Serial Control
of the
Advantage VRAM

Biamp Systems, 10074 S.W. Arctic Drive, Beaverton, Oregon 97005 U.S.A. (503) 641-7287
an affiliate of Rauland-Borg Corp.

Introduction

This document contains information for the serial control of the Advantage VRAM (Variable Resource Auto Mixer) and the Advantage VRAMeq, (Variable Resource Auto Mixer with equalizer). Specifically, this document tries to inform those looking to write their own software controls for the Advantage VRAM. It is assumed that the reader has some familiarity with standard programming practices, binary and hexadecimal numbers, the ASCII character set, asynchronous serial data connections, and RS-232 interfaces.

Decimal, Binary, and "Pseudo-hex" Numbers

This document uses three different numerical notations. The first, the most common, is the decimal notation. Whenever it is used, a "d" will appear after the number..

8 Bit binary numbers are the second format used in this paper. These numbers will be followed by "b" after their usage. If a specific bit is being referred to, the numbers will be preceded by the word "bit."

To transmit an 8 bit binary number to the Advantage VRAM, hexadecimal notation is used. Hexadecimal numbers are arrived at by splitting the number into two halves. One half consists of the first four binary digits (most significant nibble) while the other consists of the last four binary digits (least significant nibble). 2 nibbles form a byte, which takes on a decimal value of 0 to 255. Each half is then assigned a hexadecimal value. Since the binary values range from 0 to 15, usually values from 10 to 15 are given the alphabetic letters from A to F.

However, the Advantage VRAM does not utilize standard hex format. Instead, the Advantage VRAM uses what is known as "pseudo-hex." Simply put, instead of using the letters A, B, C, D, E and F the Advantage VRAM uses : ; < = > and ?, respectively. All it takes to arrive at the new notation for hex values 10 to 15d is to add 30 to the old ASCII values. In this paper, [pseudo-hex] will appear after the use of a pseudo-hex character. The changes are traditional hex are summed up below:

	Nibble Conversion		
Decimal	Hex	Pseudo-hex	Binary
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	8	1000
9	9	9	1001
10	A	:	1010
11	B	;	1011
12	C	<	1100
13	D	=	1101
14	E	>	1110
15	F	?	1111

Serial Interface - Data Communications Parameters

The Advantage VRAM communicates through its serial port at four different baud rates: 2400, 9600, 19200, and 38400. The factory default setting is 9600 baud. Changing this rate is accomplished in the advanced mode (see page 18, not a recommended procedure) or through BiampWin. The Advantage VRAM communicates with 8 data bits, no parity, and 1 stop bit. The Advantage VRAM utilizes a subset of the standard 7-bit ASCII character set.

Control

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

- Control Button Emulation. This method of control emulates Biamp's standard infrared remote control transmitter or wall-mount remote control panel. Using this method, single ASCII characters sent to the device's serial port cause the device to behave as if a biamp remote controller were attached. While Control Button Emulation is simple to perform, it only provides basic and "one-way" control of the Advantage VRAM - it allows the user to send simple commands *to* the Advantage VRAM, but it does not provide any mechanism for requesting status information *from* the Advantage VRAM.

- Advanced Control. Advanced control provides a command set which allow "two-way" control of the Advantage VRAM. Using Advanced Control commands, a system may request status information *from* the device as well as send commands *to* the device. Communication occurs with the Advantage VRAM using the Advantage VRAM's serial port.

Control Button Emulation

Control Button Emulation is the simplest form of serial control of the Advantage VRAM. This method of operation allows the user 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 transmitting system simply transmits the corresponding ASCII character to the Advantage VRAM's serial port. Each character received by the Advantage VRAM will be echoed back out the serial port.

The standard Biamp remote control devices never exceed a transmission rate of 9 characters per second. If the controlling system 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 VRAM. 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.

Using BiampWin, it is possible to program the VRAM to respond to these commands.

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

Simple vs Addressable

The simple method of control button emulation is to send any one of the control button characters through the serial port to the VRAM. The disadvantage to this method is that every device hooked into the VRAM will also hear the command. If any of the other devices have been programmed with this particular character, they will also respond.

To avoid this problem, the VRAM allows addressable control button emulation. By using the control-button-emulation command, on page 12, control button commands are sent directly to a specific device.

