DR Command Reference v03Mar2015

Table of Contents

DR Mainframe Commands

baud	Serial port baud rate
block	Mainframe frequency block
desc	Mainframe description string
hwstat	Hardware status
id	Mainframe id string
lcdbl	LCD backlight brightness
localmclk	Local AES master clock
mclkterm	AES clock input termination
monmode	Headphone monitor mode
phantpwr	Antenna phantom power
portctl	Communication port control flags
samplerate	AES sample rate
serial	Mainframe serial number
temp	Mainframe internal temperature
version	Mainframe firmware version

Termination: all commands are terminated with an ASCII **carriage return** character (hex code 0x0D), represented by **<CR>** in the examples below. All responses are terminated with an ASCII **carriage return, line feed** pair (hex codes 0x0D, 0x0A), represented by **<CRLF>** in the examples below. An ellipsis (**...**) represents members of an array that have been omitted from an example for the sake of brevity.

Verbose response: commands prefixed with an exclamation point (bang) character result in a "verbose" response containing both the name of the property being addressed and its current value (if any). The verbose response returns the property/value pair in the "assignment" form, for example `OK ingn(2)=40 <CRLF>`. This supports certain 3rd party control programming styles where the response needs to be self-describing.

baud (Serial port baud rate)

This command may be used as a query to determine the baud rate setting for the serial port, or as an update to set it. The data is an integer type. The following values are allowed:

- 9600

- 19200
- 38400
- 57600
- 115200

Examples:

	REQUEST	RESPONSE
QUERY	**`baud?<CR>`**	**`OK 57600<CRLF>`**
UPDATE	**`baud=57600<CR>`**	**`OK<CRLF>`**

block (Mainframe frequency block)

This command may be used as a query to read the mainframe frequency block. The data type is integer and is a code that specifies the frequency block for the mainframe. It may be in the range 0 to 2, with the following meanings:

Code	**Frequency Block**
0	Low
1	Mid
2	High

Examples:

	REQUEST	RESPONSE
QUERY	**`rxblock?<CR>`**	**`OK 0<CRLF>`**

desc (Mainframe description string)

This command may be used as a query to read the user defined mainframe description, or as an update to set it. The data is a string type, with a limit of 30 characters.

Note: String arguments in commands need to be passed in **quoted** form, contained in a pair of **double-quote** (") characters. A problem arises when using the `desc` command to read or write a string that already contains double-quote characters, for example: `The "Hula" Room`. The solution is to **escape** the double quotes within `The "Hula" Room` so that it can be passed as a string argument for the `desc` command. This is done by preceding the double-quote characters with a **backslash** character like this: `The \"Hula\" Room`. Now it can be passed as a string argument to the `desc` command: `desc="The \"Hula\" Room"`. Since the **backslash** serves as the escape character in quoted-string arguments, it too must be escaped if it is part of the string, so `"foo\bar"` would become `"foo\\bar"`. If necessary, any character, printable or non-printable, can be represented in the hexadecimal escaped form `\xHH` where `HH` is any 2-digit hexadecimal number. The special escaped character forms `\r` (carriage return), `\n` (new line) and `\t` (tab) are also recognized.

Examples:

	REQUEST	RESPONSE
QUERY	**`desc?<CR>`**	**`OK "Aloha Room East"<CRLF>`**
UPDATE	**`desc="Studio #2"<CR>`**	**`OK<CRLF>`**

hwstat (Hardware status)

This command may be used as a query to determine the status of device hardware. The data type is integer, and the value is a code representing one of the following states:

- **0** means normal operation
- **1** means that an antenna phantom power short has been detected
- **2** means that the number of transmitter key instances available is low
- **3** means that no more transmitter key instances are available
- **4** means that a corrupted DRM module has been detected

Example:

	REQUEST	RESPONSE
QUERY	**`hwstat?<CR>`**	**`OK 0<CRLF>`**

id (Mainframe ID string)

This command may be used as a query to read the mainframe id string. This is the "name" of the device used by the control protocol and is always "DR". The data is a string type.

Example:

	REQUEST	RESPONSE

QUERY	`id?<CR>`	`OK "DR"<CRLF>`

lcdbl (LCD backlight brightness)

This command may be used as a query to read the LCD backlight brightness level, or as an update to set it. The data type is integer, in the range 1 to 4, where 1 is the minimum brightness and 4 the maximum brightness.

Examples:

	REQUEST	RESPONSE
QUERY	`lcdbl(1)?<CR>`	`OK 4<CRLF>`
UPDATE	`lcdbl(5)=3<CR>`	`OK<CRLF>`

localmclk (Local AES master clock)

This command may be used as a query to read the local AES master clock status, or as an update to set it. The data type is integer, either "1" meaning that local AES master clock is enabled, or "0" meaning that it is not.

Examples:

	REQUEST	RESPONSE
QUERY	`localmclk?<CR>`	`OK 0<CRLF>`
UPDATE	`localmclk=1<CR>`	`OK<CRLF>`

mclkterm (AES clock input termination)

This command may be used as a query to read the AES word clock input termination status, or as an update to set it. The data type is integer, either "1" meaning that the 75 ohm word clock termination is enabled, or "0" meaning that it is not.

Examples:

	REQUEST	RESPONSE
QUERY	`mclkterm?<CR>`	`OK 0<CRLF>`

UPDATE	**`mclkterm=1<CR>`**	**`OK<CRLF>`**

monmode (Headphone monitor mix mode)

This command may be used as a query to read the headphone monitor mix mode, or as an update to set it. The data type is integer The data type is integer, either "1" meaning that the monitor signal contains a mix of the output of all receivers, or "0" meaning that it contains only the output of a single selected receiver.

