RS-232 Control
of the
Advantage DRI

Biamp Systems, 14130 N.W. Science Park, Portland, Oregon 97229 U.S.A. (503) 641-7287
an affiliate of Rauland-Borg Corp.

Introduction

This document contains technical information relating to computer control of the Biamp Advantage DRI Digital Remote Interface. This information is intended for advanced users - in particular for those who wish to develop their own computer programs to control the Advantage DRI. It is assumed that the reader is an experienced programmer and has some familiarity with standard programming practices, binary and hexadecimal numbers, the ASCII character set, asynchronous serial data communications, and RS-232 interfaces.

Hexadecimal, ASCII-Hex, and "Pseudo-Hex" Numbers

Throughout this document, hexadecimal numbers shall be represented by preceding the number with "0x". For example: the hexadecimal equivalent of the decimal number 255 is 0xFF.

Individual ASCII characters, except control characters, will be enclosed in single quotes. For example: the ASCII character 'A' has the hexadecimal value 0x41. The ASCII "carriage return" control character shall be represented as ↵. An ASCII code chart is included with this document for your convenience.

When an 8-bit binary data value is being transmitted over a serial data communications link, it is a common practice to transmit the byte as two "ASCII-hex" characters - one character represents the most significant nibble of the data value and the other character represents the least significant nibble (a nibble is 4-bits; half of a byte). Each ASCII-hex character is in the range of '0' thru '9' or 'A' thru 'F' (from the ASCII code chart, 0x30 thru 0x39 or 0x41 thru 0x46). For example, the *hexadecimal* equivalent of the decimal number 61 is 0x3D. To transmit this in an *ASCII-hex* format, first transmit the ASCII character '3' (whose hex value is 0x33), followed by the ASCII character 'D' (whose hex value is 0x44). This is the standard way to transmit an ASCII-hex value. In some cases, the lower case characters 'a' thru 'f' (0x61 thru 0x66) are accepted in addition to the upper case letters 'A' thru 'F'.

The Advantage DRI does not utilize standard ASCII-hex format. The DRI computer commands implement what I call "pseudo-hex" notation (also sometimes referred to as a "poor man's" ASCII-hex notation). Instead of representing the hexadecimal value using the ASCII characters '0' thru '9' and 'A', 'B', 'C', 'D', 'E', and 'F', the pseudo-hex format uses the ASCII characters '0' thru '9' and the characters '**:**', '**;**', '<', '=', '>', and '**?**'. As you can see by studying the ASCII code chart, a binary nibble may be converted to its equivalent pseudo-hex character by simply adding 0x30.

A binary/decimal/hexadecimal/pseudo-hex conversion chart is included at the end of this manual for your convenience.

Data Communications Parameters

The Advantage DRI has a rear-panel dip-switch which allows you to set its baud rate to either 2400, 9600, or 19200 bits per second. The factory default setting is 9600. The Advantage DRI communicates with 8 data bits, no parity, and 1 stop bit. The Advantage DRI utilizes a subset of the standard 7-bit ASCII character set.

The Advantage DRI utilizes interrupt-driven communications firmware with a 32-character receive buffer. When the Advantage DRI receives a command, it may become "busy" for as long as 200 milliseconds while it is executing the command. During this time, the Advantage DRI can successfully receive up to 32 additional characters. If more than 32 characters are sent to the Advantage DRI while it is busy executing a command, its input buffer will be overrun and some characters will be "lost".

To avoid overrunning the Advantage DRI's input buffer, it is recommended that some form of flow control be implemented. Neither hardware (DTR or RTS) nor XON/XOFF (also known as DC1/DC3 or control-S/control-Q) handshaking is supported by the Advantage DRI. However, each character which the Advantage DRI receives through its serial port will be "echoed" back to the computer. A simple form of flow control may be implemented by simply waiting for each character's echo from the Advantage DRI before transmitting the next character, since the Advantage DRI doesn't retrieve and echo an incoming character until it has finished processing the previous character.

Computer Control

The Advantage DRI has an RS-232-compatible serial port which allows it to be controlled by a computer or by a system controller such as those provided by AMX or Crestron. The Advantage DRI offers the following two methods of computer control:

- Control Button Emulation. This method of computer control allows the computer to emulate Biamp's standard infrared remote control transmitter or wall-mount remote control panel. Using this method, the computer outputs ASCII characters which are equivalent to the characters which would be generated by a remote control connected to the Advantage DRI. These ASCII characters are transmitted from the computer to the Advantage DRI's serial port. Control Button Emulation is simple to perform, however, it only provides "one-way" control of the Advantage DRI - it allows the computer to send simple commands *to* the Advantage DRI, but it does not provide any mechanism for requesting status information *from* the Advantage DRI.

- Advanced Computer Control. This method of computer control provides advanced commands which allow "two-way" control of the Advantage DRI. Using Advanced Computer Control commands, the computer may request status information *from* the device as well as send commands *to* the device. The computer communicates with the Advantage DRI using the Advantage DRI's serial port.

Control Button Emulation

Control Button Emulation is the simplest form of computer control of the Advantage DRI. This method of operation allows the computer to emulate the operation of a standard Biamp remote control transmitter.

For each button on a standard Biamp remote control, there is a corresponding ASCII character. In order to emulate a remote control button, the computer simply transmits the corresponding ASCII character to the Advantage DRI's serial port. Each character received by the Advantage DRI will be echoed back to the computer.

The standard Biamp remote control devices never exceed a transmission rate of 9 characters per second. If the computer wishes to perform Control Button Emulation at a rate of greater than 20 characters per second (50 msec per character), flow control should be implemented by waiting for the echo of each character before transmitting the next character. At slower speeds, flow control should not be necessary.