Advanced Control

The Advanced Control command set includes more powerful commands to allow more flexible control of the Advantage VRAM. Unlike Control Button Emulation (which is basically a one-way control mechanism) advanced control commands allow the VRAM to return information through the serial port,. The following list summarizes the commands available using Advanced Control, including the ASCII command character associated with each command:

!	store-as-preset *(save settings as preset)*
"	retrieve-preset *(put Advantage VRAM into preset mode)*
#	read-device-settings *(read current settings from device memory*)
$	write-settings *(write to device memory)*
&	addressable-control-button-emulation *(execute control buttons)*
(	bitwise-operator *(perform bitwise operations on memory locations)*
)	increment-decrement-memory *(change memory location value by plus or minus one)*
*	polling-status *(request status update of various functions)*
+	sleep-for-10-seconds *(sleep for 10 seconds, ignoring all communication)*
,	read-eeprom-locations *(read from non-volatile memory)*
-	write-eeprom-locations[1] *(write to non-volatile memory)*
.	set-baud[1] *(set communications speed)*
/	get-version *(retrieve the model information and firmware version date)*

Each Advanced 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 VRAM to return information through the serial port. For each string of information returned to the serial port, the Advantage VRAM terminates the string by transmitting the ASCII carriage return character (0x0D - represented in this document as ↵).

[1] Not recommended, but available for use

As mentioned earlier, the Advantage VRAM 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 to the serial port. The Advantage VRAM utilizes a subset of the standard ASCII character set. The following characters have meaning to the Advantage VRAM:

character	hexadecimal	operation
ASCII control characters	(0x00 - 0x1F)	no operation
ASCII **SPACE** character	(0x20)	no operation
! thru **/**	(0x21 - 0x2F)	Advanced Control commands
0 thru **?**	(0x30 - 0x3F)	pseudo-hex parameters for Advanced 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

Device Type Bitmask, Device Number Bitmask, and Device Model Bitmask

In a system which has more than one Advantage product connected together, the Device Type Bitmask and Device Number Bitmask command parameters provide a mechanism to individually address a particular device (or a combination of devices). Every command in the Advanced Control command set requires that a Device Type Bitmask and a Device Number Bitmask be transmitted as the last two parameter bytes before transmitting the command character itself. These two bitmask parameters 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 bitmask values will ignore the command.

The Device Type Bitmask parameter byte supports up to eight distinct device types - one bit per device type. The eight device types are:

0x01 [hex] (bit 0) Biamp Advantage DRC 4+4 digital remote control
0x02 [hex] (bit 1) Biamp Advantage EQ28X digitally-controlled graphicEQ
0x04 [hex] (bit 2) Biamp Advantage SPM522D stereo preamp/mixer
0x08 [hex] (bit 3) Biamp Advantage PMX84 programmable matrix switch
0x10 [hex] (bit 4) (reserved for future product)
0x20 [hex] (bit 5) (reserved for future product)
0x40 [hex] (bit 6) (reserved for future product)
0x80 [hex] (bit 7) Advanced Products, such as the Biamp Advantage VRAM

The Advantage VRAM will only respond to Advanced Control commands if bit 7 of the Device Type Bitmask parameter byte is a '1'. A command may be directed to more than one device type in the system by setting all of the corresponding bits in the Device Type Bitmask to '1's. If only advanced equipment is being addressed (EQ2828/8 DRI, MSP, and DDL12) 80 is the only bitmask required to use.

The Device Number Bitmask parameter byte supports up to sixty-four distinct device numbers:

0x00 [hex]	Select Device Number 0
0x01 [hex]	Select Device Number 1
0x02 [hex]	Select Device Number 2
0xFF [hex]	Select Device Number 63

A particular Advantage VRAM will only respond to Advanced Control commands if the Device Number Bitmask parameter byte corresponds to its own device number.

For instance, the bitmask 8007 serves to talk only to advanced product (**80**) number 7 (**07**).

! store-as-preset

Description:

The Advantage VRAM and Advantage VRAMeq each allow up to 17 different presets. Using the store-as-preset command, the user is allowed to store the current settings (device configurations) under a specified preset.

Syntax of Command:

*pp*80*dd*!

where

pp	=	Preset number (0 to 16d; **00** to **10** [pseudo-hex])
80	=	Device type bitmask for Advantage Advantage VRAM
dd	=	Device number bitmask (1 to 63d; **00** to **3?** [pseudo-hex])
!	=	store-as-preset command character

Syntax of response:

no response

Example:

command:	response:
0?8002!	***(none)***

This example causes the Advantage VRAM , device number 2, to store the current settings under preset number 15d (**0?** [pseudo-hex]).

