had previously been set for Destination 4, the router now switches that source to Destination 4.

### 5.3.3 All Take Command

Description: This command requests a Take of all previously set preset sources to their respective destinations.

Syntax:

Message Header:
<CR><Adr High><Adr Low><Len High><Len Low>

Cmd Data:
A

| | |
|---|---|
| A | ASCII character 'A' identifies this as All Take command. |

Example: Request a Take of all the previously set preset sources to their respective destinations.

| | | |
|---|---|---|
| Request: | In Hex | 0D 30 30 30 31 41 |
| | In ASCII | <CR> 00 01 A (spaces for readability only) |

Response: No serial response. All preset sources are switched to their respective destinations.

# 6 Generic ASCII Protocol

The Generic ASCII protocol is based on the Leitch Pass-Through Protocol described in Chapter 4 of the Leitch serial protocol reference manual "SPR-MAN", edition 4.

The Generic ASCII protocol supports a subset of the commands described in the Leitch protocol. Commands described in the Leitch protocol that are unsupported by Ensemble Designs routers are simply ignored.

The implementation of this protocol has been adapted as appropriate to reflect the features in Ensemble Designs routers.

## 6.1 Protocol Requirements

- Sources and destinations are numbered starting at 0.
- Commands must be sent one at a time. Before sending the next command, wait for the router to send the prompt ('>').
- Errors of any kind (e.g., missing parameter, unexpected characters, out of range values, etc.) result in discarding the current command. There is no error reporting.
- Parsing will not tolerate any additional characters within a command. For example, spaces embedded within a command will cause the command to be discarded.
- All digits in messages are ASCII hex digits. For example, a source specified with a high digit of '1' and a low digit of '2' represents the hexadecimal number 12H, or decimal 18.
- ASCII hex digits are not case specific. Ensemble Designs routers will accept upper or lower case ASCII hex digits. Responses from Ensemble Designs routers will always use lower case for ASCII hex digits.
- Sources and destinations can be specified using up to 4 ASCII digit characters. E.g., Source 2 can be specified as "2", "02", "002", or "0002".
- Ensemble Designs routers do not support multiple levels as described in the Pass-Through Protocol. In commands that specify a level number, Ensemble Designs routers will only respond to level 0 (unless otherwise noted). All other level designations are ignored. Levels are specified in a single hex digit from '0' to 'F'.

## 6.2 Router Responses

All commands sent to Ensemble Designs routers must be terminated with a carriage return character (0DH). Upon receiving a carriage return, Ensemble Designs routers will send a response to the automation system.

The possible responses are a simple Prompt Response or a Status Response followed by a Prompt Response. These are discussed below.

### 6.2.1 Prompt Response

This response consists of the single ASCII character '>' (3EH).

An Ensemble Designs router will send this response anytime it receives a carriage return. This is an indication to the automation system that the router is ready for the next command.

If a command is sent to an Ensemble Designs router that requires a more extensive response, the Prompt Response will be sent immediately following the more extensive response.

### 6.2.2 Status Response

Certain commands sent to an Ensemble Designs router require that a Status Response be returned by the router. This response identifies the source that is currently switched to the destination specified. The response takes this form:

Syntax: S:<Lvl><Dest>,<Src><CR><LF>

| | |
|---|---|
| S: | ASCII character 'S' (53H) followed by ASCII character ':' (3AH).<br>Identifies this as a Status Response. |
| <Lvl> | This is the level.<br>Ensemble Designs routers always report this to 0. |
| <Dest> | Destination number.<br>No leading zeros will be used.<br>Any valid ASCII hex digits. (E.g. '0' to '9' or 'A' to 'F'). |
| , | Comma character (2CH).<br>Separates destination numbers from source numbers. |
| <Src> | Source number.<br>No leading zeros will be used.<br>Any valid ASCII hex digits. (E.g. '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |
| <LF> | ASCII line feed character (0AH). |

Example: In response to a **Direct Crosspoint Take** command, suppose that the router reports that Source 11 (0BH) is selected on Destination 4. The response would look as follows. (Note that the router will always report the Level as 0):

Response: In Hex 53 3A 30 34 2C 62 0D 0A
In ASCII S:04,b<CR><LF>

Note: After the **Status Response** is sent, a **Prompt Response** is sent.

## 6.3 Commands

This section lists Generic ASCII Protocol commands that originated in the Leitch Pass-Through Protocol and are supported by Ensemble Designs products.

Commands from the original Leitch Pass-Through Protocol that are not listed in this section are not supported by Ensemble Designs routers and will be ignored.

### 6.3.1 Direct Crosspoint Take Command

Description: This command requests a Take operation on the specified destination-source pair(s).

Syntax: @<Space>X:<Lvls>/<Dest>,<Src>[/<Dest>,<Src>…]<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| X: | ASCII 'X' followed by ASCII ':'.<br>Identifies this as a Direct Crosspoint Take command. |
| <Lvls> | The level(s) to be affected. Multiple levels may be specified by listing several digits in a row.<br>Ensemble Designs routers only support level 0 and ignores other levels.<br>A level is specified by a single valid ASCII hex digit.<br>(E.g. '0' to '9' or 'A' to 'F'). |
| / | ASCII forward slash character '/' (2FH).<br>Separates Levels from destination-source pairs.<br>Forward slash is also used to separate multiple destination-source pairs when more than one pair is specified. |
| <Dest> | Destination number.<br>Up to 4 characters (including leading zeros) may be used to specify the destination.<br>Any valid ASCII hex digits. (E.g. '0' to '9' or 'A' to 'F'). |
| , | Comma character (2CH).<br>Separates destination numbers from source numbers in each destination-source pair. |
| <Src> | Source number.<br>Up to 4 characters (including leading zeros) may be used to specify the source.<br>Any valid ASCII hex digits. (E.g. '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example 1: Perform a Take of Source 10 (0AH) on Destination 4. Level 0 must be specified.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 58 3A 30 2F 34 2C 41 0D |
| | In ASCII | @ X:0/4,A<CR> |
| Response: | In Hex | 53 3A 30 34 2C 61 0D 0A |
| | In ASCII | S:04,a<CR><LF> |