The following table summarizes the ASCII character codes for Control Button Emulation corresponding to each of the 40 remote control buttons supported by the Advantage DRI. These button codes are also summarized on the ASCII code chart provided at the end of this manual. The remote control buttons on the standard Biamp transmitter are numbered from left to right going from bottom to top with the lower left-hand button being button number 1.

Control Button Emulation ASCII Codes (with factory default button definitions shown)

button 1	'B'	(0x42)	Vol Down ch.5	button 21	'V'	(0x56)	Toggle Mute ch.1
button 2	'C'	(0x43)	Vol Down ch.6	button 22	'W'	(0x57)	Toggle Mute ch.2
button 3	'D'	(0x44)	Vol Down ch.7	button 23	'X'	(0x58)	Toggle Mute ch.3
button 4	'E'	(0x45)	Vol Down Main output	button 24	'Y'	(0x59)	Toggle Mute ch.4
button 5	'F'	(0x46)	Vol Up ch.5	button 25	'Z'	(0x5A)	Recall Preset 1
button 6	'G'	(0x47)	Vol Up ch.6	button 26	'['	(0x5B)	Recall Preset 2
button 7	'H'	(0x48)	Vol Up ch.7	button 27	'\'	(0x5C)	Recall Preset 3
button 8	'I'	(0x49)	Vol Up Main output	button 28	']'	(0x5D)	Recall Preset 4
button 9	'J'	(0x4A)	Toggle Mute ch.5	button 29	'^'	(0x5E)	Recall Preset 5
button 10	'K'	(0x4B)	Toggle Mute ch.6	button 30	'_'	(0x5F)	Recall Preset 6
button 11	'L'	(0x4C)	Toggle Mute ch.7	button 31	'`'	(0x60)	Recall Preset 7
button 12	'M'	(0x4D)	Toggle Mute Main output	button 32	'b'	(0x62)	Recall Preset 8
button 13	'N'	(0x4E)	Vol Down ch.1	button 33	'c'	(0x63)	Vol Down ch.8
button 14	'O'	(0x4F)	Vol Down ch.2	button 34	'd'	(0x64)	Vol Up ch.8
button 15	'P'	(0x50)	Vol Down ch.3	button 35	'e'	(0x65)	Toggle Mute ch.8
button 16	'Q'	(0x51)	Vol Down ch.4	button 36	'f'	(0x66)	Vol Down Aux1 out ch.11
button 17	'R'	(0x52)	Vol Up ch.1	button 37	'g'	(0x67)	Vol Up Aux1 out ch.11
button 18	'S'	(0x53)	Vol Up ch.2	button 38	'h'	(0x68)	Toggle Mute Aux1 ch.11
button 19	'T'	(0x54)	Vol Up ch.3	button 39	'i'	(0x69)	Vol Down Aux In ch.9
button 20	'U'	(0x55)	Vol Up ch.4	button 40	'j'	(0x6A)	Vol Up Aux In ch.9

Device Select Prefix Characters

When using Advanced Computer Control, up to sixty-four Advantage DRIs may be linked together and individually controlled by the computer (a rear-panel dip-switch is used to set a unique device number to each unit). When using Control Button Emulation, a limited subset of device addressing may be performed, which allows individual control of up to four Advantage DRIs (with device numbers 1 thru 4). This is accomplished by transmitting a device select prefix code immediately prior to each control button ASCII character code. The device select prefix code is inspected by each device to determine whether or not the device should react to the button code which immediately follows. If a button code is not immediately preceded by a device select prefix character, then all Advantage DRIs in the system will react to that button code. The following table summarizes the ASCII characters to use for selecting various devices. This information is also summarized in the ASCII code chart provided at the end of this manual.

Device Select Prefix Codes

select device 1	'l'	(0x6C)
select device 2	'm'	(0x6D)
select devices 1 & 2	'n'	(0x6E)
select device 3	'o'	(0x6F)
select devices 1 & 3	'p'	(0x70)
select devices 2 & 3	'q'	(0X71)
select devices 1 & 2 & 3	'r'	(0x72)
select device 4	's'	(0x73)
select devices 1 & 4	't'	(0x74)
select devices 2 & 4	'u'	(0x75)
select devices 1 & 2 & 4	'v'	(0x76)
select devices 3 & 4	'w'	(0x77)
select devices 1 & 3 & 4	'x'	(0x78)
select devices 2 & 3 & 4	'y'	(0x79)
select devices 1 & 2 & 3 & 4	'z'	(0x7A)

Advanced Computer Control

The Advanced Computer Control command set includes commands which allow the Advantage DRI to return information about the system to the computer, unlike Control Button Emulation which is basically a one-way control mechanism. The following list summarizes the commands available using Advanced Computer Control, including the ASCII command character associated with each command:

#	do-volume-action	perform the specified volume action (up, down, mute, etc).
$	define-preset-mix	define a preset mix and store it in non-volatile memory.
%	get-preset-mix	retrieve the preset mix settings for the specified preset.
&	get/set-volume	get or set the volume for the specified channel.
(	do-logic-action	perform the specified logic output action (on, off, toggle).
)	do-preset-action	perform the specified preset action (recall, store).
/	get-version	retrieve the model information and firmware version date.

Each Advanced Computer Control command requires at least two parameter bytes (four pseudo-hex characters) to be sent prior to the command character. Each command will be explained in detail on the following pages.

Some of the commands cause the Advantage DRI to return information to the computer. For each string of information returned to the computer, the Advantage DRI terminates the string by transmitting the ASCII carriage return character (0x0D - represented in this document as ↵).