Examples:

	REQUEST	RESPONSE
QUERY	**`monmode?<CR>`**	**`OK 1<CRLF>`**
UPDATE	**`monmode=0<CR>`**	**`OK<CRLF>`**

phantpwr (Antenna phantom power)

This command may be used as a query to read the antenna phantom power status, or as an update to set it. The data type is integer and is a code that specifies the phantom power configuration. It may be in the range 0 to 3, with the following meanings:

Code	**Phantom Power Configuration**
0	No phantom power
1	Phantom power on antenna A only
2	Phantom power on antenna B only
3	Phantom power on both antenna A and B

Examples:

	REQUEST	RESPONSE
QUERY	**`phantpwr?<CR>`**	**`OK 0<CRLF>`**
UPDATE	**`phantpwr=3<CR>`**	**`OK<CRLF>`**

portctl (communication port control flags)

This command may be used as a query to read the port control flags, or as an update to set them. The communication port is specified by using the address syntax. Addresses must be in the range 0 to 4, representing the one of the following:

- **0** - USB port
- **1** - RS232 port
- **2** - TCP port 1
- **3** - TCP port 2
- **4** - HTTP port

The data type is integer, in the range 0 to 3. The value is a code representing the control settings for the communication port:

Code	Port Setting
0	Port is disabled
1	Port is receive only
2	Port is send only
3	Port is send/receive

If the port address is wildcarded, then the data type is an array of integer of size 4. By default, all communication ports are send/receive (full duplex) but in some advanced 3rd party control scenarios a port may need to be set otherwise. **Note: to preserve the ability to communicate with the device, changes to the USB port setting have no effect, the default of send/receive is always in force.**

Examples:

	REQUEST	RESPONSE
QUERY	`portctl(1)?<CR>`	`OK 3<CRLF>`
QUERY	`portctl(*)?<CR>`	`OK {3,3,3,3}<CRLF>`
UPDATE	`portctl(1)=2<CR>`	`OK<CRLF>`
UPDATE	`portctl(*)={3,3,3,3}<CR>`	`OK<CRLF>`

samplerate (AES sample rate)

This command may be used as a query to read the AES sample rate, or as an update to set it. The data type is integer, in samples per second. Two sample rates are allowed:

- **48000**
- **96000**

Examples:

	REQUEST	RESPONSE
QUERY	**`samplerate?<CR>`**	**`OK 48000<CRLF>`**
UPDATE	**`samplerate=96000<CR>`**	**`OK<CRLF>`**

serial (Mainframe serial number)

This command may be used as a query to read the mainframe's serial number. The data is a string type.

Example:

	REQUEST	RESPONSE
QUERY	**`serial?<CR>`**	**`OK "6100101"<CRLF>`**

temp (Mainframe internal temperature)

This command may be used as a query to read the internal temperature of the mainframe. The data type is integer, in degrees C.

Example:

	REQUEST	RESPONSE
QUERY	**`temp?<CR>`**	**`OK 32<CRLF>`**

version (Mainframe firmware version)

This command may be used as a query to read the mainframe's firmware version number. The data is a string type.

Example:

	REQUEST	RESPONSE
QUERY	**`version?<CR>`**	**`OK "1.0.1"<CRLF>`**

DR Channel Receiver Commands

rxalevel	Receiver audio attenuator level
rxameter	Receiver audio level meter
rxaudtype	Receiver audio output type
rxblock	Receiver frequency block
rxfreq	Receiver frequency
rxid	Receiver ID
rxlink	Receiver link status
rxmonmute	Receiver monitor mute status
rxmute	Receiver mute status
rxmutetog	Receiver mute toggle
rxname	Receiver name
rxphase	Receiver audio phase
rxpresent	Receiver present status
rxpwr	Receiver power on/off
rxrmeter	Receiver RF level meter
rxscan	Receiver scan state
rxsmartnr	Receiver smart noise reduction
rxsquelch	Receiver squelch status
rxstat	Receiver status
rxtone	Receiver output test tone
rxver	Receiver firmware version

Termination: all commands are terminated with an ASCII **carriage return** character (hex code 0x0D), represented by **`<CR>`** in the examples below. All responses are terminated with an ASCII **carriage return, line feed** pair (hex codes 0x0D, 0x0A), represented by **`<CRLF>`** in the examples below. An ellipsis (**...**) represents members of an array that have been omitted from an example for the sake of brevity.

Verbose response: commands prefixed with an exclamation point (bang) character result in a "verbose" response containing both the name of the property being addressed and its current value (if any). The verbose response returns the property/value pair in the "assignment" form, for example **`OK rxphase(2)=0 <CRLF>`**. This supports certain 3rd party control programming styles where the response needs to be self-describing.

rxalevel (Receiver audio attenuator level)

This command may be used as a query to read the receiver output level setting, or as an update to set it. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, representing the attenuation (or gain) in dB. The range is -35 to +8 dB. If the channel address is wildcarded, then the data type is an array of integer of size 6. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxalevel(1)?<CR>`	`OK rxalevel(1)=-10<CRLF>`
QUERY	`!rxalevel(*)?<CR>`	`OK rxalevel(*)={0,0,0,0,0,0}<CRLF>`
UPDATE	`!rxalevel(5)=0<CR>`	`OK rxalevel(5)=0<CRLF>`
UPDATE	`!rxalevel(*)={0,0,-3, 0,99,99}<CR>`	`OK rxalevel(*)={0,0,-3,0,0,0}<CRLF>`

rxameter (Receiver audio level meter)