Comments:

While there are 16 designated presets on the main control screen in BiampWin, it is also possible to access and write to the power-up preset, **00**. This preset is used by the VRAM at power-up to load its startup configuration.

However, the current settings at power-down are normally saved to this preset. Consult the user's manual (regarding BiampWin) if you wish to disable saving of current settings at power-down.

" retrieve-preset

Description:

The retrieve-preset command configures the Advantage VRAM and Advantage VRAMeq according to a preset definition in non-volatile memory. The user can retrieve any of the 17 available presets.

Syntax of Command:

*pp*80*dd*"

where

pp	=	Preset number (1 to 16d; **00** to **10** [pseudo-hex])
80	=	Device type bitmask
dd	=	Device number bitmask (1 to 63d; to **3?** [pseudo-hex])
"	=	retrieve-preset command character

Syntax of response:

no response

Example:

command:	response:
108003!	***(none)***

This example configures the Advantage VRAM , device number 3, according to the settings stored in preset number 16d (**10** [psuedo-hex].

Comments:

Depending on how the VRAM is configured from BiampWin, recalling preset 0 will either recall the default power-up configuration or recall the state of the VRAM at the last power-down. Please consult the BiampWin user's manual for more information

read-current-device-settings

Description:

The Advantage VRAM stores the settings of its pre-amp, volume, logic outputs and other miscellaneous configuration data in 96 bytes of data. The Advantage VRAMeq also stores equalizer data in this area of memory. The read-device-settings command can be used to retrieve the contents of these memory locations.

Syntax of Command:

*nnaa*80*dd*#

nn	=	Number of bytes to read (limited by starting address; 1 to 96d; **01** to **60** [pseudo-hex])
aa	=	Starting memory address (0 to 95d; **00** to **5?** [pseudo-hex])
80	=	Device type bitmask for Advantage VRAM
dd	=	Device number bitmask (1 to 63d; **00** to **3?** [pseudo-hex])
#	=	read-device-setting command character

Syntax of response:

xx…(up to 96 data values)…↵

where

xx	=	Data value

Example:

command:	response:
10008002#	**642800001000001000??00??03?<03?<**

In this example, a Advantage VRAM (device number 2) is queried for the contents of the first 16d (**10** [pseudo-hex]) memory locations.

Comments:

From the beginning of the data structure (byte **00**), bytes 0-15d are miscellaneous settings. Bytes 16-31d are logic output settings, and bytes 32-95d are pre-amp settings, volume, and equalizer settings. See the memory map for exact details of memory mapping of device functions.

$ write-current-device-settings

Description:

When used in conjunction with the read-device-settings command, the write-device-settings command allows the user to manually adjust any aspect of the Advantage VRAM or Advantage VRAMeq settings.

Syntax of Command:

*xx…(up to 16 data values)…nnaa*80*dd*$

where

xx	=	Up to 16 data values, sent in reverse order, highest memory address first
nn	=	Number of bytes to write (limited by starting address; 1 to 96d; **01** to **60** [pseudo-hex])
aa	=	Starting memory address (0 to 95d; **00** to **5?** [pseudo-hex])
80	=	Device type bitmask for Advantage VRAM
dd	=	Device number bitmask (1 to 63d; **00** to **3?** [pseudo-hex])
$	=	write-device-setting command character

Syntax of response:

no response

Example:

command:	response:
919191032880 02$	***(none)***

This command causes a Advantage VRAM (device number 2) to write **03** bytes, **919191** [pseudo-hex], to setting location 40d (**28** [pseudo-hex]).

Comments:

The increment-decrement-memory command “)” can provide a simpler way of modifying a device setting by a single step, especially for settings that require the increasing or decreasing of a value

& addressable-control-button-emulation

Description:

The Advantage VRAM and Advantage VRAMeq can be controlled by a 40 button standard IR remote control that sends single ASCII characters. These characters are then echoed to all linked devices with control ports. Using addressable-control-button emulation allows the user to send control button emulation commands to a specific device.

Sending buttons 41- 48 and 49-56 simulates a logic input instead of a control button. Note that these buttons are not available on the remote control.

Syntax of Command:

*ee*80*dd*&

where

ee	=	Button to emulate (1 to 40d; **01** to **28** [pseudo-hex])
80	=	Device type bitmask
dd	=	Device number bitmask (1 to 63d; **00** to **3?** [pseudo-hex])
&	=	control-button-emulation command character

Syntax of response:

no response

Example:

command:	response:
018001&	***(none)***

This command tells the Advantage VRAM (device number 1) to emulate remote control button number 1.