As mentioned earlier, the Advantage DRI will echo all characters it receives, regardless of whether or not the characters are valid commands or parameters. Characters greater than 0x7F are reserved and should not be transmitted by the computer. The Advantage DRI utilizes a subset of the standard ASCII character set. The following characters have meaning to the Advantage DRI:

character	hexadecimal	operation
ASCII control characters	(0x00 - 0x1F)	no operation
ASCII **SPACE** character	(0x20)	no operation
! thru **/**	(0x21 - 0x2F)	Advanced Computer Control commands
0 thru **?**	(0x30 - 0x3F)	pseudo-hex parameters for Advanced Computer Control commands
@	(0x40)	Control Button Emulation Repeat Code
A	(0x41)	no operation
B thru **`**	(0x42 - 0x60)	Control Button Emulation commands (buttons 01 - 31)
a	(0x61)	no operation
b thru **j**	(0x62 - 0x6A)	Control Button Emulation commands (buttons 32 - 40)
k thru **z**	(0x6B - 0x7A)	Control Button Emulation Device Select Prefix commands
{ thru **DEL**	(0x7B - 0x7F)	no operation
0x80 thru 0xFF	(0x80 - 0xFF)	RESERVED

An ASCII code chart showing all Advantage DRI commands and codes is provided later in this document for your convenience. Please note that all ASCII *control* characters (including space, tab, carriage return, and line feed) will be echoed by the Advantage DRI and then ignored - they have no special significance to the Advantage DRI.

Device Type Bitmask and Device Number Parameters

In a system which has more than one Advantage product connected to the computer, the Device Type Bitmask and Device Number parameters provide a mechanism for the computer to address a command to a particular device (or a combination of devices). Every command in the Advanced Computer Control command set requires that a Device Type Bitmask and a Device Number be transmitted as the last two parameter bytes before the computer transmits the command character itself. These two parameter bytes provide a device addressing capability to specify which of the devices in the system should execute the command. All devices which are not specifically addressed by these two command parameters will ignore the command.

The Device Type Bitmask parameter byte for the Advantage DRI is always 0x80. This Device Type Bitmask is also shared with other Biamp devices, including the Advantage EQ281/8, Advantage EQ282/8, and Advantage SMS200. It is anticipated that all new devices developed by Biamp will utilize 0x80 as their Device Type Bitmask. Other Device Type Bitmasks currently in use are:

0x01 Biamp Advantage DRC 4+4 digital remote control
0x02 Biamp Advantage EQ28X digitally-controlled graphic equalizer
0x04 Biamp Advantage SPM522D stereo preamp/mixer
0x08 Biamp Advantage PMX84 programmable matrix switch

The Device Number parameter byte provides support for up to sixty-four devices, numbered 0 thru 63 (‘00’ thru ‘3?’ in pseudo-hex). The Advanced Computer Control command structure also supports a “broadcast” command, which allows you to send a command to ***all*** devices which match the Device Type Bitmask. To send a “broadcast” command, specify 255 (‘??’ in pseudo-hex) as the Device Number parameter.

Each Advantage DRI’s device number is assigned using rear-panel dip-switches. Except for broadcast commands, any particular Advantage DRI will respond to an Advanced Computer Control command only if bit 7 of the Device Type Bitmask is a ‘1’ ***and*** the Device Number parameter matches its dip-switch settings.

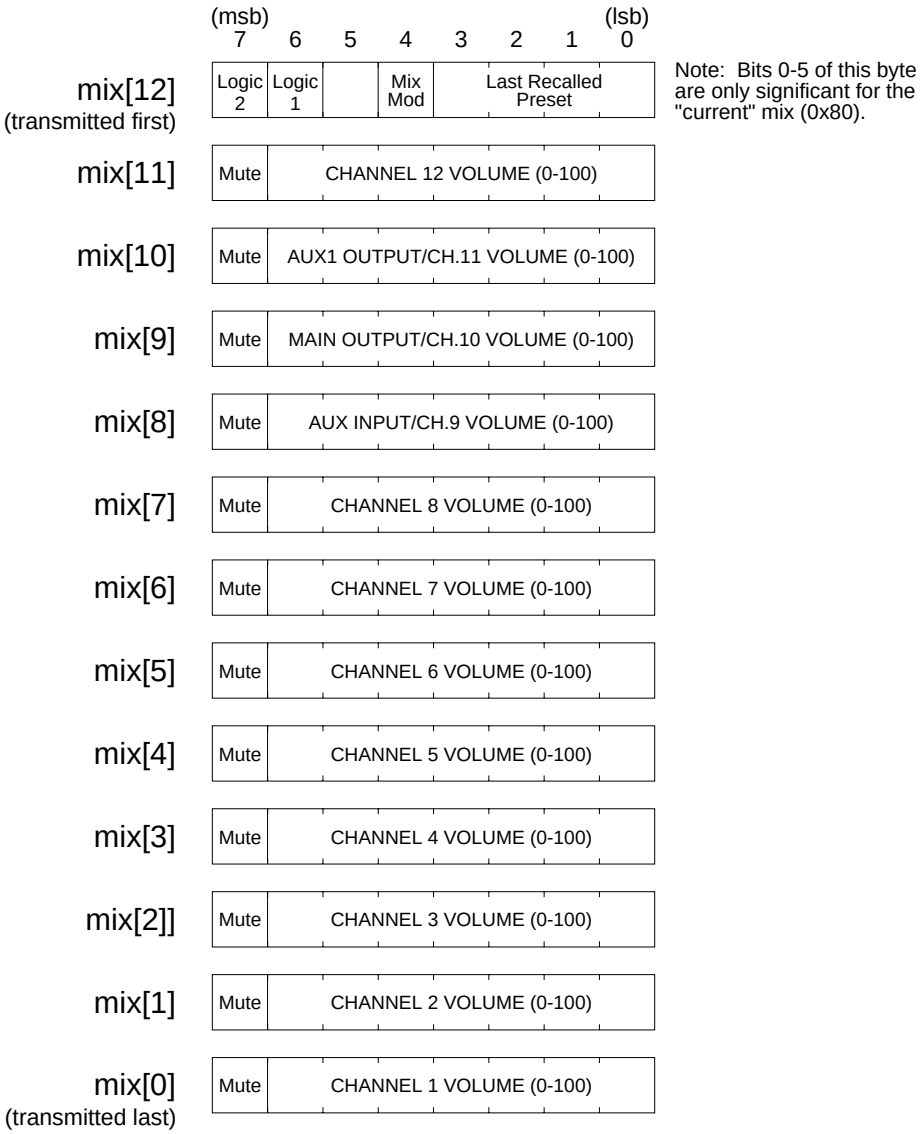
Advanced Computer Control Data Structures