Example 2: Perform a Take of sources on multiple destinations: Source 9 on Destination 3, Source 7 on Destination 1, and Source 5 on Destination 2.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 58 3A 30 2F 33 2C 39 2F 31 2C 37 2F 32 2C 35 0D |
| | In ASCII | @ X:0/3,9/1,7/2,5<CR> |

| | | |
|---|---|---|
| Response: | In Hex | 53 3A 30 33 2C 39 0D 0A<br>53 3A 30 31 2C 37 0D 0A<br>53 3A 30 32 2C 35 0D 0A |
| | In ASCII | S:03,9<CR><LF><br>S:01,7<CR><LF><br>S:02,5<CR><LF> |

### 6.3.2 Crosspoint Status Command

Description: This command requests the crosspoint status of the specified destination.

Syntax: @<Space>X?<Lvl><Dest><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| X? | ASCII 'X' followed by ASCII '?'.<br>Identifies this as a Crosspoint Status command. |
| <Lvl> | The 9430 Router only supports level 0.<br>Must be ASCII digit '0'. |
| <Dest> | Destination number.<br>Up to 4 characters (including leading zeros) may be used to specify the destination.<br>Any valid ASCII hex digits (for example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Get the status of Destination 7. Level 0 must be specified. Assume that the router reports that Source 6 is selected on Destination 7.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 58 3F 30 37 0D |
| | In ASCII | @ X?07<CR> |
| Response: | In Hex | 53 3A 30 37 2C 36 0D 0A |
| | In ASCII | S:07,6<CR><LF> |

### 6.3.3 Level Status Command

Description: This command requests the crosspoint status of all destinations within the router. Ensemble Designs routers do not have levels, so the level designation is ignored, but it must be a valid ASCII hex digit. The router returns the status of all destinations in the router.

Syntax: @<Space>S?<Lvl><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| S? | ASCII 'S' followed by ASCII '?'.<br>Identifies this as a Level Status command. |
| <Lvl> | The router will ignore this, except it must be a valid ASCII hex digit (for example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Get the status of all destinations in the router. In this example we specify level 0, but any hex digit will suffice. Assume that the router has four destinations with the following crosspoint selections:

Destination 0 has source 1.
Destination 1 has source 9.
Destination 2 has source 6.
Destination 3 has source 11 (decimal).

The router will send as many responses as there are destinations to report. (Each response uses the same format as responses to the Crosspoint Status Command.)

Request: In Hex 40 20 53 3F 30 0D
In ASCII @ S?0<CR>

Response: In Hex

```
53 3A 30 30 2C 31 0D 0A
53 3A 30 31 2C 39 0D 0A
53 3A 30 32 2C 36 0D 0A
53 3A 30 33 2C 42 0D 0A
```

In ASCII

```
S:00,1<CR><LF>
S:01,9<CR><LF>
S:02,6<CR><LF>
S:03,B<CR><LF>
```

### 6.3.4 Preset Crosspoint Command

Description: This command stores crosspoints in the preset buffer for later execution. The preset destination-source pairs are not actually switched until a subsequent Execute Preset Buffer command is issued.

Syntax: @<Space>P:<Lvls>/<Dest>,<Src>[/<Dest>,<Src>…]<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| P: | ASCII 'P' followed by ASCII ':' (3AH).<br>Identifies this as a Preset Crosspoint command. |
| <Lvls> | The level(s) to be affected. Multiple levels may be specified by listing several digits in a row.<br>Ensemble Designs routers only support level 0 and ignore other levels.<br>A level is specified by a single valid ASCII hex digit.<br>(For example, '0' to '9' or 'A' to 'F'). |
| / | ASCII forward slash character '/' (2FH).<br>Separates Levels from destination-source pairs.<br>Forward slash is also used to separate multiple destination-source pairs when more than one pair is specified. |
| <Dest> | Destination number.<br>Up to 4 characters (including leading zeros) may be used to specify the destination.<br>Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| , | Comma character (2CH).<br>Separates destination numbers from source numbers in each destination-source pair. |
| <Src> | Source number.<br>Up to 4 characters (including leading zeros) may be used to specify the source.<br>Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example 1: Perform a preset of Source 10 (0AH) on Destination 4. Level 0 must be specified.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 50 3A 30 2F 34 2C 41 0D |
| | In ASCII | @ P:0/4,A<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

Example 2: Perform presets of the following source-destination pairs: Source 9 on Destination 3, Source 7 on Destination 1, and Source 5 on Destination 2.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 50 3A 30 2F 33 2C 39 2F 31 2C 37 2F 32 2C 35 0D |
| | In ASCII | @ P:0/3,9/1,7/2,5<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.3.5 Execute Preset Buffer Command

Description: This command executes a Take operation on all crosspoints previously stored in the preset buffer.

Syntax: @<Space>B:E<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| B: | ASCII 'B' followed by ASCII ':' (3AH).<br>Identifies this as one of the "buffer" commands. |
| E | ASCII 'E'.<br>Identifies this as the Execute Preset Buffer command. |
| <CR> | ASCII carriage return character (0DH). |

Example: Execute the crosspoints previously stored in the preset buffer. Assume the presets are as follows: Source 9 on Destination 3, Source 7 on Destination 1, and Source 5 on Destination 2.

```
Request:    In Hex      40 20 42 3A 45 0D
            In ASCII    @ B:E<CR>

Response:   In Hex      53 3A 30 33 2C 39 0D 0A
                        53 3A 30 31 2C 37 0D 0A
                        53 3A 30 32 2C 35 0D 0A

            In ASCII    S:03,9<CR><LF>
                        S:01,7<CR><LF>
                        S:02,5<CR><LF>
```

### 6.3.6 Clear Preset Buffer Command

Description: This command clears all crosspoints in the preset buffer.