Comments:

BiampWin provides the easiest method of entering button and logic input definitions.

(**bitwise-operator (firmware dates 7/23/98 and later)**

Description:

Many of the settings available on the Advantage VRAM are controlled by the status of individual bits in the device settings. To adjust one of these bits (for instance to mute or un-mute a channel) , use the bitwise-operator command.

Syntax of Command:

*vvssttaa*80*dd*)

where

vv	=	Bits to clear (**00** to **??** [pseudo-hex]; 00 indicates nothing to clear)
ss	=	Bits to set (**00** to **??** [pseudo-hex]; 00 indicates nothing to set)
tt	=	Bits to toggle (**00** to **??** [pseudo-hex]; 00 indicates nothing to toggle)
aa	=	Memory address (0 to 95d**; 00** to **5?** [pseudo-hex])
80	=	Device type bitmask for Advantage VRAM
dd	=	Device number bitmask (1 to 63d; **00** to **3?** [pseudo-hex])
(	=	bitwise-operator command character

Syntax of response:

no response

Example:

command:	response:
0000803<8007(	***(none)***

Here the mute bit of the main volume control (memory location **3<** [pseudo-hex]) is toggled.

command	response
0040003<8007(	***(none)***

This example sets the phantom power on for channel 1.

Comments:

It is easiest to think of the settings in binary, using the data from the memory map notes, and then convert the setting to pseudo-hex.

) increment-decrement-memory (firmware dates 7/23/98 and later)

Description:

Sometimes it is desired to adjust a value in device memory by increasing or decreasing it one step. A common application of this would be to adjust the main or auxiliary volume.

Syntax of Command:

*ooffaa*80*dd*)

where

oo	=	Upper or lower limit, depending on direction of change (limited by setting to be incremented, 0 to 23d for preamp 0 to 31d for fader 0 to 30d for eq tone 0 to 15d for eq frequency)
ff	=	Increment or decrement (**00** or **01**; 00 is decrement, 01 increment)
aa	=	Memory address (32 to 96d; **20** to **60** [pseudo-hex])
80	=	Device type bitmask for Advantage VRAM
dd	=	Device number bitmask (1 to 63d; **00** to **3?** [pseudo-hex])
)	=	increment-decrement-memory command character

Syntax of response:

no response

Example:

command:	response:
02003<8007)	***(none)***

This example sets the lower bound as 02, and then decreases memory location **3<** [pseudo-hex] by one. This location happens to be the main volume level. Repeated use of this command will force the fader to its lower limit of 2d steps from the bottom.

Comments:

The increment / decrement command is limited to preamp gains, volume, and equalizer faders only.

* polling-status

Description:

In order to give the user a glimpse into the current status of the Advantage VRAM and Advantage VRAMeq, the polling-status command can be used. When directed, this command will return information regarding the mix status of the device, it's auxiliary and main analog to digital levels, as well as the presence of clipping, activity of the logic output, auxiliary and main outputs.

Syntax of Command:

80*dd**

where

80	=	Device type bitmask
dd	=	Device number bitmask (1 to 63d; **00** to **3?** [pseudo-hex])
*	=	polling-status command character

Syntax of response:

ppyyzzcclluumm↵

where

pp	=	Last preset and communication bit used with BiampWin (first nibble is for Biamp use only, second nibble indicates current preset, **0** to **?** [pseudo-hex])
yy	=	Auxiliary analog to digital converter level (level ranges from **00** to **??** [pseudo-hex])
zz	=	Main analog to digital converter level (level ranges from **00** to **??** [pseudo-hex])
cc	=	Clipping presence (high bit indicates presence, **00** or **01**)
ll	=	Logic output status (binary high bit indicates active, **00** to **??** [pseudo-hex])
uu	=	Auxiliary output status (binary high output indicates channel active, **00** to **??** [pseudo-hex])
mm	=	Main output status (binary high output indicates channel active, **00** to **??** [pseudo-hex]

Example:

command:	response:
8002*	**292:2;00000303**

In this example, a Advantage VRAM (device number 2) reports that it's last preset was 9. The 2 appearing before the 9 is used by BiampWin for communication purposes. The auxiliary and main a/d converter levels are 42 and 43d, or about 16% of max. There is no clipping, nor any output to the logic out. Finally, both the auxiliary and main outputs are being fed by channels 1 and 2 (**03** translates to 00000011b, indicating that the first two channels are on).