This command may be used as a query to read the receiver audio level meter. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, in the range -80 to +5, representing the audio level in dB, referenced to the transmitter's threshold of limiting. This is the raw level *before* application of the receiver's audio level setting (rxalevel), which controls the rear panel audio output level (if rxameter = 0dB and rxalevel = 0dB then the output level is 0dBu). If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxameter(1)?<CR>`	`OK rxameter(1)=-23<CRLF>`
QUERY	`!rxameter(*)?<CR>`	`OK rxameter(*)={-2,4,-10,-53,-71,-95}<CRLF>`

rxaudtype (Receiver audio output type)

This command may be used as a query to read the receiver audio type. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that the output audio type is AES/EBU, or "0" meaning that it is analog. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	**`!rxaudtype(3)?<CR>`**	**`OK rxaudtype(3)=0<CRLF>`**
QUERY	**`!rxaudtype(*)?<CR>`**	**`OK rxaudtype(*)={0,0,0,0,1,1}<CRLF>`**

rxblock (Receiver frequency block)

This command may be used as a query to read the receiver frequency block. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, representing the block number. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	**`!rxblock(1)?<CR>`**	**`OK rxblock(1)=-10<CRLF>`**
QUERY	**`!rxblock(*)?<CR>`**	**`OK rxblock(*)={20,21,21,22,0,0}<CRLF>`**

rxfreq (Receiver frequency)

This command may be used as a query to read the receiver frequency, or as an update to set it. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, representing the frequency in kHz. If the channel address is wildcarded, then the data type is an array of integer of size 6. In this case the value **999999** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	**`!rxfreq(1)?<CR>`**	**`OK rxfreq(1)=471200<CRLF>`**
QUERY	**`!rxfreq(*)?<CR>`**	**`OK rxfreq(*)={471200,520900,...,540000}<CRLF>`**

UPDATE	**!rxfreq(1)=471200<CR>**	**OK rxfreq(1)=471.200<CRLF>**
UPDATE	**!rxfreq(*)={471200,520900,...,999999}<CR>**	**OK rxfreq(*)={471200,520900,...,540000}<CRLF>**

rxid (Receiver ID)

This command may be used as a query to read the receiver's ID string. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data is a string type.

Example:

	REQUEST	RESPONSE
QUERY	**!rxid(1)?<CR>**	**OK rxid(1)="DRM"<CRLF>**

rxlink (Receiver link status)

This command may be used as a query to read the link status. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that the data link is established, or "0" meaning that it is not. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	**!rxlink(3)?<CR>**	**OK rxlink(3)=0<CRLF>**
QUERY	**!rxlink(*)?<CR>**	**OK rxlink(*)={0,1,0,0,0,0}<CRLF>**

rxmonmute (Receiver monitor mute status)

This command may be used as a query to read the receiver monitor mute status, or as an update to set it. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that the output is muted, or "0" meaning that it is not. If the channel address is wildcarded, then the data type is an array of integer of size 6. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxmonmute(3)?<CR>`	`OK rxmonmute(3)=1<CRLF>`
QUERY	`!rxmonmute(*)?<CR>`	`OK rxmonmute(*)={0,0,0,0,1,0}<CRLF>`
UPDATE	`!rxmonmute(2)=0<CR>`	`OK rxmonmute(2)=0<CRLF>`
UPDATE	`!rxmonmute(*)={0,0,0,99,99,99}<CR>`	`OK rxmonmute(*)={0,0,0,0,0,0}<CRLF>`

rxmute (Receiver mute status)

This command may be used as a query to read the receiver mute status, or as an update to set it. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that the output is muted, or "0" meaning that it is not. If the channel address is wildcarded, then the data type is an array of integer of size 6. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxmute(3)?<CR>`	`OK rxmute(3)=1<CRLF>`
QUERY	`!rxmute(*)?<CR>`	`OK rxmute(*)={0,0,0,0,1,0}<CRLF>`
UPDATE	`!rxmute(2)=0<CR>`	`OK rxmute(2)=0<CRLF>`
UPDATE	`!rxmute(*)={0,1,0,99,99,99}<CR>`	`OK rxmute(*)={0,1,0,0,0,0}<CRLF>`

rxmutetog (Receiver mute toggle)

This command may be used as a simple comand to toggle the receiver mute status. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The response to a verbose command is the new mute status for that receiver.

Examples:

	REQUEST	RESPONSE
COMMAND	`!rxmutetog(4)<CR>`	`OK rxmute(4)=1<CRLF>`

rxname (Receiver name)

This command may be used as a query to read the receiver name, or as an update to set it. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is string, with a limit of 15 characters.

Note: String arguments in commands need to be passed in **quoted** form, contained in a pair of **double-quote** (") characters. A problem arises when using the `inlb` command to read or write a string that already contains double-quote characters, for example: `The "Hula" Room`. The solution is to **escape** the double quotes within `The "Hula" Room` so that it can be passed as a string argument for the `inlb` command. This is done by preceding the double-quote characters with a **backslash** character like this: `The \"Hula\" Room`. Now it can be passed as a string argument to the `inlb` command: `inlb(1)="The \"Hula\" Room"`. Since the **backslash** serves as the escape character in quoted-string arguments, it too must be escaped if it is part of the string, so `"foo\bar"` would become `"foo\\bar".` If necessary, any character, printable or non-printable, can be represented in the hexadecimal escaped form `\xHH` where `HH` is any 2-digit hexadecimal number. The special escaped character forms `\r` (carriage return), `\n` (new line) and `\t` (tab) are also recognized..