Syntax: @<Space>B:C<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| B: | ASCII 'B' followed by ASCII ':' (3AH).<br>Identifies this as one of the "buffer" commands. |
| C | ASCII 'C'<br>Identifies this as the Clear Preset Buffer command. |
| <CR> | ASCII carriage return character (0DH). |

Example: Clear the crosspoints previously stored in the preset buffer.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 42 3A 43 0D |
| | In ASCII | @ B:C<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

## 6.4 Ensemble Designs "ED" Commands

This section lists supplemental commands in the Generic ASCII Protocol that were not part of the original Leitch Pass-Through Protocol. These commands were developed to support additional features in Ensemble Designs products.

### 6.4.1 Layering Engine: Request Background Program Crosspoint

Description: This command requests the crosspoint selected on the Layering Engine's background program bus.

Syntax: @ <Space>ED?L<LE#>:B<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H). Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII '?' (3FH). Identifies this as an Ensemble Designs status command. |
| L | ASCII 'L'. Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering engine number followed by ASCII ':' (3AH). Layering engines are numbered starting at 0. A single character is used to specify the layering engine number. Valid ASCII hex digits are '0' or '1'. |
| T: | ASCII 'B'. Identifies this as a Background program crosspoint status request. |
| <CR> | ASCII carriage return character (0DH). |

Example: Request the background program crosspoint in Layering Engine #1:

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3F 4C 31 3A 42 0D |
| | In ASCII | @ ED?L1:B<CR> |

Response Syntax: EDLB:<LE#><Src><CR>

| | |
|---|---|
| ED | ASCII "ED" indicates this is a response to one of the "ED" status request commands. |
| L | ASCII 'L'. Identifies this as a response to one of the "Layering Engine" status request commands. |
| B: | ASCII 'B' followed by ASCII ':' (3AH). Identifies this as a Background program crosspoint status response. |
| <LE#> | Layering engine number. |
| <Src> | Source number currently selected on the background program bus of the Layering Engine. No leading zeros will be used. |
| <CR> | ASCII carriage return character (0DH). |

Example: Assuming that source number 3 is selected on the background program bus in Layering Engine #1, the response would look as follows:

| | | |
|---|---|---|
| Request: | In Hex | 45 44 4C 42 3A 31 33 0D |
| | In ASCII | EDLB:13 |

### 6.4.2 Layering Engine: Request Background Preset Crosspoint

Description: This command requests the crosspoint selected on the Layering Engine's background Preset bus.

Syntax: @<Space>ED?L<LE#>:P<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED? | ASCII "ED" followed by ASCII '?' (3FH).<br>Identifies this as an Ensemble Designs status command. |
| L | ASCII 'L'.<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering engine number followed by ASCII ':' (3AH). Layering engines are numbered starting at 0. A single character is used to specify the layering engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| P | ASCII 'P'.<br>Identifies this as a background Preset crosspoint status request. |
| <CR> | ASCII carriage return character (0DH). |

Example: Request the background preset crosspoint in Layering Engine #0:

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3F 4C 30 3A 50 0D |
| | In ASCII | @ ED?L0:P<CR> |

Response Syntax: EDLP:<LE#><Src><CR>

| | |
|---|---|
| ED | ASCII "ED" indicates this is a response to one of the "ED" status request commands. |
| L | ASCII 'L'.<br>Identifies this as a response to one of the "Layering Engine" status request commands. |
| P: | ASCII 'P' followed by ASCII ':' (3AH).<br>Identifies this as a background Preset crosspoint status response. |
| <LE#> | Layering engine number. |
| <Src> | Source number currently selected on the background preset bus of the Layering Engine. No leading zeros will be used. |
| <CR> | ASCII carriage return character (0DH). |

Example: Assuming that source number 10 (0AH) is selected on the background preset bus in Layering Engine #0, the response would look as follows:

| | | |
|---|---|---|
| Request: | In Hex | 45 44 4C 50 3A 30 41 0D |
| | In ASCII | EDLP:0A |

### 6.4.3 Layering Engine: Request Keyer Status

Description: This command requests the on/off status of a keyer in the Layering Engine.

Syntax: @<Space>ED?L<LE#>:K<Key#><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED? | ASCII "ED" followed by ASCII '?' (3FH).<br>Identifies this as an Ensemble Designs status command. |
| L | ASCII 'L'.<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering engine number followed by ASCII ':' (3AH). Layering engines are numbered starting at 0. A single character is used to specify the layering engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| K | ASCII 'K'.<br>Identifies this as a Keyer status request. |
| <Key#> | Keyer number. Keyers are numbered starting at 0. A single ASCII hex digit is used to specify the keyer. |
| <CR> | ASCII carriage return character (0DH). |

Example: Request the status of Keyer 1 in Layering Engine #0:

Request: In Hex 40 20 45 44 3F 4C 30 3A 4B 31 0D
In ASCII @ ED?L0:K1<CR>

Response Syntax: EDLK:<LE#><Key#><Stat><CR>

| | |
|---|---|
| ED | ASCII "ED" indicates this is a response to one of the "ED" status request commands. |
| L | ASCII 'L'.<br>Identifies this as a response to one of the "Layering Engine" status request commands. |
| K: | ASCII 'K' followed by ASCII ':' (3AH).<br>Identifies this as a Keyer status response. |
| <LE#> | Layering engine number. |
| <Key#> | Keyer number. |
| <Stat> | Keyer status: 'N' indicates keyer is on air. 'F' indicates keyer is off air. |
| <CR> | ASCII carriage return character (0DH). |

Example: Assuming that keyer 1 is on air in Layering Engine #0, the response would look as follows:

Request: In Hex 45 44 4C 4B 3A 30 31 4E 0D
In ASCII EDLK:01N

### 6.4.4 Layering Engine: Set Transition Duration Command

Description: This command sets the transition duration for the Layering Engine.