Comments:

Preset:
If the last preset selected was 16d, (the startup preset) and a communication bit is set, there can be some confusion. Preset 16d, without a communication bit has a pseudo-hex value of **10**, or 00010000b. However, when a communication bit is set, say, bit 8, the resultant binary is 10010000, or **90** [pseudo-hex]. In all cases other than preset 16d the communication bits always remain in the most significant nibble and the preset remains in the least significant nibble.

Logic outputs:
As there are 8 logic outputs, each bit in the pseudo-hex value represents a specific output. The outputs are ordered from most significant bit to least significant bit. For instance, an output of **?;** [pseudo-hex] would coincide with an binary value of 11111011b. Going from msb to lsb, this indicates that pins 8, 7, 6, 4, 2, and 1 are active.

Main and auxiliary outputs:
These work in a similar manner to the logic outputs. Each of the 8 outputs are represented by a bit in the pseudo-hex value. Each channel is ordered from lowest to highest, lsb to msb.

+ sleep-for-10-seconds

Description:

The sleep-for-10-seconds command allows the Advantage VRAM and Advantage VRAMeq to fall "asleep" for 10 seconds, ignoring all communication. During this 10 seconds of sleep, the Advantage VRAM will not respond to nor echo any commands that it receives.

Syntax of Command:

80*dd*+

where

80	=	Device type bitmask for the Advantage VRAM
dd	=	Device number bitmask (1 to 63d; **00** to **3?** [pseudo-hex])
+	=	sleep-for-10-seconds command character

Syntax of response:

no response

Example:

command:	response:
800;+	***(none)***

This example causes the Advantage VRAM (device number 11d) to sleep for 10 seconds.

Comments:

, read-eeprom-locations

Description:

Specifying the read-eeprom-locations command causes the Advantage VRAM and Advantage VRAMeq models to read a specified number of bytes starting at any valid memory location in any memory bank. This information is then passed to the serial port, from the last byte of sequence to the first byte specified. The Advantage VRAM has 16 banks with 256 bytes each

Syntax of Command:

*bbaann*80*dd*,

where

bb	=	Bank select (0 to 15d; **00** to **0?** [pseudo-hex])
aa	=	Starting memory address (0 to 255d; **00** to **??** [pseudo-hex])
nn	=	Number of bytes to read minus one (limited by starting address)
80	=	Device type bitmask for Advantage VRAM
dd	=	Device number bitmask (1 to 63d*;* **00** to **3?** [pseudo-hex])
,	=	read-eeprom-locations command character

Syntax of response:

xx…(up to 256 data values)…↵

where

xx	=	Data value

Example:

command:	response:
0>74068001,	**010?0?0?0?0?0?**

This command causes the Advantage VRAM (device number 1) to go to bank 14d (**0>** [pseudo-hex]) and dump to the user the 7 bytes (since 7 - 1 is **06**) from byte 116d (**74** [pseudo-hex]) on. The output indicates that byte 122d contains **01**, while bytes 116 to 121d all contain **0?** [pseudo-hex].

Comments:

- write-eeprom-locations

Description:

The write-eeprom-locations command allows the user to write directly to the eeprom, placing specific characters in designated memory locations. The Advantage VRAM and Advantage VRAMeq each allow the user to program all of the eeprom's 16 banks of 256 bytes. While this provides a powerful method of setting or changing configuration parameter, it also provides an easy way to screw things up.

Syntax of Command:

*xx…(up to 16 data values)…bbaanncc*80*dd-*

where

xx	=	Up to 16 data values, sent in reverse order, highest memory address first.
bb	=	Bank select (0 to 15d; **00** to **0?** [pseudo-hex];)
aa	=	Starting memory address (0 to 255d; **00** to **??** [pseudo-hex];)
nn	=	Number of bytes to write minus one (limited by starting address)
cc	=	Checksum which consists of the 1's compliment of the eight bit sum of *nn* + *aa* + *bb* + *xx* + ...
80	=	Device type bitmask for Advantage VRAM
dd	=	Device number bitmask (1 to 63d; **00** to **3?** [pseudo-hex])
-	=	write-eeprom-locations command character

Syntax of response:

no response

Example:

command:	response:
0=07=>000>8006-	***(none)***

This example commands an Advantage VRAM (device number 6) to access bank 7d of the non-volatile memory. In this bank, it writes 1 byte (recall that 1 - 1 is **00**), **0=**, to memory location 222d (=> pseudo-hex). Finally, as a checksum, the command provides **0>** (00001110b), the one's compliment of the sum of **0=**, **07**, **=>**, and **00** [pseudo-hex]. If the command had specified more than one byte, then the Advantage VRAM would have entered the data from the highest memory location to the lowest.