Examples:

	REQUEST	RESPONSE
QUERY	**!rxname(1)?<CR>**	**OK rxname(1)="Chairman"<CRLF>**
UPDATE	**!rxname(2)="David"<CR>**	**OK rxname(2)="David"<CRLF>**

rxphase (Receiver audio phase)

This command may be used as a query to read the receiver audio phase status, or as an update to set it. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that the audio phase inverted (shifted by 180 degrees), or "0" meaning that it is not. If the channel address is wildcarded, then the data type is an array of integer of size 6. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	**!rxphase(3)?<CR>**	**OK rxphase(3)=0<CRLF>**
QUERY	**!rxphase(*)?<CR>**	**OK rxphase(*)={0,1,0,0,0,0}<CRLF>**

UPDATE	`!rxphase(2)=1<CR>`	`OK rxphase(2)=1<CRLF>`
UPDATE	`!rxphase(*)={0,0,1,99,99,99}<CR>`	`OK rxphase(*)={0,0,1,1,1,1}<CRLF>`

rxpresent (Receiver present status)

This command may be used as a query to read the receiver present status. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that the receiver is installed in the mainframe, or "0" meaning that it is not. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxpresent(3)?<CR>`	`OK rxpresent(3)=1<CRLF>`
QUERY	`!rxpresent(*)?<CR>`	`OK rxpresent(*)={1,1,1,1,0,0}<CRLF>`

rxpwr (Receiver power on/off)

This command may be used as a query to read the receiver power on/off status, or as an update to set it. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that the receiver is powered on, or "0" meaning that it is powered off. If the channel address is wildcarded, then the data type is an array of integer of size 6. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxpwr(3)?<CR>`	`OK rxpwr(3)=0<CRLF>`
QUERY	`!rxpwr(*)?<CR>`	`OK rxpwr(*)={1,1,1,0,0,0}<CRLF>`
UPDATE	`!rxpwr(2)=1<CR>`	`OK rxpwr(2)=1<CRLF>`
UPDATE	`!rxpwr(*)={1,1,1,0,99,99}<CR>`	`OK rxpwr(*)={1,1,1,0,0,0}<CRLF>`

rxrmeter (Receiver RF level meter)

This command may be used as a query to read the receiver RF level meter. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, in the range 0 to 50, representing the RF level in dBuV. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxrmeter(1)?<CR>`	`OK rxrmeter(1)=45<CRLF>`
QUERY	`!rxrmeter(*)?<CR>`	`OK rxrmeter(*)={0,12,45,0,0,0}<CRLF>`

rxscan (Receiver scan state)

This command may be used as a query to read the receiver scan state, or as an update to set it. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that the receiver is scanning, or "0" meaning that it is not. If the channel address is wildcarded, then the data type is an array of integer of size 6. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command.

Examples:

	REQUEST	RESPONSE
QUERY	`!rxscan(3)?<CR>`	`OK rxscan(3)=0<CRLF>`
QUERY	`!rxscan(*)?<CR>`	`OK rxscan(*)={1,0,0,0,0,0}<CRLF>`
UPDATE	`!rxscan(2)=1<CR>`	`OK rxscan(2)=1<CRLF>`
UPDATE	`!rxscan(*)={1,0,0,99,99,99}<CR>`	`OK rxscan(*)={1,0,0,0,0,0}<CRLF>`

rxsmartnr (Receiver smart noise reduction)

This command may be used as a query to read the smart noise reduction status, or as an update to set it. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. If the channel address is wildcarded, then the data type is an array of integer of size 6. In this case the value **99** may be used in an update to indicate that a particular receiver is to **remain unchanged** by the command. The data type is integer, and is a code representing one of the following:

Code	Noise Reduction Level

0	None
1	Normal
2	Full

Examples:

	REQUEST	RESPONSE
QUERY	**`!rxsmartnr(3)?<CR>`**	**`OK rxsmartnr(3)=1<CRLF>`**
QUERY	**`!rxsmartnr(*)?<CR>`**	**`OK rxsmartnr(*)={1,1,1,1,1,0}<CRLF>`**
UPDATE	**`!rxsmartnr(2)=0<CR>`**	**`OK rxsmartnr(2)=0<CRLF>`**
UPDATE	**`!rxsmartnr(*)={0,1,0,99,99,99}<CR>`**	**`OK rxsmartnr(*)={1,1,1,1,1,1}<CRLF>`**

rxsquelch (Receiver squelch status)

This command may be used to determine if a receiver is squelched. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that the receiver is squelched (no audio output), or "0" meaning that it is not. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	**`!rxsquelch(1)?<CR>`**	**`OK rxsquelch(1)=1<CRLF>`**
QUERY	**`!rxsquelch(*)?<CR>`**	**`OK rxsquelch(*)={0,1,1,0,0,0}<CRLF>`**

rxstat (Receiver status)

This command may be used as a query to read *real-time* receiver status. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is array of integer, with a length of 9. The values contained in the array are interpreted by position as follows:

Position	**Value**
1	Receiver present status (0 = not present, 1 = present)
2	Power status (0 = powered down, 1 = powered up)

3	Link Status (0 = no link, 1 = link established)
4	Audio meter (in range -80 to +5 dB, referenced to the transmitters's threshold of limiting, *before* application of rxalevel setting)
5	RF meter (in range 0 to 50 dBuV)
6	Scan status (0 = normal operation, 1 = receiver scanning)
7	Antenna diversity status (1 = antenna A selected, 2 = antenna B selected)
8	Mute status (0 = unmuted, 1 = muted)
9	Squelch status (0 = unsquelched, 1 = squelched)

If a receiver is not present in the mainframe or is powered down, then the remaining data is invalid.