Preset Mix Status Data Structure

The data structure used for preset mixes consists of an array of 13 bytes (which are transmitted as 26 pseudo-hex characters). Mix[0] through mix[11] represent the volume level and mute status of channels 1 through 12. Mix[12] represents the status of the logic outputs, and also has other special meanings for the "current" mix.

Advantage DRI Preset Mix Data Structure

13 -byte array with elements numbered mix[0] thru mix[12]



The volumes specified in mix[0] thru mix[11] have a range of 0 (minimum volume) thru 100 (maximum volume, ‘64’ in pseudo-hex). The Advantage DRI only recognizes even-numbered volume steps - if an odd number is specified for a volume, the Advantage DRI will round it down to the next lower even number. For example, volume level 75 and volume level 74 are exactly the same as far as the Advantage DRI is concerned.

Seven data bits are required to specify a number in the range of 0 to 100. The eighth data bit (bit 7) is used to indicate the mute status for each channel. If bit 7 is a ‘1’, the channel is muted (set to minimum volume) and the volume level specified in bits 0 thru 6 specify what volume level the channel will return to if it receives an un-mute command.

Bit 6 of mix[12] indicates the status of logic output number 1 - a '1' in this bit means the logic output is on, and a '0' in this bit means the logic output is off. When the Advantage DRI is used to control and Advantage DLA93, logic output 1 connects to the DLA93’s master/slave logic input. Turning off logic output 1 puts the DLA93 into master mode and turning on logic output 1 puts the DLA93 into slave mode.

Bit 7 of mix[12] indicates the status of logic output number 2 - a '1' in this bit means the logic output is on, and a '0' in this bit means the logic output is off. When the Advantage DRI is used to control and Advantage DLA93, logic output 2 connects to the DLA93’s automix/manual logic input. Turning off logic output 2 puts the DLA93 into automix (gated) mode and turning on logic output 2 puts the DLA93 into manual (non-gated) mode.

When the preset mix being specified is the "current" mix (mix number 0x80), bits 0 thru 3 of mix[12] indicate which preset mix was the last mix recalled and bit 4 of mix[12] indicates whether or not the current mix has been modified since the last time a preset mix has been recalled.

Advanced Computer Control Command Notation

For the following descriptions of the Advanced Computer Control command set, the following conventions will be used. Each ASCII character which represents a pseudo-hex nibble will be shown in *italics*, with the following letters representing certain types of parameters:

- *a* a pseudo-hex nibble specifying an action code.
- *d* one of the pseudo-hex nibbles in the device number which indicates which device the command is addressed to.
- *m* one of the pseudo-hex nibbles occurring in the preset mix data structure.
- *n* a pseudo hex nibble specifying a channel number, logic output number, or preset mix number.
- *v* one of the pseudo-hex nibbles specifying a volume level.

do-volume-action

Description:

The do-volume-action command causes the Advantage DRI to perform the specified volume action (volume up, volume down, toggle mute, mute, or un-mute) for the specified channel.

Syntax of Command:

*an*80*dd*#

where:

a	=	Volume Action: 0 = NOP, 1 = vol down, 2 = vol up, 3 = toggle mute, 4 = mute, 5 = un-mute.
n	=	Channel Number (pseudo-hex): 1 - 8 = channel 1 - 8. 9 = Aux Input. **:** = Main Output. **;** = Aux1 Output. < = ch 12.
80	=	Device Type Bitmask (pseudo-hex)
dd	=	Device Number (pseudo-hex)
#	=	do-volume-action command character (0x23)

Syntax of Response:

(no response)

Example:

2:8001#

This example causes Advantage DRI number 1 to increase the main output volume by one step.

Comments:

The Advantage DRI supports 51 discrete volume levels or steps: 0 (minimum volume) thru 100 (maximum volume) in increments of 2. On a 0 - 100 scale, each volume up or volume down command increments or decrements the volume by 2 (one step).

The typical execution time for the do-volume-action command is 5 milliseconds (not including the time required to transmit the command string).

$ define-preset-mix

Description:

The define-preset-mix command provides a new definition for the specified preset mix. The new preset mix is stored in the Advantage DRI's non-volatile memory. This command does not change the "current mix" (it does not immediately change any of the volumes or logic outputs).

Syntax of Command:

*mmmmmmmmmmmmmmmmmmmmmmmmmmmmnn*80*dd***$**

where:

mmmm...mmmm	=	Preset Mix Data Structure (26 pseudo-hex chars) (see page 8)
nn	=	Preset Mix Number (pseudo-hex): 01 thru 08 for presets 1 thru 8.
80	=	Device Type Bitmask (pseudo-hex)
dd	=	Device Number (pseudo-hex)
$	=	define-preset-mix command character (0x24)

Syntax of Response:

(no response)

Example:

400000640000003264640000640480 01$

This example causes Advantage DRI number 1 to redefine its preset mix number 4 to have logic output 1 on, logic output 2 off, channels 1, 4, 5, and main output at maximum volume, channel 6 at 50%, and all other volumes at minimum.

Comments:

The typical execution time for the define-preset-mix command is 150 milliseconds (not including the time required to transmit the command string).

