

LINN PRODUCT SOFTWARE

**Majik Kontrol Pre-amplifier - RS232 ASCII Interface Specification and Commands
Version 01.00**

Last Revision: 15 June 2006
Eamonn Brady

Revision History

<i>Revision</i>	<i>Description</i>	<i>Author</i>	<i>Date</i>
00.01	Initial Draft (modified from Exotik Exotik+DA)	Ian Wilson	3 Feb 2006
00.02	Added Hardware version and fixed some inaccuracies	Eamonn Brady	13 February 2006
00.03	Fixed some typos	Eamonn Brady	15 June 2006
01.00	Linfo release	Eamonn Brady	19 June 2006

Table of Contents

INTRODUCTION	1
1: MESSAGE PROTOCOL	2
1.1: Overview	2
1.2: Message Syntax	2
1.3: Identifier Considerations	3
1.4: Syntax of Commands and Responses	4
1.4.1: Command Syntax	4
1.4.1.1: Command Help	4
1.4.1.2: Command	4
1.4.2: Solicited Response Overview	5
1.4.2.1: Initial Response	5
1.4.2.1.1: Initial Response Failure	5
1.4.2.2: Final Response	5
1.4.2.2.1: Final Response Failure	5
1.4.3: Unsolicited Response Overview	6
1.4.3.1: Unsolicited Response	6
1.4.3.2: Unsolicited Response Events	6
2: SYSTEM COMMANDS	7
2.1: Identity Commands	7
2.1.1: ID	7
2.1.2: GID	7
2.2: Communication Commands	9
2.2.1: BAUD	9
2.2.2: RESET	9
2.2.3: ECHO	9
2.2.3.1: Power_Up Message	9
2.3: Polling Command	11
2.3.1: POLL	11
2.3.2: Polling Explained	12
2.4: Status Command	13
2.4.1: STATUS	13
2.4.1.1: Status Codes	13
2.5: IR	15
2.6: INIT	15
2.7: VERSION	15
2.8: COUNTER	16
3: MAJIK KONTROL COMMANDS	17
3.1: Command Help	Error! Bookmark not defined.

3.2: System Commands	17
3.3: Other Commands	18
3.3.1: STANDBY	18
3.3.2: MUTE	18
3.3.3: VOLUME	19
3.3.4: BALANCE	19
3.3.4.1: BALANCE / BALANCE_LR (left/right)	19
3.3.5: INPUT	20
3.3.5.1: AUDIO	20
3.3.6: RECORD	20
3.3.7: NORMALISE	21
3.3.8: SYSTEM	21
3.3.8.1: VOLUME	21
3.3.8.2: STATUS	21
APPENDICE A : FORMAT OF COMMAND TABLE	22
APPENDICE B : ESCAPE SEQUENCES	23
APPENDICE C : COMMUNICATIONS SETTINGS	24

Introduction

This document describes how to control the Majik Kontrol Pre-amplifier and peripherals through an RS232 interface.

There are three main sections to this document:

1: Message Protocol

- This section describes how commands are constructed and how they may be used.

2 System Commands

- This section lists the commands, which allow the Majik Kontrol to be used as part of a system driven through an RS232 interface.

3: Majik Kontrol Commands

- This section defines a list of commands for controlling Majik Kontrol. This section is further subdivided into the subsets of the Majik Kontrol commands.

1: Message Protocol

1.1: Overview

The RS232 interface on the Majik Kontrol allows it to be controlled by a touch screen, PC or any computer with an RS232 port. The Majik Kontrol obeys the commands received through the RS232 interface and replies to confirm successful or unsuccessful operation.

The RS232 interface uses an initial response then final response method to acknowledge receiving the command and then completing the task. The interface also supports device and group identifiers to allow a number of units to be connected together. The controlling device can also supply a source identification, which the Majik Kontrol will echo as the destination for the replies.

Previous products were termed as slave devices, in regards that nothing was transmitted until something was received, e.g. a task or status command. The Majik Kontrol however, will transmit unsolicited messages when something within the product changes, e.g. the volume changes.