Example:

	REQUEST	RESPONSE
QUERY	**`!rxstat(1)?<CR>`**	**`OK rxstat(1)={1,1,1,-12,22,0,1,0,0}<CRLF>`**

rxtone (Receiver output test tone)

This command may be used as an update to set the receiver test tone status. If the channel address is wildcarded, then the data type is an array of integer of size 6. The data type is integer and is a code that specifies the tone level and frequency. It may be in the range 0 to 2, with the following meanings:

Code	Test Tone
0	Off
1	1 kHz tone at nominal 0dB level
2	1 kHz tone at full scale level

Examples:

	REQUEST	RESPONSE
UPDATE	**`!rxtone(2)=1<CR>`**	**`OK rxtone(2)=1<CRLF>`**
UPDATE	**`!rxtone(*)={1,0,0,99,99,99}<CR>`**	**`OK rxtone(*)={1,0,0,0,0,0}<CRLF>`**

rxver (Receiver firmware version)

This command may be used as a query to read the receiver's firmware version number. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data is a string type.

Example:

	REQUEST	RESPONSE
QUERY	**`!rxver(1)?<CR>`**	**`OK rxver(1)="1.0"<CRLF>`**

DR Channel Transmitter Commands

txautoon	Transmitter auto-on
txbatt	Transmitter battery selection
txbl	Transmitter backlight control
txblevel	Transmitter battery level
txbtime	Transmitter battery timer
txbutton	Transmitter button mode
txbwarn	Transmitter battery warning
txgain	Transmitter audio gain
txmute	Transmitter mute status
txname	Transmitter name
txphase	Transmitter audio phase
txrolloff	Transmitter low frequency roll off
txstat	Transmitter status
txver	Transmitter firmware version

Termination: all commands are terminated with an ASCII **carriage return** character (hex code 0x0D), represented by **`<CR>`** in the examples below. All responses are terminated with an ASCII **carriage return, line feed** pair (hex codes 0x0D, 0x0A), represented by **`<CRLF>`** in the examples below. An ellipsis (**...**) represents members of an array that have been omitted from an example for the sake of brevity.

Verbose response: commands prefixed with an exclamation point (bang) character result in a "verbose" response containing both the name of the property being addressed and its current value (if any). The verbose response returns the property/value pair in the "assignment" form, for example **`OK rxphase(2)=0 <CRLF>`**. This supports certain 3rd party control programming styles where the response needs to be self-describing.

txautoon (Transmitter auto-on)

This command may be used as a query to read the transmitter auto-on status. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that auto-on is enabled, or "0" meaning that it is not. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	`!txautoon(3)?<CR>`	`OK txautoon(3)=1<CRLF>`
QUERY	`!txautoon(*)?<CR>`	`OK txautoon(*)={0,0,0,0,1,0}<CRLF>`

txbatt (Transmitter battery selection)

This command may be used as a query to read the transmitter battery selection. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. If the channel address is wildcarded, then the data type is an array of integer of size 6. The data type is integer and is a code that specifies the battery type. It may be in the range 0 to 1, with the following meanings:

Code	**Battery Type**
0	Alkaline battery (battery level displayed)
1	Lithium battery (battery level displayed)

Examples:

	REQUEST	RESPONSE
QUERY	`!txbatt(3)?<CR>`	`OK txbatt(3)=0<CRLF>`
QUERY	`!txbatt(*)?<CR>`	`OK txbatt(*)={1,0,0,0,0,0}<CRLF>`

txbl (Transmitter backlight timeout)

This command may be used as a query to read the transmitter backlight control setting. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. If the channel address is wildcarded, then the data type is an array of integer of size 6. The data type is integer and is a code that specifies the backlight timeout. It may be in the range 0 to 2, with the following meanings:

Code	**Backlight timeout**
0	No timeout
1	Times out in 30 seconds
2	Times out in 5 minutes

Examples:

	REQUEST	RESPONSE
QUERY	`!txbl(3)?<CR>`	`OK txbl(3)=1<CRLF>`
QUERY	`!txbl(*)?<CR>`	`OK txbl(*)={1,1,0,0,1,0}<CRLF>`

txblevel (Transmitter battery level)

This command may be used as a query to read the transmitter battery level. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, in the range 0 to 31, representing the battery voltage in tenth-volt increments. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	`!txbattlevel(1)?<CR>`	`OK txbattlevel(1)=29<CRLF>`
QUERY	`!txbattlevel(*)?<CR>`	`OK txbattlevel(*)={29,27,30,0,0,0}<CRLF>`

txbtime (Transmitter battery time)

This command may be used as a query to read the transmitter battery elapsed time. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, representing the elapsed time in minutes, in the range 0 to 599. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	`!txbtime(1)?<CR>`	`OK txbtime(1)=-10<CRLF>`
QUERY	`!txbtime(*)?<CR>`	`OK txbtime(*)={20,31,210,234,0,0}<CRLF>`

txbutton (Transmitter button function)

This command may be used as a query to read the transmitter button function. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. If the channel address is wildcarded, then the data type is an array of integer of size 6. The data type is integer and is a code that specifies the button function. It may be in the range 0 to 2, with the following meanings:

Code	Button Function
0	No function
1	Mute on/off
2	Power on/off

Examples:

	REQUEST	RESPONSE
QUERY	`!txbutton(3)?<CR>`	`OK txbutton(3)=1<CRLF>`
QUERY	`!txbutton(*)?<CR>`	`OK txbutton(*)={1,1,0,0,1,0}<CRLF>`

txbwarn (Transmitter battery warning)