. set-baud

Description:

The set-baud rate command allows the user to specify the baud rate at which the Advantage VRAM and Advantage VRAMeq operate. The units operate at 2400, 9600, 19200, and 38400 baud. In order to specify which of these baud rates to use, the Advantage VRAM refers tothem by the numbers 0,1,2 and 3; respectively.

Syntax of Command:

*rrii*80*dd*.

where

rr	=	Baud rate (**00** to **03**)
ii	=	Compliment of selected baud rate (**0<** to **0?** [pseudo-hex])
80	=	Device type bitmask for Advantage VRAM
dd	=	Device number bitmask (1 to 63d; **00** to **3?** [pseudo-hex])
.	=	set-baud command character

Syntax of response:

no response

Example:

command:	response:
00??8002.	***(none)***

This command changes the baud of the Advantage VRAM (device number 2) to 2400 (mode **00** [pseudo-hex]).

Comments:

Changing the baud value will immediately disconnect the user from the Advantage VRAM until the user has changed the baud of the device connected to serial port also. Therefore, this command can be dangerous and is not recommended.

/ get-version

Description:

The get-version command causes the Advantage VRAM and Advantage VRAMeq to return the model identification code and firmware version to the user. The firmware version is the release date, in the American format *mmddyy*. It is important to note that the Advantage VRAM will return this date in decimal format, **not** pseudo-hex.

Syntax of Command:

80*dd*/

where

80	=	Device type bitmask for Advantage VRAM
dd	=	Device number bitmask (1 to 63d; **00** to **3?** [pseudo-hex])
/	=	get-version command character

Syntax of response:

12*mmddyy*↵

where

12	=	Model i.d. for Advantage VRAM
mm	=	2 digit decimal month character
dd	=	2 digit decimal day character
yy	=	2 digit decimal year character

Example:

command:	response:
800=/	**12060598**

This command asks a Advantage VRAM, number 13d, (**0=** [pseudo-hex]) to return its model i.d. and firmware date. In this case, the model i.d. is **12** [pseudo-hex] and firmware date is 6/5/98.

Comments:

Using the ? character will act as a wild card for any of the parameters.

ASCII Code Chart

with Decimal & Hexadecimal Equivalents and Advantage DRI Commands

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

Address		Used for storage of	Byte Controls	Value Ranges	Corresponds to
0 0	0	Auto (gated) to main status	Ch.1 to 8, from lsb to msb. Each bit controls one channel	0 or 1	0 not auto (gated), 1 auto (gated)
0 1	1	Main out channel on or off	Ch.1 to 8, from lsb to msb. Each bit controls one channel	0 or 1, bit only active when coinciding bit in byte 2 is value 0	0 off, 1 on
0 2	2	Auto (gated) to aux status	Ch.1 to 8, from lsb to msb. Each bit controls one channel	0 or 1	0 not auto (gated), 1 auto (gated)
0 3	3	Aux out channel on or off	Ch.1 to 8, from lsb to msb. Each bit controls one channel	0 or 1, bit only active when coinciding bit in byte 2 is value 0	0 off, 1 on
0 4	4	Direct out follow gate status	Ch.1 to 8, from lsb to msb. Each bit controls one channel	0 or 1	0 not follow gated, 1 follow gated
0 5	5	Direct out on or off	Ch.1 to 8, from lsb to msb. Each bit controls one channel	0 or 1, bit only active when coinciding bit in byte 4 is value 0	0 off, 1 on
0 6	6	Logic output follow gate status	Ch.1 to 8, from lsb to msb. Each bit controls one channel	0 or 1	0 not follow gated, 1 follow gated
0 7	7	Logic output on or off	Ch.1 to 8, from lsb to msb. Each bit controls one channel	0 or 1, bit only active when coinciding bit in byte 6 is value 0	0 off, 1 on
0 8	8	Config bits	See note 1	See note 1	See note 1
0 9	9	MaxNOM presence	Maximum number of open mic	0 to 7	number of open mics
0 :	10	Main and aux attenuation	attenuation	0 to 15	see table
0 ;	11	Channel on time	reload the channel on	0 to 255	value*.025 seconds
0 <	12	Designated mic	designated mic	See note 2	See note 2
0 =	13	reserved			
0 >	14	reserved			
0 ?	15	reserved			
1 0	16	Turn on delay	Ch.1 delay	0 to 255	.025*value seconds
1 1	17	Turn on delay	Ch.2 delay	0 to 255	.025*value seconds
1 2	18	Turn on delay	Ch.3 delay	0 to 255	.025*value seconds
1 3	19	Turn on delay	Ch.4 delay	0 to 255	.025*value seconds
1 4	20	Turn on delay	Ch.5 delay	0 to 255	.025*value seconds
1 5	21	Turn on delay	Ch.6 delay	0 to 255	.025*value seconds
1 6	22	Turn on delay	Ch.7 delay	0 to 255	.025*value seconds
1 7	23	Turn on delay	Ch.8 delay	0 to 255	.025*value seconds
1 8	24	Logic output turnoff delay	Ch.1 delay	0 to 255	.025*value seconds
1 9	25	Logic output turnoff delay	Ch.2 delay	0 to 255	.025*value seconds
1 :	26	Logic output turnoff delay	Ch.3 delay	0 to 255	.025*value seconds
1 ;	27	Logic output turnoff delay	Ch.4 delay	0 to 255	.025*value seconds
1 <	28	Logic output turnoff delay	Ch.5 delay	0 to 255	.025*value seconds
1 =	29	Logic output turnoff delay	Ch.6 delay	0 to 255	.025*value seconds
1 >	30	Logic output turnoff delay	Ch.7 delay	0 to 255	.025*value seconds
1 ?	31	Logic output turnoff delay	Ch.8 delay	0 to 255	.025*value seconds
2 0	32	gain, phantom, hpf	Ch.1	See note 3	See note 3
2 1	33	gain, phantom, hpf	Ch.2	See note 3	See note 3