% get-preset-mix-settings

Description:

The get-preset-mix-settings command causes the Advantage DRI to return the settings of the specified preset mix. The preset mix settings will be returned in the Preset Mix Data Structure format. To retrieve the "current mix" settings, specify 0x80 as the preset mix number.

Syntax of Command:

*nn*80*dd***%**

where:

nn	=	Preset Mix Number (pseudo-hex) 01 thru 08 for presets 1 thru 8. 80 for the "current mix".
80	=	Device Type Bitmask (pseudo-hex)
dd	=	Device Number (pseudo-hex)
%	=	get-preset-mix-settings command character (0x25)

Syntax of Response:

mmmmmmmmmmmmmmmmmmmmmmmmmm↵

where:

mmmm...mmmm	=	Preset Mix Data Structure (26 pseudo-hex chars) (see page 8)

Example:

command:
028001%

response:
400000640000003264640000 64↵

This example causes Advantage DRI number 1 to retrieve its settings for preset mix number 2. In this example, preset mix number 2's definition was: logic output 1 on, logic output 2 off, channels 1, 4, 5, and main output at maximum volume, channel 6 at 50%, and all other volumes at minimum.

Comments:

& set-volume

Description:

The set-volume command causes the Advantage DRI to set the specified channel to a new volume level. The volume level is a 7-bit value which ranges from 0 (minimum volume) to 100 (maximum volume, 0x64). The eighth data bit (bit 7) of the volume byte indicates whether or not that particular channel should be muted. When bit 7 indicates that the channel should be muted (bit 7 = '1'), the 7 least significant bits (bits 0 - 6) indicate the volume level which will be restored by the Advantage DRI if that channel subsequently becomes un-muted.

Syntax of Command:

*vvnn*80*dd***&**

where:

vv	=	Volume level (pseudo-hex)
nn	=	Channel Number (pseudo-hex) 1 - 8 = channel 1 - 8. 9 = Aux Input. **:** = Main Output. **;** = Aux1 Output. < = ch 12.
80	=	Device Type Bitmask (pseudo-hex)
dd	=	Device Number Bitmask (pseudo-hex)
&	=	set-volume command character (0x26)

Syntax of Response:

(no response)

Example:

4<0:8001&

This example causes Advantage DRI number 1 to set its main output volume (channel 10, '0:' in pseudo-hex) to 76% of full travel (volume step 76, '4<' in pseudo-hex).

Comments:

The Advantage DRI supports 51 discrete volume levels or steps: 0 (minimum volume) thru 100 (maximum volume) in increments of 2. Any odd-numbered volume setting will be truncated to the next lowest even number. For example, as far as the Advantage DRI is concerned, specifying a volume of 75 is no different from specifying a volume of 74.

The typical execution time for the set-volume command is 200 microseconds (not including the time required to transmit the command string).

& get-volume

Description:

The get-volume command causes the Advantage DRI to return the current volume setting for the specified channel. The volume level is a 7-bit value which ranges from 0 (minimum volume) to 100 (0x64 - maximum volume). The eighth bit (bit 7) of the volume byte indicates whether or not that particular channel is muted. When bit 7 indicates that the channel is muted (bit 7 = '1'), the 7 least significant bits (bits 0 - 6) indicate the volume level which will be restored by the Advantage DRI if that channel subsequently becomes un-muted.

Syntax of Command:

??*nn*80*dd***&**

where:

??	=	0xff in pseudo-hex (command specifier)
nn	=	Channel Number (pseudo-hex) 1 - 8 = channel 1 - 8. 9 = Aux Input. **:** = Main Output. **;** = Aux1 Output. < = ch 12.
80	=	Device Type Bitmask (pseudo-hex)
dd	=	Device Number Bitmask (pseudo-hex)
&	=	get-volume command character (0x26)

Syntax of Response:

vv↵

where:

vv	=	Volume (in pseudo-hex) of the specified channel.

Example:

command:
??028001&

response:
64↵

This example causes Advantage DRI number 1 to return the volume setting for channel 2. In this example, the volume is set at 100% (64 in pseudo-hex).

Comments:

(do-logic-action

Description:

The do-logic-action command causes the Advantage DRI to perform the specified logic output action (turn on, turn off, toggle).

Syntax of Command:

*an*80*dd***(**

where:

a	=	Logic Output Action: 0 = NOP (no operation), 1 = turn off, 2 = turn on, 3 = toggle.
n	=	Logic Output Number (1 or 2)
80	=	Device Type Bitmask (pseudo-hex)
dd	=	Device Number Bitmask (pseudo-hex)
(	=	do-logic-action command character (0x28)

Syntax of Response:

(no response)

Example:

218001(

This example causes Advantage DRI number 1 to turn on logic output number 1.

Comments:

When the Advantage DRI is connected to an Advantage DLA93, logic output number 1 corresponds to the master/slave function of the DLA93 (off = master, on = slave) and logic output number 2 corresponds to the automix/manual function of the DLA93 (off = automix, on = manual mix).

The typical execution time for the do-logic-action command is 300 microseconds (not including the time required to transmit the command string).

) do-preset-action

Description:

The do-preset-action command causes the Advantage DRI to perform the specified preset action (recall a preset mix or store the current mix as a preset). The Advantage DRI supports eight normal preset mixes (1 thru 8) plus a "temporary" preset mix (preset number 0). Action code 3 allows you to store the current settings as preset 0 (the temporary mix) and recall one of the eight normal presets. You can then, sometime later, restore the settings to the way they were by recalling preset 0.

Syntax of Command:

*an*80*dd***)**

where:

a	=	Preset Action: 0 = NOP (no operation), 1 = recall, 2 = store, 3 = store temporary mix then recall.
n	=	Preset Number (0 through 8)
80	=	Device Type Bitmask (pseudo-hex)
dd	=	Device Number Bitmask (pseudo-hex)
)	=	do-preset-action command character (0x29)

Syntax of Response:

(no response)

Example:

`178001)`

This example causes Advantage DRI number 1 to recall preset mix number 7.