This command may be used as a query to read the transmitter battery warning level. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. If the channel address is wildcarded, then the data type is an array of integer of size 6. The data type is integer and is a code that specifies the battery warning level for the transmitter. It may be in the range 0 to 4, with the following meanings:

Code	Warning Level
0	Off
1	Less than 25% battery life left
2	Less than 10% battery life left
3	Reserved, not used
4	Battery time expired alarm

Examples:

	REQUEST	RESPONSE
QUERY	`!txbwarn(1)?<CR>`	`OK txbwarn(1)=0<CRLF>`

QUERY	**!txbwarn(*)?<CR>**	**OK txbwarn(*)={0,1,0,0,0,0}<CRLF>**

txgain (Transmitter audio gain)

This command may be used as a query to read the transmitter audio gain. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, representing the gain on a scale of 0 to 42. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	**!txgain(1)?<CR>**	**OK txgain(1)=22<CRLF>**
QUERY	**!txgain(*)?<CR>**	**OK txgain**(*)={24,26,20,27,30,30}<CRLF>

txmute (Transmitter mute status)

This command may be used as a query to read the transmitter mute status. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that the transmitter is muted, or "0" meaning that it is not. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	**!txmute(3)?<CR>**	**OK txmute(3)=1<CRLF>**
QUERY	**!txmute(*)?<CR>**	**OK txmute(*)={0,0,0,0,1,0}<CRLF>**

txname (Transmitter name)

This command may be used as a query to read the transmitter name. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is string, with a limit of 15 characters.

Examples:

	REQUEST	RESPONSE
QUERY	`!txname(1)?<CR>`	`OK txname(1)="Chairman"<CRLF>`

txphase (Transmitter audio phase)

This command may be used as a query to read the transmitter audio phase status. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is integer, either "1" meaning that the audio phase inverted (shifted by 180 degrees), or "0" meaning that it is not. If the channel address is wildcarded, then the data type is an array of integer of size 6.

Examples:

	REQUEST	RESPONSE
QUERY	`!txphase(3)?<CR>`	`OK txphase(3)=0<CRLF>`
QUERY	`!txphase(*)?<CR>`	`OK txphase(*)={0,1,0,0,0,0}<CRLF>`

txrolloff (Transmitter low frequency rolloff)

This command may be used as a query to read the transmitter low frequency rolloff. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. If the channel address is wildcarded, then the data type is an array of integer of size 6. The data type is integer and is a code that specifies the low frequency rolloff in Hz. It may be in the range 0 to 5, with the following meanings:

Code	**Low Frequency Rolloff**
0	35 Hz
1	50 Hz
2	70 Hz
3	100 Hz
4	120 Hz
5	150 Hz

Examples:

	REQUEST	RESPONSE

QUERY	`!txrolloff(3)?<CR>`	`OK txrolloff(3)=0<CRLF>`
QUERY	`!txrolloff(*)?<CR>`	`OK txrolloff(*)={1,1,2,1,0,0}<CRLF>`

txstat (Transmitter status)

This command may be used as a query to read *real-time* transmitter status. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data type is array of integer, with a length of 8. The values contained in the array are interpreted by position as follows:

Position	**Value**
1	Link Status (0 = no link, 1 = link established)
2	Battery timer (elapsed time in minutes)
3	Battery warning status (0 = ok, 1 = less than 25%, 2 = less than 10%, 4 = timer alarm)
4	Mute status (0 = unmuted, 1 = muted)
5	Front panel lock status (0 = unlocked, 1 = locked)
6	Audio meter (in range -80 to +5 dB, referenced to the threshold of limiting)
7	Audio limiter status (0 = not in limiting, 1 = in limiting)
8	Audio clipping status (0 = not clipping, 1 = clipping)

Data is valid only when link is established (link status = 1).

Example:

	REQUEST	RESPONSE
QUERY	`!txstat(1)?<CR>`	`OK txstat(1)={1,72,0,0,0,-18,0,0}<CRLF>`

txver (Transmitter firmware version)

This command may be used as a query to read the transmitter's firmware version number. The channel is specified by using the address syntax. Addresses must be in the range 1 to 6. The data is a string type.

Example:

	REQUEST	RESPONSE
QUERY	**`!txver(1)?<CR>`**	**`OK txver(1)="1.0"<CRLF>`**

DR Network Setup Commands

defgate	Default gateway
dhcpen	DHCP enable
httpport	HTTP port number
ipaddr	IP address
macaddr	MAC address
netmask	Network mask
tcpport	TCP port number

Termination: all commands are terminated with an ASCII **carriage return** character (hex code 0x0D), represented by **<CR>** in the examples below. All responses are terminated with an ASCII **carriage return, line feed** pair (hex codes 0x0D, 0x0A), represented by **<CRLF>** in the examples below. An ellipsis (**...**) represents members of an array that have been omitted from an example for the sake of brevity.

Verbose response: commands prefixed with an exclamation point (bang) character result in a "verbose" response containing both the name of the property being addressed and its current value (if any). The verbose response returns the property/value pair in the "assignment" form, for example **`OK ingn(2)=40 <CRLF>`**. This supports certain 3rd party control programming styles where the response needs to be self-describing.

defgate (default gateway)

This command may be used as a query to read the Default Gateway address, or as an update to set it. The data type is string, containing the address in IP "dotted quad" format.