2 2	34	gain, phantom, hpf	Ch.3	See note 3	See note 3
2 3	35	gain, phantom, hpf	Ch.4	See note 3	See note 3
2 4	36	gain, phantom, hpf	Ch.5	See note 3	See note 3
2 5	37	gain, phantom, hpf	Ch.6	See note 3	See note 3
2 6	38	gain, phantom, hpf	Ch.7	See note 3	See note 3
2 7	39	gain, phantom, hpf	Ch.8	See note 3	See note 3
2 8	40	Main Feed	Ch. 1 level	See note 4	See note 4
2 9	41	Aux Feed	Ch. 1 level	See note 4	See note 4
2 :	42	Main Feed	Ch. 2 level	See note 4	See note 4
2 ;	43	Aux Feed	Ch. 2 level	See note 4	See note 4
2 <	44	Main Feed	Ch. 3 level	See note 4	See note 4
2 =	45	Aux Feed	Ch. 3 level	See note 4	See note 4
2 >	46	Main Feed	Ch. 4 level	See note 4	See note 4
2 ?	47	Aux Feed	Ch. 4 level	See note 4	See note 4
3 0	48	Main Feed	Ch. 5 level	See note 4	See note 4
3 1	49	Aux Feed	Ch. 5 level	See note 4	See note 4
3 2	50	Main Feed	Ch. 6 level	See note 4	See note 4
3 3	51	Aux Feed	Ch. 6 level	See note 4	See note 4
3 4	52	Main Feed	Ch. 7 level	See note 4	See note 4
3 5	53	Aux Feed	Ch. 7 level	See note 4	See note 4
3 6	54	Main Feed	Ch. 8 level	See note 4	See note 4
3 7	55	Aux Feed	Ch. 8 level	See note 4	See note 4
3 8	56	Main Feed	Aux 1 level	See note 4	See note 4
3 9	57	Aux Feed	Aux 1 level	See note 4	See note 4
3 :	58	Main Feed	Aux 2 level	See note 4	See note 4
3 ;	59	Aux Feed	Aux 2 level	See note 4	See note 4
3 <	60	Main Output	Main out level	See note 4	See note 4
3 =	61	Aux Output	Aux out level	See note 4	See note 4
3 >	62	Last recalled preset	Preset number	0 to 16	Preset number
3 ?	63	reserved			
4 0	64	Bass	Ch. 1	0 to 30	-9 dB to 9dB with 0 dB at 15
4 1	65	Mid	Ch. 1	0 to 30	-9 dB to 9dB with 0 dB at 15
4 2	66	Mid Freq	Ch. 1	0 to 15	(value+1)*220 Hz
4 3	67	High	Ch. 1	0 to 30	-9 dB to 9dB with 0 dB at 15
4 4	68	Bass	Ch. 2	0 to 30	-9 dB to 9dB with 0 dB at 15
4 5	69	Mid	Ch. 2	0 to 30	-9 dB to 9dB with 0 dB at 15
4 6	70	Mid Freq	Ch. 2	0 to 15	(value+1)*220 Hz
4 7	71	High	Ch. 2	0 to 30	-9 dB to 9dB with 0 dB at 15
4 8	72	Bass	Ch. 3	0 to 30	-9 dB to 9dB with 0 dB at 15
4 9	73	Mid	Ch. 3	0 to 30	-9 dB to 9dB with 0 dB at 15
4 :	74	Mid Freq	Ch. 3	0 to 15	(value+1)*220 Hz