Comments:

The typical execution time for the do-preset-action command is 30 milliseconds for recalling a preset, 175 milliseconds for storing a preset, and 200 milliseconds for storing the temporary preset then recalling (not including the time required to transmit the command string).

/ get-version

Description:

The get-version command causes the Advantage DRI to return its model identifier code and firmware version to the computer. The model identifier string is always '10' for the Advantage DRI. The firmware version number is simply the release date of the firmware, in the format of *mmddyy*. These date values are decimal digits, not pseudo-hex notation. For example, December 31, 1997 would be represented as **123197**.

Syntax of Command:

0080*dd*/

where:

00	=	'00' (command specifier)
80	=	Device Type Bitmask (pseudo-hex)
dd	=	Device Number Bitmask (pseudo-hex)
/	=	get-version command character (0x2F)

Syntax of Response:

10*mmddyy*↵

where:

10	=	Advantage DRI model identifier
mm	=	2-digit decimal month number
dd	=	2-digit decimal day of the month
yy	=	2-digit decimal year number

Example:

command:
008001/

response:
10070297↵

This example causes device number 1 to return its model I.D. and firmware version. In this example, device number 1 is an Advantage DRI and its firmware version date is July 2, 1997.

Advanced Computer Command Summary

*an*80*dd*#	do-volume-action
*mmmmmmmmmmmmmmmmmmmmmmmmmmmmnn*80*dd*$	define-preset-mix
*nn*80*dd*%	get-preset-mix-settings
*vvnn*80*dd*&	set-volume
??*nn*80*dd*&	get-volume
*an*80*dd*(	do-logic-action
*an*80*dd*)	do-preset-action
0080*dd*/	get-version

a a pseudo-hex nibble specifying an action code.
d one of the pseudo-hex nibbles in the device number which indicates which device the command is addressed to.
m one of the pseudo-hex nibbles occurring in the preset mix data structure.
n a pseudo hex nibble specifying a channel number, logic output number, or preset mix number.
v one of the pseudo-hex nibbles specifying a volume level.

.

ASCII Code Chart

with Decimal & Hexadecimal Equivalents and Advantage DRI Commands

000. 0x00	016. 0x10	032. 0x20	048. 0x30	064. 0x40	080. 0x50	096. 0x60	112. 0x70
NUL	**DLE**	**(space)**	**0** nibble 0x0	**@** repeat code	**P** button 15	**`** button 31	**p** select 1,3
001. 0x01	017. 0x11	033. 0x21	049. 0x31	065. 0x41	081. 0x51	097. 0x61	113. 0x71
SOH	**DC1**	**!** vol limits	**1** nibble 0x1	**A**	**Q** button 16	**a**	**q** select 2,3
002. 0x02	018. 0x12	034. 0x22	050. 0x32	066. 0x42	082. 0x52	098. 0x62	114. 0x72
STX	**DC2**	**"** do-button	**2** nibble 0x2	**B** button 01	**R** button 17	**b** button 32	**r** select 1,2,3
003. 0x03	019. 0x13	035. 0x23	051. 0x33	067. 0x43	083. 0x53	099. 0x63	115. 0x73
ETX	**DC3**	**#** do-volume	**3** nibble 0x3	**C** button 02	**S** button 18	**c** button 33	**s** select 4
004. 0x04	020. 0x14	036. 0x24	052. 0x34	068. 0x44	084. 0x54	100. 0x64	116. 0x74
EOT	**DC4**	**$** define-preset	**4** nibble 0x4	**D** button 03	**T** button 19	**d** button 34	**t** select 1,4
005. 0x05	021. 0x15	037. 0x25	053. 0x35	069. 0x45	085. 0x55	101. 0x65	117. 0x75
ENQ	**NAK**	**%** get-preset	**5** nibble 0x5	**E** button 04	**U** button 20	**e** button 35	**u** select 2,4
006. 0x06	022. 0x16	038. 0x26	054. 0x36	070. 0x46	086. 0x56	102. 0x66	118. 0x76
ACK	**SYN**	**&** get/set-volume	**6** nibble 0x6	**F** button 05	**V** button 21	**f** button 36	**v** select 1,2,4
007. 0x07	023. 0x17	039. 0x27	055. 0x37	071. 0x47	087. 0x57	103. 0x67	119. 0x77
BEL	**ETB**	**'**	**7** nibble 0x7	**G** button 06	**W** button 22	**g** button 37	**w** select 3,4
008. 0x08	024. 0x18	040. 0x28	056. 0x38	072. 0x48	088. 0x58	104. 0x68	120. 0x78
BS	**CAN**	**(** do-logic	**8** nibble 0x8	**H** button 07	**X** button 23	**h** button 38	**x** select 1,3,4
009. 0x09	025. 0x19	041. 0x29	057. 0x39	073. 0x49	089. 0x59	105. 0x69	121. 0x79
HT	**EM**	**)** do-preset	**9** nibble 0x9	**I** button 08	**Y** button 24	**i** button 39	**y** select 2,3,4
010. 0x0A	026. 0x1A	042. 0x2A	058. 0x3A	074. 0x4A	090. 0x5A	106. 0x6A	122. 0x7A
LF	**SUB**	***** get-status	**:** nibble 0xA	**J** button 09	**Z** button 25	**j** button 40	**z** select 1,2,3,4
011. 0x0B	027. 0x1B	043. 0x2B	059. 0x3B	075. 0x4B	091. 0x5B	107. 0x6B	123. 0x7B
VT	**ESC**	**+** sleep 10 sec.	**;** nibble 0xB	**K** button 10	**[** button 26	**k** select none	**{**
012. 0x0C	028. 0x1C	044. 0x2C	060. 0x3C	076. 0x4C	092. 0x5C	108. 0x6C	124. 0x7C
FF	**FS**	**,** read memory	**<** nibble 0xC	**L** button 11	**** button 27	**l** select 1	**¦**
013. 0x0D	029. 0x1D	045. 0x2D	061. 0x3D	077. 0x4D	093. 0x5D	109. 0x6D	125. 0x7D
CR	**GS**	**-** write memory	**=** nibble 0xD	**M** button 12	**]** button 28	**m** select 2	**}**
014. 0x0E	030. 0x1E	046. 0x2E	062. 0x3E	078. 0x4E	094. 0x5E	110. 0x6E	126. 0x7E
SO	**RS**	**.** set defaults	**>** nibble 0xE	**N** button 13	**^** button 29	**n** select 1,2	**~**
015. 0x0F	031. 0x1F	047. 0x2F	063. 0x3F	079. 0x4F	095. 0x5F	111. 0x6F	127. 0x7F
SI	**US**	**/** get version	**?** nibble 0xF	**O** button 14	**_** button 30	**o** select 3	**DEL**