Example:

	REQUEST	RESPONSE
QUERY	**`defgate?<CR>`**	**`OK "172.16.4.1"<CRLF>`**
UPDATE	**`defgate="172.16.4.1"<CR>`**	**`OK<CRLF>`**

dhcpen (DHCP enable)

This command may be used as a query to read the DHCP enable status, or as an update to set it. The data type is integer, either "1" meaning that the DHCP client feature is enabled, or "0" meaning that it is not. If enabled, DHCP (Dynamic Host Configuration Protocol) is used at power up to obtain an IP address. **Note:** If this setting is changed, the new value takes effect the next time the device is powered up.

Example:

	REQUEST	RESPONSE
QUERY	**`dhcpen?<CR>`**	**`OK 0<CRLF>`**
UPDATE	**`dhcpen=1<CR>`**	**`OK<CRLF>`**

httpport (HTTP port number)

This command may be used as a query to read the HTTP port number assignment, or as an update to set it. The data type is integer, in the range 0 to 65535, representing the port number used for HTTP connections to the device. The default value is 80.

Example:

	REQUEST	RESPONSE
QUERY	**`httpport?<CR>`**	**`OK 80<CRLF>`**
UPDATE	**`httpport=80<CR>`**	**`OK<CRLF>`**

ipaddr (IP address)

This command may be used as a query to read the IP address of the device, or as an update to set it. The data type is string, containing the address in IP "dotted quad" format.

Example:

	REQUEST	RESPONSE
QUERY	**`ipaddr?<CR>`**	**`OK "172.16.4.151"<CRLF>`**
UPDATE	**`ipaddr="172.16.4.151"<CR>`**	**`OK<CRLF>`**

macaddr (MAC address)

This command may be used as a query to read the ethernet MAC address of the device. The data type is string, containing the address in IEEE MAC-48 format.

Example:

	REQUEST	RESPONSE
QUERY	**`macaddr?<CR>`**	**`OK "00-24-34-32-00-22"<CRLF>`**

netmask (network mask)

This command may be used as a query to read the Network Mask, or as an update to set it. The data type is string, containing the mask in IP "dotted quad" format.

Example:

	REQUEST	RESPONSE
QUERY	**`netmask?<CR>`**	**`OK "255.255.255.0"<CRLF>`**
UPDATE	**`netmask="255.255.255.0"<CR>`**	**`OK<CRLF>`**

tcpport (TCP port number)

This command may be used as a query to read the TCP port number assignment, or as an update to set it. The data type is integer, in the range 0 to 65535, representing the port number used for TCP connections to the device. The default value is 4080.

Example:

	REQUEST	RESPONSE
QUERY	**`tcpport?<CR>`**	**`OK 4080<CRLF>`**
UPDATE	**`tcpport=4080<CR>`**	**`OK<CRLF>`**

DR Macro Management & Related Commands

exit	Exit from macro
macro	Macro command
macroclr	Macro clear
macroti	Macro title
macvrport	Macro verbose response port
ropmac	"Run on Powerup" macro
run	Run a macro
sendstr	Send string to port

Termination: all commands are terminated with an ASCII **carriage return** character (hex code 0x0D), represented by **<CR>** in the examples below. All responses are terminated with an ASCII **carriage return, line feed** pair (hex codes 0x0D, 0x0A), represented by **<CRLF>** in the examples below. An ellipsis (**...**) represents members of an array that have been omitted from an example for the sake of brevity.

Verbose response: commands prefixed with an exclamation point (bang) character result in a "verbose" response containing both the name of the property being addressed and its current value (if any). The verbose response returns the property/value pair in the "assignment" form, for example **`OK ingn(2)=40 <CRLF>`**. This supports certain 3rd party control programming styles where the response needs to be self-describing.

exit (exit a macro)

This command may be used to exit a **macro**, usually from within a conditional (if-then-else) statement within the macro.

Example:

	REQUEST	RESPONSE
COMMAND	**`exit<CR>`**	**`OK<CRLF>`**

macro (macro command)

This command may be used as a query to read one command from a macro, or as an update to set a command. The command is specified by using the 2 dimensional address syntax. Addresses for the first dimension specify the macro and must be in the range 1 to 256. Addresses for the second dimension specify the index of the command within the macro and must be in the range 1 to 64. The data type is string, with a limit of 110 characters.

Note: String arguments in commands need to be passed in **quoted** form, contained in a pair of **double-quote** (") characters. A problem arises when using the `macro` command to read or write a command that already contains double-quote characters, for example the command `desc="whatever"`, which contains the quoted string argument `"whatever"`. The solution is to **escape** the double quotes within `desc="whatever"` so that it can itself be passed as a string argument for the `macro` command. This is done by preceding the double-quote characters with a **backslash** character like this: `desc=\"whatever\"`. Now it can be passed as a string argument to the `macro` command: `macro(1,1)="desc=\"whatever\""`. Since the **backslash** serves as the escape character in quoted-string arguments, it too must be escaped if it is part of the string, so `"foo\bar"` would become `"foo\\bar"`. If necessary, any character, printable or non-printable, can be represented in the hexadecimal escaped form `\xHH` where `HH` is any 2-digit hexadecimal number. The special escaped character forms `\r` (carriage return), `\n` (new line) and `\t` (tab) are also recognized.

Examples:

	REQUEST	RESPONSE
QUERY	**`macro(1,3)?<CR>`**	**`OK "ingn(3)=55"<CRLF>`**
QUERY	**`macro(1,4)?<CR>`**	**`OK "desc=\"Unit #1 East\""<CRLF>`**
UPDATE	**`macro(12,50)="xpmt(2,10)=1"<CR>`**	**`OK<CRLF>`**
UPDATE	**`macro(12,51)="desc=\"Classroom 17\""<CR>`**	**`OK<CRLF>`**

macroclr (macro clear)

This command may be used to clear a macro. All lines in the macro will be erased. Addresses must be in the range 1 to 256.