Syntax: @<Space>ED:L<LE#>:T:<Dur><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L'<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| T: | ASCII 'T' followed by ASCII ':' (3AH).<br>Identifies this as a Set Transition Duration command. |
| <Dur> | Transition duration in frames. Duration value must be between 5 and 999, inclusive.<br>Up to 3 ASCII hex digits (including leading zeros) may be used to specify the duration.<br>Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Set the transition duration to 60 (3CH) frames in Layering Engine #0.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 4C 30 3A 54 3A 33 43 0D |
| | In ASCII | @ L0:T:3C<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.5 Layering Engine: Select Preset Crosspoint on Background Layer

Description: This command selects a crosspoint on the preset bus of the background layer in the Layering Engine.

Syntax: @<Space>ED:L<LE#>:P:<Src><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L'<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| P: | ASCII 'P' followed by ASCII ':' (3AH).<br>Identifies this as the Select Preset Crosspoint command. |
| <Src> | Source number.<br>Up to 4 characters (including leading zeros) may be used to specify the source.<br>Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Select crosspoint 11 (0BH) on the preset bus of the background layer in Layering Engine #1.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 4C 31 3A 50 3A 42 0D |
| | In ASCII | @ L1:P:B<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

## 6.4.6 Layering Engine: Perform a Cut Transition on the Background Layer

Description: This command performs a cut transition on the background layer in the Layering Engine.

Syntax: @<Space>ED:L<LE#>:C<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H). Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH). Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L' Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number. Valid ASCII hex digits are '0' or '1'. |
| C | ASCII 'C'. Identifies this as the Perform a Cut Transition on the Background Layer command. |
| <CR> | ASCII carriage return character (0DH). |

Example: Perform a cut transition on the background layer in Layering Engine #0. The key layers are not affected by this command.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 4C 30 3A 43 0D |
| | In ASCII | @ L0:C<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

## 6.4.7 Layering Engine: Perform a Mix Transition on the Background Layer

Description: This command performs a mix transition on the background layer in the Layering Engine.

Syntax: @<Space>ED:L<LE#>:M<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L'<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| M | ASCII 'M'.<br>Identifies this as the Perform a Mix Transition on the Background Layer command. |
| <CR> | ASCII carriage return character (0DH). |

Example: Perform a mix transition on the background layer in Layering Engine #0 using the current Transition Duration value. The key layers are not affected by this command.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 4C 30 3A 4D 0D |
| | In ASCII | @ L0:M<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.8 Layering Engine: Select Preset Crosspoint on Background Layer and Perform a Cut Transition

Description: This command selects a crosspoint on the preset bus in the Layering Engine and then performs a cut transition on the background layer.

Syntax: @<Space>ED:L<LE#>:PC:<Src><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H). Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH). Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L' Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number. Valid ASCII hex digits are '0' or '1'. |
| PC: | ASCII 'P' followed by ASCII 'C' followed by ASCII ':' (3AH). Identifies this as the Select Preset Crosspoint and Perform a Cut Transition command. |
| <Src> | Source number. Up to 4 characters (including leading zeros) may be used to specify the source. Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Select crosspoint 9 on the preset bus in Layering Engine #1 and then perform a cut transition on the background layer. The key layers are not affected by this command.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 4C 31 3A 50 43 3A 39 0D |
| | In ASCII | @ L1:PC:9<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.9 Layering Engine: Select Preset Crosspoint on Background Layer and Perform a Mix Transition

| | |
|---|---|
| Description: | This command selects a crosspoint on the preset bus in the Layering Engine and then performs a mix transition on the background layer. |
| Syntax: | @<Space>ED:L<LE#>:PM:<Src><CR> |

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H). Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH). Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L' Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number. Valid ASCII hex digits are '0' or '1'. |
| PM: | ASCII 'P' followed by ASCII 'M' followed by ASCII ':' (3AH). Identifies this as the Select Preset Crosspoint and Performs a Mix Transition command. |
| <Src> | Source number. Up to 4 characters (including leading zeros) may be used to specify the source. Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

| | | |
|---|---|---|
| Example: | Select crosspoint 9 on the preset bus in Layering Engine #1 and then perform a mix transition on the background layer using the current Transition Duration value. The key layers are not affected by this command. | |
| Request: | In Hex | 40 20 4C 31 3A 50 4D 3A 39 0D |
| | In ASCII | @ L1:PM:9<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

## 6.4.10 Layering Engine: Recall Keyer Preset

Description: This command assigns a keyer preset to a key layer in the Layering Engine.

Syntax: @<Space>ED:L<LE#>:K<Key#>:P:<Pst><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L'<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| K | ASCII 'K'.<br>Identifies this as one of the keyer commands. |
| <Key#> | Key layer number followed by ASCII ':' (3AH). Keyers are numbered starting at 0. A single character is used to specify the key layer number.<br>Any valid ASCII hex digit, depending on how many keyers the Layering Engine has. (For example, '0' to '9' or 'A' to 'F'). |
| P: | ASCII 'P' followed by ASCII ':' (3AH).<br>Identifies this as the Recall Keyer Preset command. |
| <Pst> | Identifies which preset to assign to the key layer.<br>Up to 4 characters (including leading zeros) may be used to specify the preset number. Presets are numbered starting at 0. Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Assign the 9th keyer preset to key layer #0 in Layering Engine #1.

**Note:** Presets are numbered starting at 0 in this command, so the 9th preset is identified as #8.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 4C 31 3A 4B 30 3A 50 3A 38 0D |
| | In ASCII | @ L1:K0:P:8<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.11 Layering Engine: Recall Logo

Description: This command assigns a logo to a key layer in the Layering Engine.