binary	decimal	hex	pseudo
0000 0000	0.	0x00	00
0000 0001	1.	0x01	01
0000 0010	2.	0x02	02
0000 0011	3.	0x03	03
0000 0100	4.	0x04	04
0000 0101	5.	0x05	05
0000 0110	6.	0x06	06
0000 0111	7.	0x07	07
0000 1000	8.	0x08	08
0000 1001	9.	0x09	09
0000 1010	10.	0x0a	0:
0000 1011	11.	0x0b	0;
0000 1100	12.	0x0c	0<
0000 1101	13.	0x0d	0=
0000 1110	14.	0x0e	0>
0000 1111	15.	0x0f	0?
0001 0000	16.	0x10	10
0001 0001	17.	0x11	11
0001 0010	18.	0x12	12
0001 0011	19.	0x13	13
0001 0100	20.	0x14	14
0001 0101	21.	0x15	15
0001 0110	22.	0x16	16
0001 0111	23.	0x17	17
0001 1000	24.	0x18	18
0001 1001	25.	0x19	19
0001 1010	26.	0x1a	1:
0001 1011	27.	0x1b	1;
0001 1100	28.	0x1c	1<
0001 1101	29.	0x1d	1=
0001 1110	30.	0x1e	1>
0001 1111	31.	0x1f	1?
0010 0000	32.	0x20	20
0010 0001	33.	0x21	21
0010 0010	34.	0x22	22
0010 0011	35.	0x23	23
0010 0100	36.	0x24	24
0010 0101	37.	0x25	25
0010 0110	38.	0x26	26
0010 0111	39.	0x27	27
0010 1000	40.	0x28	28
0010 1001	41.	0x29	29
0010 1010	42.	0x2a	2:
0010 1011	43.	0x2b	2;
0010 1100	44.	0x2c	2<
0010 1101	45.	0x2d	2=
0010 1110	46.	0x2e	2>
0010 1111	47.	0x2f	2?
0011 0000	48.	0x30	30
0011 0001	49.	0x31	31
0011 0010	50.	0x32	32
0011 0011	51.	0x33	33
0011 0100	52.	0x34	34
0011 0101	53.	0x35	35
0011 0110	54.	0x36	36
0011 0111	55.	0x37	37
0011 1000	56.	0x38	38
0011 1001	57.	0x39	39
0011 1010	58.	0x3a	3:
0011 1011	59.	0x3b	3;
0011 1100	60.	0x3c	3<
0011 1101	61.	0x3d	3=
0011 1110	62.	0x3e	3>
0011 1111	63.	0x3f	3?

binary	decimal	hex	pseudo
0100 0000	64.	0x40	40
0100 0001	65.	0x41	41
0100 0010	66.	0x42	42
0100 0011	67.	0x43	43
0100 0100	68.	0x44	44
0100 0101	69.	0x45	45
0100 0110	70.	0x46	46
0100 0111	71.	0x47	47
0100 1000	72.	0x48	48
0100 1001	73.	0x49	49
0100 1010	74.	0x4a	4:
0100 1011	75.	0x4b	4;
0100 1100	76.	0x4c	4<
0100 1101	77.	0x4d	4=
0100 1110	78.	0x4e	4>
0100 1111	79.	0x4f	4?
0101 0000	80.	0x50	50
0101 0001	81.	0x51	51
0101 0010	82.	0x52	52
0101 0011	83.	0x53	53
0101 0100	84.	0x54	54
0101 0101	85.	0x55	55
0101 0110	86.	0x56	56
0101 0111	87.	0x57	57
0101 1000	88.	0x58	58
0101 1001	89.	0x59	59
0101 1010	90.	0x5a	5:
0101 1011	91.	0x5b	5;
0101 1100	92.	0x5c	5<
0101 1101	93.	0x5d	5=
0101 1110	94.	0x5e	5>
0101 1111	95.	0x5f	5?
0110 0000	96.	0x60	60
0110 0001	97.	0x61	61
0110 0010	98.	0x62	62
0110 0011	99.	0x63	63
0110 0100	100.	0x64	64
0110 0101	101.	0x65	65
0110 0110	102.	0x66	66
0110 0111	103.	0x67	67
0110 1000	104.	0x68	68
0110 1001	105.	0x69	69
0110 1010	106.	0x6a	6:
0110 1011	107.	0x6b	6;
0110 1100	108.	0x6c	6<
0110 1101	109.	0x6d	6=
0110 1110	110.	0x6e	6>
0110 1111	111.	0x6f	6?
0111 0000	112.	0x70	70
0111 0001	113.	0x71	71
0111 0010	114.	0x72	72
0111 0011	115.	0x73	73
0111 0100	116.	0x74	74
0111 0101	117.	0x75	75
0111 0110	118.	0x76	76
0111 0111	119.	0x77	77
0111 1000	120.	0x78	78
0111 1001	121.	0x79	79
0111 1010	122.	0x7a	7:
0111 1011	123.	0x7b	7;
0111 1100	124.	0x7c	7<
0111 1101	125.	0x7d	7=
0111 1110	126.	0x7e	7>
0111 1111	127.	0x7f	7?