Example:

	REQUEST	RESPONSE
COMMAND	**`macroclr(3)<CR>`**	**`OK<CRLF>`**

macroti (macro title)

This command may be used as a query to read the title of a macro, or as an update to set the title. The macro is specified by using the address syntax. Addresses must be in the range 1 to 256. The data type is string, with a limit of 30 characters.

Note: String arguments in commands need to be passed in **quoted** form, contained in a pair of **double-quote** (") characters. A problem arises when using the `macroti` command to read or write a string that already contains double-quote characters, for example: `The "Hula" Room`. The solution is to **escape** the double quotes within `The "Hula" Room` so that it can be passed as a string argument for the `macroti` command. This is done by preceding the double-quote characters with a **backslash** character like this: `The \"Hula\" Room`. Now it can be passed as a string argument to the `macroti` command: `macroti(1,1)="The \"Hula\" Room"`. Since the **backslash** serves as the escape character in quoted-string arguments, it too must be escaped if it is part of the string, so `"foo\bar"` would become `"foo\\bar"`. If necessary, any character, printable or non-printable, can be represented in the hexadecimal escaped form `\xHH` where `HH` is any 2-digit hexadecimal number. The special escaped character forms `\r` (carriage return), `\n` (new line) and `\t` (tab) are also recognized.

Examples:

	REQUEST	RESPONSE
QUERY	**`macroti(1)?<CR>`**	**`OK "Sidebar nbr 2"<CRLF>`**
UPDATE	**`macroti(12)="Setup #3 West"<CR>`**	**`OK<CRLF>`**

macvrport (macro verbose response port)

This command may be used as a query to determine the default port for verbose responses generated by macro execution, or as an update to set the port. The data is an integer type with the following possible values:

- **1** - RS232 port
- **2** - TCP port 1
- **3** - TCP port 2

When commands are executed within a macro, the response to the command is normally discarded. However, if a command is prefixed with an exclamation point (bang) character (verbose mode), the response is sent to either the RS232 port or the TCP port to provide feedback to 3rd party control systems connected to these ports. This command is used to control which port receives the responses to verbose mode commands contained in a macro.

Examples:

	REQUEST	RESPONSE
QUERY	**`macvrport?<CR>`**	**`OK 1<CRLF>`**
UPDATE	**`macvrport=2<CR>`**	**`OK<CRLF>`**

ropmac ("run on powerup" macro)

This command may be used as a query to determine the "run on powerup" macro for the device. It may also be used as an update to set the macro. The data is an integer type in the range 0 to 256, where "0" has the special meaning "none".

Examples:

	REQUEST	RESPONSE
QUERY	**`ropmac?<CR>`**	**`OK 0<CRLF>`**
UPDATE	**`ropmac=5<CR>`**	**`OK<CRLF>`**

run (run a macro)

This command may be used to run a macro. A single macro may be run by using the command form. In this case the macro is specified by using the address syntax. Addresses must be in the range 1 to 256. More than one macro may be run with a single command by using the update form. In this case the data type is array of integer, with a variable length in the range 1 - 16. The values contained in the array specify which macros to run.

Examples:

	REQUEST	RESPONSE
COMMAND	**`run(3)<CR>`**	**`OK<CRLF>`**
UPDATE	**`run={1,3,5}<CR>`**	**`OK<CRLF>`**

sendstr (send string to port)

This command may be used as an update to send an arbitrary ASCII string to either the RS232 port or one of the TCP ports of a DR receiver. The port is specified by using the address syntax. The port address may be one of the following:

- **1** - RS232 port
- **2** - TCP port 1
- **3** - TCP port 2

The ASCII characters to be sent are given as the argument to the `sendstr` command. The data type is string, with a limit of 127 characters in a quoted string argument, and 255 characters if the argument is supplied as a string variable reference (in macros only).

This command is intended for use within macros and is useful for sending strings to 3rd party equipment attached to the RS232 or TCP ports for control or notification purposes. It may also be sent over a communications port by an external controller, but in this case the argument *must* be a quoted string..

Note: String arguments in commands need to be passed in **quoted** form, contained in a pair of **double-quote** (") characters. A problem arises when using the `sendstr` command to read or write a string that already contains double-quote characters, for example: `The "Hula" Room`. The solution is to **escape** the double quotes within `The "Hula" Room` so that it can be passed as a string argument for the `macroti` command. This is done by preceding the double-quote characters with a **backslash** character like this: `The \"Hula\" Room`. Now it can be passed as a string argument to the `macroti` command: `macroti(1,1)="The \"Hula\" Room"`. Since the **backslash** serves as the escape character in quoted-string arguments, it too must be escaped if it is part of the string, so `"foo\bar"` would become `"foo\\bar"`. If necessary, any character, printable or non-printable, can be represented in the hexadecimal escaped form `\xHH` where `HH` is any 2-digit hexadecimal number. The special escaped character forms `\r` (carriage return), `\n` (new line) and `\t` (tab) are also recognized.

Examples:

	REQUEST	RESPONSE
UPDATE	**`sendstr(1)="FI22;"<CR>`** (RS232 port, quoted string argument)	**`OK<CRLF>`**
UPDATE	**`sendstr(1)=@temp@`** (RS232 port, variable argument, works *only* in a macro)	**`OK<CRLF>`**