Syntax: @<Space>ED:L<LE#>:K<Key#>:L:<Logo><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L'<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| K | ASCII 'K'.<br>Identifies this as one of the keyer commands. |
| <Key#> | Key layer number followed by ASCII ':' (3AH). Keyers are numbered starting at 0. A single character is used to specify the key layer number.<br>Any valid ASCII hex digit, depending on how many keyers the Layering Engine has. (For example, '0' to '9' or 'A' to 'F'). |
| L: | ASCII 'L' followed by ASCII ':' (3AH).<br>Identifies this as the Recall Logo command. |
| <Logo> | Identifies which logo to assign to the key layer.<br>Up to 4 characters (including leading zeros) may be used to specify the logo number. Logos are numbered starting at 0.<br>Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Assign logo 13 (0DH) to key layer #0 in Layering Engine #1.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 4C 31 3A 4B 30 3A 4C 3A 44 0D |
| | In ASCII | @ L1:K0:L:D<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.12 Layering Engine: Logo Animation Run

Description: This command starts running the animation in the logo currently loaded in a keyer in the Layering Engine. If the currently loaded logo contains no animation, this command is ignored.

Syntax: @<Space>ED:L<LE#>:K<Key#>:LAR:<Logo><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L'<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| K | ASCII 'K'.<br>Identifies this as one of the keyer commands. |
| <Key#> | Key layer number followed by ASCII ':' (3AH). Keyers are numbered starting at 0. A single character is used to specify the key layer number.<br>Any valid ASCII hex digit, depending on how many keyers the Layering Engine has. (For example, '0' to '9' or 'A' to 'F'). |
| LAR: | ASCII 'LAR'<br>Identifies this as the Logo Animation Run command. |
| <CR> | ASCII carriage return character (0DH). |

Example: Run the animation in the currently loaded logo in key layer #1 in Layering Engine #0.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3A 4C 30 3A 4B 31 3A 4C 41 52 0D |
| | In ASCII | @ ED:L0:K1:LAR<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.13 Layering Engine: Logo Animation Stop

Description: This command stops a logo animation that is currently running in a keyer in the Layering Engine. If the currently loaded logo contains no animation, this command is ignored.

Syntax: @<Space>ED:L<LE#>:K<Key#>:LAS<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L'<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| K | ASCII 'K'.<br>Identifies this as one of the keyer commands. |
| <Key#>: | Key layer number followed by ASCII ':' (3AH). Keyers are numbered starting at 0. A single character is used to specify the key layer number.<br>Any valid ASCII hex digit, depending on how many keyers the Layering Engine has. (For example, '0' to '9' or 'A' to 'F'). |
| LAS: | ASCII "LAS".<br>Identifies this as the Logo Animation Stop command. |
| <CR> | ASCII carriage return character (0DH). |

Example: Stop the animation that is running in the currently loaded logo in key layer #1 in Layering Engine #0.

Request: In Hex 40 20 45 44 3A 4C 30 3A 4B 31 3A 4C 41 53 0D
In ASCII @ ED:L0:K1:LAS<CR>

Response: In Hex 3E
In ASCII '>'

### 6.4.14 Layering Engine: Logo Animation Loop

Description: This command starts a logo animation running and loops repeatedly to keep running the animation in a keyer in the Layering Engine. If the currently loaded logo contains no animation, this command is ignored.

Syntax: @<Space>ED:L<LE#>:K<Key#>:LAL<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H). Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH). Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L' Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number. Valid ASCII hex digits are '0' or '1'. |
| K | ASCII 'K'. Identifies this as one of the keyer commands. |
| <Key#>: | Key layer number followed by ASCII ':' (3AH). Keyers are numbered starting at 0. A single character is used to specify the key layer number. Any valid ASCII hex digit, depending on how many keyers the Layering Engine has. (For example, '0' to '9' or 'A' to 'F'). |
| LAL: | ASCII "LAL". Identifies this as the Logo Animation Loop command. |
| <CR> | ASCII carriage return character (0DH). |

Example: Loop to repeatedly run the animation in the currently loaded logo in key layer #1 in Layering Engine #0.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3A 4C 30 3A 4B 31 3A 4C 41 4C 0D |
| | In ASCII | @ ED:L0:K1:LAL<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.15 Layering Engine: Cut Key Layer On

Description: This command cuts the key layer on in the Layering Engine. If the key layer is already on, this command has no effect.

Syntax: @<Space>ED:L<LE#>:K<Key#>:CN<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L'<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| K | ASCII 'K'.<br>Identifies this as one of the keyer commands. |
| <Key#>: | Key layer number followed by ASCII ':' (3AH). Keyers are numbered starting at 0. A single character is used to specify the key layer number.<br>Any valid ASCII hex digit, depending on how many keyers the Layering Engine has. (For example, '0' to '9' or 'A' to 'F'). |
| CN | ASCII 'C' followed by ASCII 'N'.<br>Identifies this as the Cut Key Layer On command. |
| <CR> | ASCII carriage return character (0DH). |

Example: Cut key layer #1 on in Layering Engine #0.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 4C 30 3A 4B 31 3A 43 4E 0D |
| | In ASCII | @ L0:K1:CN<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.16 Layering Engine: Cut Key Layer Off

Description: This command cuts the key layer off in the Layering Engine. If the key layer is already off, this command has no effect.

Syntax: @<Space>ED:L<LE#>:K<Key#>:CF<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L'<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| K | ASCII 'K'.<br>Identifies this as one of the keyer commands. |
| <Key#>: | Key layer number followed by ASCII ':' (3AH). Keyers are numbered starting at 0. A single character is used to specify the key layer number.<br>Any valid ASCII hex digit, depending on how many keyers the Layering Engine has. (For example, '0' to '9' or 'A' to 'F'). |
| CF | ASCII 'C' followed by ASCII 'F'.<br>Identifies this as the Cut Key Layer Off command. |
| <CR> | ASCII carriage return character (0DH). |

Example: Cut key layer #1 off in Layering Engine #0.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 4C 30 3A 4B 31 3A 43 46 0D |
| | In ASCII | @ L0:K1:CF<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.17 Layering Engine: Mix Key Layer On

Description: This command mixes the key layer on in the Layering Engine using the current Transition Duration value. If the key layer is already on, this command has no effect.