4 ;	75	High	Ch. 3	0 to 30	-9 dB to 9dB with 0 dB at 15
4 <	76	Bass	Ch. 4	0 to 30	-9 dB to 9dB with 0 dB at 15
4 =	77	Mid	Ch. 4	0 to 30	-9 dB to 9dB with 0 dB at 15
4 >	78	Mid Freq	Ch. 4	0 to 15	(value+1)*220 Hz
4 ?	79	High	Ch. 4	0 to 30	-9 dB to 9dB with 0 dB at 15
5 0	80	Bass	Ch. 5	0 to 30	-9 dB to 9dB with 0 dB at 15
5 1	81	Mid	Ch. 5	0 to 30	-9 dB to 9dB with 0 dB at 15
5 2	82	Mid Freq	Ch. 5	0 to 15	(value+1)*220 Hz
5 3	83	High	Ch. 5	0 to 30	-9 dB to 9dB with 0 dB at 15
5 4	84	Bass	Ch. 6	0 to 30	-9 dB to 9dB with 0 dB at 15
5 5	85	Mid	Ch. 6	0 to 30	-9 dB to 9dB with 0 dB at 15
5 6	86	Mid Freq	Ch. 6	0 to 15	(value+1)*220 Hz
5 7	87	High	Ch. 6	0 to 30	-9 dB to 9dB with 0 dB at 15
5 8	88	Bass	Ch. 7	0 to 30	-9 dB to 9dB with 0 dB at 15
5 9	89	Mid	Ch. 7	0 to 30	-9 dB to 9dB with 0 dB at 15
5 :	90	Mid Freq	Ch. 7	0 to 15	(value+1)*220 Hz
5 ;	91	High	Ch. 7	0 to 30	-9 dB to 9dB with 0 dB at 15
5 <	92	Bass	Ch. 8	0 to 30	-9 dB to 9dB with 0 dB at 15
5 =	93	Mid	Ch. 8	0 to 30	-9 dB to 9dB with 0 dB at 15
5 >	94	Mid Freq	Ch. 8	0 to 15	(value+1)*220 Hz
5 ?	95	High	Ch. 8	0 to 30	-9 dB to 9dB with 0 dB at 15

Attenuation Table

value	dB
0	-80
1	-40
2	-35
3	-30
4	-25
5	-20
6	-19
7	-18
8	-17
9	-16
10	-15
11	-14
12	-13
13	-12
14	-11
15	-10

Note 1 [Config bits]) High bit indicates the presence of the following (from lsb to msb)

bit 1- last mic hold
bit 2- force into manual
bit 3- teleconference mode
bit 4- aux 1 +/-6dB gain
bit 5- aux 2 +/-6 dB gain
bit 6- disable NOM attenuation
bit 7- logic out don't track last mic channel
bit 8- direct out don't track last mic channel.

Example: 01101010 has

bit 2- manual mode engaged
bit 4- aux 1 at +6dB gain
bit 6- NOM attenuation disabled
bit 7- the logic out is not tracking the last mic channel

Note 2 [Mic]: The bit that is high represents the designated mic on. If no bit is high, the system is either on last mic hold or has no designated mic.

Example: 01000000 has

7- mic 7 as designated mic

Note 3 [Gain]: Multi-purpose byte. 5 least significant bits represent gain and range in value from 0 to 23.

Gain = (value-2)*3dB.

Add binary 100000 for phantom power, binary 1000000 for high pass filter

Example: 01100111 has

bits (1, 2, 3)- a gain of (7-2)*3=15dB
bit 6- phantom power on
bit 7- high pass filter on

Note 4 [Feeds]: Multi-purpose byte. 5 least significant bits represent feed levels and values range from 0 to 31. 0 dB occurs at 25 value. Setting the msb, 128, (adding 10000000 binary) to high indicates muting. Another way of thinking of this is that the un-muted levels range from 0 to 31, while the muted levels range from 128 to 159.

Example: 10000011 has

bits (1, 2)- volume level of 3 (out of 31)
bit 8- muting on