binary	decimal	hex	pseudo
1000 0000	128.	0x80	80
1000 0001	129.	0x81	81
1000 0010	130.	0x82	82
1000 0011	131.	0x83	83
1000 0100	132.	0x84	84
1000 0101	133.	0x85	85
1000 0110	134.	0x86	86
1000 0111	135.	0x87	87
1000 1000	136.	0x88	88
1000 1001	137.	0x89	89
1000 1010	138.	0x8a	8:
1000 1011	139.	0x8b	8;
1000 1100	140.	0x8c	8<
1000 1101	141.	0x8d	8=
1000 1110	142.	0x8e	8>
1000 1111	143.	0x8f	8?
1001 0000	144.	0x90	90
1001 0001	145.	0x91	91
1001 0010	146.	0x92	92
1001 0011	147.	0x93	93
1001 0100	148.	0x94	94
1001 0101	149.	0x95	95
1001 0110	150.	0x96	96
1001 0111	151.	0x97	97
1001 1000	152.	0x98	98
1001 1001	153.	0x99	99
1001 1010	154.	0x9a	9:
1001 1011	155.	0x9b	9;
1001 1100	156.	0x9c	9<
1001 1101	157.	0x9d	9=
1001 1110	158.	0x9e	9>
1001 1111	159.	0x9f	9?
1010 0000	160.	0xa0	:0
1010 0001	161.	0xa1	:1
1010 0010	162.	0xa2	:2
1010 0011	163.	0xa3	:3
1010 0100	164.	0xa4	:4
1010 0101	165.	0xa5	:5
1010 0110	166.	0xa6	:6
1010 0111	167.	0xa7	:7
1010 1000	168.	0xa8	:8
1010 1001	169.	0xa9	:9
1010 1010	170.	0xaa	::
1010 1011	171.	0xab	:;
1010 1100	172.	0xac	:<
1010 1101	173.	0xad	:=
1010 1110	174.	0xae	:>
1010 1111	175.	0xaf	:?
1011 0000	176.	0xb0	;0
1011 0001	177.	0xb1	;1
1011 0010	178.	0xb2	;2
1011 0011	179.	0xb3	;3
1011 0100	180.	0xb4	;4
1011 0101	181.	0xb5	;5
1011 0110	182.	0xb6	;6
1011 0111	183.	0xb7	;7
1011 1000	184.	0xb8	;8
1011 1001	185.	0xb9	;9
1011 1010	186.	0xba	;:
1011 1011	187.	0xbb	;;
1011 1100	188.	0xbc	;<
1011 1101	189.	0xbd	;=
1011 1110	190.	0xbe	;>
1011 1111	191.	0xbf	;?

binary	decimal	hex	pseudo
1100 0000	192.	0xc0	<0
1100 0001	193.	0xc1	<1
1100 0010	194.	0xc2	<2
1100 0011	195.	0xc3	<3
1100 0100	196.	0xc4	<4
1100 0101	197.	0xc5	<5
1100 0110	198.	0xc6	<6
1100 0111	199.	0xc7	<7
1100 1000	200.	0xc8	<8
1100 1001	201.	0xc9	<9
1100 1010	202.	0xca	<:
1100 1011	203.	0xcb	<;
1100 1100	204.	0xcc	<<
1100 1101	205.	0xcd	<=
1100 1110	206.	0xce	<>
1100 1111	207.	0xcf	<?
1101 0000	208.	0xd0	=0
1101 0001	209.	0xd1	=1
1101 0010	210.	0xd2	=2
1101 0011	211.	0xd3	=3
1101 0100	212.	0xd4	=4
1101 0101	213.	0xd5	=5
1101 0110	214.	0xd6	=6
1101 0111	215.	0xd7	=7
1101 1000	216.	0xd8	=8
1101 1001	217.	0xd9	=9
1101 1010	218.	0xda	=:
1101 1011	219.	0xdb	=;
1101 1100	220.	0xdc	=<
1101 1101	221.	0xdd	==
1101 1110	222.	0xde	=>
1101 1111	223.	0xdf	=?
1110 0000	224.	0xe0	>0
1110 0001	225.	0xe1	>1
1110 0010	226.	0xe2	>2
1110 0011	227.	0xe3	>3
1110 0100	228.	0xe4	>4
1110 0101	229.	0xe5	>5
1110 0110	230.	0xe6	>6
1110 0111	231.	0xe7	>7
1110 1000	232.	0xe8	>8
1110 1001	233.	0xe9	>9
1110 1010	234.	0xea	>:
1110 1011	235.	0xeb	>;
1110 1100	236.	0xec	><
1110 1101	237.	0xed	>=
1110 1110	238.	0xee	>>
1110 1111	239.	0xef	>?
1111 0000	240.	0xf0	?0
1111 0001	241.	0xf1	?1
1111 0010	242.	0xf2	?2
1111 0011	243.	0xf3	?3
1111 0100	244.	0xf4	?4
1111 0101	245.	0xf5	?5
1111 0110	246.	0xf6	?6
1111 0111	247.	0xf7	?7
1111 1000	248.	0xf8	?8
1111 1001	249.	0xf9	?9
1111 1010	250.	0xfa	?:
1111 1011	251.	0xfb	?;
1111 1100	252.	0xfc	?<
1111 1101	253.	0xfd	?=
1111 1110	254.	0xfe	?>
1111 1111	255.	0xff	??