Syntax: @<Space>ED:L<LE#>:K<Key#>:MN<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L'<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| K | ASCII 'K'.<br>Identifies this as one of the keyer commands. |
| <Key#>: | Key layer number followed by ASCII ':' (3AH). Keyers are numbered starting at 0. A single character is used to specify the key layer number.<br>Any valid ASCII hex digit, depending on how many keyers the Layering Engine has. (For example, '0' to '9' or 'A' to 'F'). |
| MN | ASCII 'M' followed by ASCII 'N'.<br>Identifies this as the Mix Key Layer On command. |
| <CR> | ASCII carriage return character (0DH). |

Example: Mix key layer #1 on in Layering Engine #0.

Request: In Hex 40 20 4C 30 3A 4B 31 3A 4D 4E 0D
In ASCII @ L0:K1:MN<CR>

Response: In Hex 3E
In ASCII '>'

### 6.4.18 Layering Engine: Mix Key Layer Off

Description: This command mixes the key layer off in the Layering Engine. If the key layer is already off, this command has no effect.

Syntax: @<Space>ED:L<LE#>:K<Key#>:MF<CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| L | ASCII 'L'<br>Identifies this as one of the "Layering Engine" commands. |
| <LE#>: | Layering Engine number followed by ASCII ':' (3AH). Layering Engines are numbered starting at 0. A single character is used to specify the Layering Engine number.<br>Valid ASCII hex digits are '0' or '1'. |
| K | ASCII 'K'.<br>Identifies this as one of the keyer commands. |
| <Key#>: | Key layer number followed by ASCII ':' (3AH). Keyers are numbered starting at 0. A single character is used to specify the key layer number.<br>Any valid ASCII hex digit, depending on how many keyers the Layering Engine has. (For example, '0' to '9' or 'A' to 'F'). |
| MF | ASCII 'M' followed by ASCII 'F'.<br>Identifies this as the Mix Key Layer Off command. |
| <CR> | ASCII carriage return character (0DH). |

Example: Mix key layer #1 off in Layering Engine #0.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 4C 30 3A 4B 31 3A 4D 46 0D |
| | In ASCII | @ L0:K1:MF<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.19 Recall an Action Register

Description: This command recalls an Action register. If the Action register is disabled or is not configured, this command will be ignored.

Syntax: @<Space>ED:ACT:<Action#><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H). Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH). Identifies this as a supplemental Ensemble Designs command. |
| ACT: | ASCII "ACT" followed by ASCII ':' (3AH). Identifies this as a Recall Action Register command. |
| <Action#> | Action register number. Action registers are numbered starting at 0. Up to 4 characters (including leading zeros) may be used to specify the Action register number. Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Recall the 8th Action register.

**Note:** Action registers are numbered starting at 0 in this command, so the 8th Action register is identified as #7.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3A 41 43 54 3A 37 0D |
| | In ASCII | @ ED:ACT:7<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.20 Recall a Salvo Register

Description: This command recalls a Salvo register. If the Salvo register is disabled or is not configured, this command will be ignored.

Syntax: @<Space>ED:SAL:<Salvo#><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| SAL: | ASCII "SAL" followed by ASCII ':' (3AH).<br>Identifies this as a Recall Salvo Register command. |
| <Salvo#> | Salvo register number. Salvo registers are numbered starting at 0. Up to 4 characters (including leading zeros) may be used to specify the Salvo register number.<br>Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Recall the 6th Salvo register.

**Note:** Salvo registers are numbered starting at 0 in this command, so the 6th Salvo register is identified as #5.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3A 53 41 4C 3A 35 0D |
| | In ASCII | @ ED:SAL:5<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.21 Assign a Multiviewer Layout

Description: This command assigns a predefined layout to a multiviewer. If the layout does not exist, this command will be ignored.

Syntax: @<Space>ED:MV<Mv#>:LAY:<LName><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| MV | ASCII "MV".<br>Identifies this as a Multiviewer command. |
| <Mv#>: | Multiviewer number followed by ASCII ':' (3AH). Multiviewers are numbered starting at 0. Up to 4 characters (including leading zeros) may be used to specify the Multiviewer number.<br>Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| LAY: | ASCII "LAY" followed by ASCII ':' (3AH).<br>Identifies this as an Assign Multiviewer Layout command. |
| <LName> | ASCII string identifying the name of the layout to be assigned to the multiviewer. |
| <CR> | ASCII carriage return character (0DH). |

Example: Assign the layout named "Grok" to Multiviewer #0.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3A 4D 56 30 3A 4C 41 59 3A 47 72 6F 6B 0D |
| | In ASCII | @ ED:MV0:LAY:Grok<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.22 Select Video Decoder Input

Description: This command selects the input to the video decoder.

Syntax: @<Space>ED:DEC:INP:<Input><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| DEC: | ASCII "DEC" followed by ASCII ':' (3AH).<br>Identifies this as a Video Decoder command. |
| INP: | ASCII "INP" followed by ASCII ':' (3AH).<br>Identifies this as a Video Decoder Input command. |
| <Input> | ASCII string identifying the input to assign to the decoder.<br>There are 3 possible choices:<br>"UNI" selects the Unicast input.<br>"MUL" selects the Multicast input.<br>"DVB" selects the DVB-ASI input. |
| <CR> | ASCII carriage return character (0DH). |

Example: Select the Unicast input to the video decoder.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3A 44 45 43 3A 49 4E 50 3A 55 4E 49 0D |
| | In ASCII | @ ED:DEC:INP:UNI<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.23 Set Video Bit Rate for Encoder

Description: This command selects the video bit rate for the High Resolution video encoder. The Low Resolution encoder is unaffected.

Syntax: @<Space>ED:ENC:VBR:<BitRate><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| ENC: | ASCII "ENC" followed by ASCII ':' (3AH).<br>Identifies this as a Video Encoder command. |
| VBR: | ASCII "VBR" followed by ASCII ':' (3AH).<br>Identifies this as an Encoder Video Bit Rate command. |
| <BitRate> | Video Bit Rate in kilobytes/seconds. Up to 4 characters (including leading zeros) may be used to specify the video bit rate.<br>Any valid ASCII hex digits. (For example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Set the high resolution video encoder's bit rate to 4000 (FA0H) kilobytes/second.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3A 45 4E 43 3A 56 42 52 3A 46 41 30 0D |
| | In ASCII | @ ED:ENC:VBR:FA0<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.24 Set Enable for Multicast Stream

Description: This command enables or disables the multicast video stream.

Syntax: @<Space>ED:STRM:MUL:ENAB:<on/off><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| STRM: | ASCII "STRM" followed by ASCII ':' (3AH).<br>Identifies this as a Stream command. |
| MUL: | ASCII "MUL" followed by ASCII ':' (3AH).<br>Identifies this as a Multicast Stream command. |
| ENAB: | ASCII "ENAB" followed by ASCII ':' (3AH).<br>Identifies this as a Multicast Stream Enable command. |
| <on/off> | An ASCII '0' disables the multicast stream.<br>An ASCII '1' enables the multicast stream. |
| <CR> | ASCII carriage return character (0DH). |

Example: Enable the multicast stream.

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3A 53 54 52 4D 3A 4D 55 4C 3A 45 4E 41 42 3A 31 0D |
| | In ASCII | @ ED:STRM:MUL:ENAB:1<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.25 Set IP Address for Multicast Stream

Description: This command sets the multicast video stream’s IP address.

Syntax: @<Space>ED:STRM:MUL:IP:<ipAddr><CR>

| | |
|---|---|
| @<Space> | ASCII ‘@’ (40H) followed by ASCII space character ‘ ’ (20H). Identifies the start of a command. |
| ED: | ASCII “ED” followed by ASCII ‘:’ (3AH). Identifies this as a supplemental Ensemble Designs command. |
| STRM: | ASCII “STRM” followed by ASCII ‘:’ (3AH). Identifies this as a Stream command. |
| MUL: | ASCII “MUL” followed by ASCII ‘:’ (3AH). Identifies this as a Multicast Stream command. |
| IP: | ASCII “IP” followed by ASCII ‘:’ (3AH). Identifies this as a Multicast Stream IP Address command. |
| <ipAddr> | Any ASCII string with a valid IP Address. |
| <CR> | ASCII carriage return character (0DH). |

Example: Set the multicast stream’s IP address to 123.456.789.12.

Request: In Hex

40 20 45 44 3A 53 54 52 4D 3A 4D 55 4C 3A 49 50 3A 31 32 33 2E 34 35
36 2E 37 38 39 2E 31 32 0D

In ASCII @ ED:STRM:MUL:IP:123.456.789.12<CR>

Response: In Hex 3E
In ASCII ‘>’

### 6.4.26 Set Port Number for Multicast Stream

Description: This command sets the multicast video stream’s port number.

Syntax: @<Space>ED:STRM:MUL:PORT:<port#><CR>

| | |
|---|---|
| @<Space> | ASCII ‘@’ (40H) followed by ASCII space character ‘ ’ (20H).<br>Identifies the start of a command. |
| ED: | ASCII “ED” followed by ASCII ‘:’ (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| STRM: | ASCII “STRM” followed by ASCII ‘:’ (3AH).<br>Identifies this as a Stream command. |
| MUL: | ASCII “MUL” followed by ASCII ‘:’ (3AH).<br>Identifies this as a Multicast Stream command. |
| PORT: | ASCII “PORT” followed by ASCII ‘:’ (3AH).<br>Identifies this as a Multicast Stream Port Number command. |
| <port#> | Up to 4 characters (including leading zeros) may be used to specify the port number.<br>Any valid ASCII hex digits. (For example, ‘0’ to ‘9’ or ‘A’ to ‘F’). |
| <CR> | ASCII carriage return character (0DH). |

Example: Set the multicast stream’s port number to 5678 (162EH).

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3A 53 54 52 4D 3A 4D 55 4C 3A 50 4F 52 54 3A 31 36 32 45 0D |
| | In ASCII | @ ED:STRM:MUL:PORT:162E<CR> |
| Response: | In Hex | 3E |
| | In ASCII | ‘>’ |

### 6.4.27 Set Enable for Unicast A Stream

Description: This command enables or disables the video stream for Unicast A.

Syntax: @<Space>ED:STRM:UNIA:ENAB:<on/off><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H). Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH). Identifies this as a supplemental Ensemble Designs command. |
| STRM: | ASCII "STRM" followed by ASCII ':' (3AH). Identifies this as a Stream command. |
| UNIA: | ASCII "UNIA" followed by ASCII ':' (3AH). Identifies this as a Unicast A Stream command. |
| ENAB | ASCII "ENAB" followed by ASCII ':' (3AH). Identifies this as a Unicast A Stream Enable command. |
| <on/off> | An ASCII '0' disables Unicast A stream. An ASCII '1' enables Unicast A stream. |
| <CR> | ASCII carriage return character (0DH). |

Example: Enable the Unicast A stream.

Request: In Hex 40 20 45 44 3A 53 54 52 4D 3A 55 4E 49 41 3A 45 4E 41 42 3A 31 0D

In ASCII @ ED:STRM:UNIA:ENAB:1<CR>

Response: In Hex 3E
In ASCII '>'

### 6.4.28 Set IP Address for Unicast A Stream

Description: This command sets the IP address for the Unicast A video stream.

Syntax: @<Space>ED:STRM:UNIA:IP:<ipAddr><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H). Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH). Identifies this as a supplemental Ensemble Designs command. |
| STRM: | ASCII "STRM" followed by ASCII ':' (3AH). Identifies this as a Stream command. |
| UNIA: | ASCII "UNIA" followed by ASCII ':' (3AH). Identifies this as a Unicast A Stream command. |
| IP: | ASCII "IP" followed by ASCII ':' (3AH). Identifies this as a Unicast A Stream IP Address command. |
| <ipAddr> | Any ASCII string with a valid IP Address. |
| <CR> | ASCII carriage return character (0DH). |

Example: Set Unicast A stream's IP address to 123.456.789.12.

Request: In Hex 40 20 45 44 3A 53 54 52 4D 3A 55 4E 49 41 3A 49 50 3A 31 32 33 2E 34 35 36 2E 37 38 39 2E 31 32 0D

In ASCII @ ED:STRM:UNIA:IP:123.456.789.12<CR>

Response: In Hex 3E
In ASCII '>'

### 6.4.29 Set Port Number for Unicast A Stream

Description: This command sets the port number for the Unicast A video stream.

Syntax: @<Space>ED:STRM:UNIA:PORT:<port#><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| STRM: | ASCII "STRM" followed by ASCII ':' (3AH).<br>Identifies this as a Stream command. |
| UNIA: | ASCII "UNIA" followed by ASCII ':' (3AH).<br>Identifies this as a Unicast A Stream command. |
| PORT: | ASCII "PORT" followed by ASCII ':' (3AH).<br>Identifies this as a Unicast A Stream Port Number command. |
| <port#> | Up to 4 characters (including leading zeros) may be used to specify the port number.<br>Any valid ASCII hex digits. (E.g. '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Set Unicast A stream's port number to 5678 (162EH).

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3A 53 54 52 4D 3A 55 4E 49 41 3A 50 4F 52 54 3A 31 36 32 45 0D |
| | In ASCII | @ ED:STRM:UNIA:PORT:162E<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |

### 6.4.30 Set Enable for Unicast B Stream

Description: This command enables or disables the video stream for Unicast B.

Syntax: @<Space>ED:STRM:UNIB:ENAB:<on/off><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| STRM: | ASCII "STRM" followed by ASCII ':' (3AH).<br>Identifies this as a Stream command. |
| UNIB: | ASCII "UNIB" followed by ASCII ':' (3AH).<br>Identifies this as a Unicast B Stream command. |
| ENAB: | ASCII "ENAB" followed by ASCII ':' (3AH).<br>Identifies this as a Unicast B Stream Enable command. |
| <on/off> | An ASCII '0' disables Unicast B stream.<br>An ASCII '1' enables Unicast B stream. |
| <CR> | ASCII carriage return character (0DH). |

Example: Enable the Unicast B stream.

Request: In Hex 40 20 45 44 3A 53 54 52 4D 3A 55 4E 49 42 3A 45 4E 41 42 3A 31 0D

In ASCII @ ED:STRM:UNIB:ENAB:1<CR>

Response: In Hex 3E
In ASCII '>'

### 6.4.31 Set IP Address for Unicast B Stream

Description: This command sets the IP address for the Unicast B video stream.

Syntax: @<Space>ED:STRM:UNIB:IP:<ipAddr><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H). Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH). Identifies this as a supplemental Ensemble Designs command. |
| STRM: | ASCII "STRM" followed by ASCII ':' (3AH). Identifies this as a Stream command. |
| UNIB: | ASCII "UNIB" followed by ASCII ':' (3AH). Identifies this as a Unicast B Stream command. |
| IP: | ASCII "IP" followed by ASCII ':' (3AH). Identifies this as a Unicast B Stream IP Address command. |
| <ipAddr> | Any ASCII string with a valid IP Address. |
| <CR> | ASCII carriage return character (0DH). |

Example: Set Unicast B stream's IP address to 123.456.789.12.

Request: In Hex 40 20 45 44 3A 53 54 52 4D 3A 55 4E 49 42 3A 49 50 3A 31 32 33 2E 34 35 36 2E 37 38 39 2E 31 32 0D

In ASCII @ ED:STRM:UNIB:IP:123.456.789.12<CR>

Response: In Hex 3E
In ASCII '>'

### 6.4.32 Set Port Number for Unicast B Stream

Description: This command sets the port number for the Unicast B video stream.

Syntax: @<Space>ED:STRM:UNIB:PORT:<port#><CR>

| | |
|---|---|
| @<Space> | ASCII '@' (40H) followed by ASCII space character ' ' (20H).<br>Identifies the start of a command. |
| ED: | ASCII "ED" followed by ASCII ':' (3AH).<br>Identifies this as a supplemental Ensemble Designs command. |
| STRM: | ASCII "STRM" followed by ASCII ':' (3AH).<br>Identifies this as a Stream command. |
| UNIB: | ASCII "UNIB" followed by ASCII ':' (3AH).<br>Identifies this as a Unicast B Stream command. |
| PORT: | ASCII "PORT" followed by ASCII ':' (3AH).<br>Identifies this as a Unicast B Stream Port Number command. |
| <port#> | Up to 4 characters (including leading zeros) may be used to specify the port number.<br>Any valid ASCII hex digits (for example, '0' to '9' or 'A' to 'F'). |
| <CR> | ASCII carriage return character (0DH). |

Example: Set Unicast B stream's port number to 5678 (162EH).

| | | |
|---|---|---|
| Request: | In Hex | 40 20 45 44 3A 53 54 52 4D 3A 55 4E 49 42 3A 50 4F 52 54 3A 31 36 32 45 0D |
| | In ASCII | @ ED:STRM:UNIB:PORT:162E<CR> |
| Response: | In Hex | 3E |
| | In ASCII | '>' |