1.2: Message Syntax

The general syntax is as follows: (Source_ID) (Group_ID) (Destination_ID) Command NL

Where:

Source_ID Syntax: #Source_ID#

is a unique identifier, used to denote the source of the message. Enclosed by the '#' delimiter, the maximum identifier size is 20 ASCII alphanumeric characters (excluding spaces).

Destination_ID Syntax: @Destination_ID@

is a unique identifier, used to denote the destination of the message. Enclosed by the '@' delimiter, the maximum identifier size is 20 ASCII alphanumeric characters (excluding spaces).

Group_ID Syntax: &Group_ID&

is a unique identifier, used to denote a specific group of products. Enclosed by the '&' delimiter, the maximum identifier size is 20 ASCII alphanumeric characters (excluding spaces).

Command Syntax: \$Command\$

is the command from the host for the product. Enclosed by the '\$' delimiter.

NL Syntax: 13dec and 10dec (0Dhex and 0Ahex)

are the line termination characters, carriage return and line feed.

Note:

Nesting of fields is not permissible, nor is the use of the special delimiter characters as part of the field strings themselves, unless they are expressed as an escape sequence (see Appendix B : Escape Sequences).

Spaces are permissible before and after an identifier, but are not allowed within the actual identifier, unless they are expressed as an escape sequence (see Appendix B : Escape Sequences).

For example, # recorddeck # is valid whereas # record deck # is invalid.

By using an escape sequence, the second example becomes valid, i.e. # record\0x20deck #

1.3: Identifier Considerations

The full transmission format uses four fields as shown.

```
#Source_ID# &Group_ID& @Destination_ID@ $Message$
```

Where fields are omitted the results are defined in the following notes.

```
..... $Message$ refer to note 1
..... @Destination_ID@ $Message$ refer to note 2
..... &Group_ID& ..... $Message$ refer to note 3
..... &Group_ID& @Destination_ID@ $Message$ refer to note 4
#Source_ID# ..... $Message$ refer to note 5
#Source_ID# ..... @Destination_ID@ $Message$ refer to note 6
#Source_ID# &Group_ID& ..... $Message$ refer to note 7
#Source_ID# &Group_ID& @Destination_ID@ $Message$ refer to note 8
```

Note Details

- 1 - A product recognising the command will issue an initial response and try to perform the task.
 - A successful or unsuccessful final response will be issued subsequently.
 - Products not recognising the command will remain silent.
 - If no product recognises the command then there will be no reply.
 - If more than one product recognises the command then there may be a comms clash on the replies.
- 2 - The destination product is responsible for all replies.
 - Invalid commands will generate an error response.
 - The replying product will transfer the destination to the source field on a reply.
 - All products not matching the destination must remain silent and not attempt to handle the command.
 - If two products have the same id, then a comms clash may occur.
- 3 - All products within the group should attempt the task.
 - Products out with the group should ignore the task.
 - There are no replies from any boxes.
- 4 - All products within the group should attempt the task.
 - Products out with the group should ignore the task.
 - Only the product, which matches the destination identity, should reply.
 - Invalid commands will generate an error response.
 - If there are more than two products in the group with the same destination identity then a comms clash may occur.
 - The destination identity becomes the source identity in any reply traffic.
- 5 - As for note 1, with the source identity becoming the destination identity in any replies.
- 6 - As for note 2, with the source identity becoming the destination identity in any replies.
- 7 - As for note 3. There are no replies.
- 8 - As for note 4, with the source identity becoming the destination identity in any replies.

1.4: Syntax of Commands and Responses

1.4.1: Command Syntax

The command message has two variations:

1.4.1.1: Command Help

This allows the host to find out what type of parameters the command requires.

Syntax: **\$? Command\$NL**

Where: **\$** is the command start delimiter
 ? is a request for help
 Command is the command help request is for
 \$ is the command end delimiter
 NL are the line termination characters - carriage return, line feed.

Additionally, if 'Command' is a '?' then the command set of the product will be provided, with an initial response followed by a final response for each command supported by the product.

This is a change to the previous method, where the command set of the product was output as a single response, with each command being separated from the next by a space and no help text was included.

1.4.1.2: Command

This is the method by which the host controls the product

Syntax: **\$Command (Param (Param))\$NL**

Where: **\$** is the command start delimiter
 Command is the command string
 Param is the parameter string (0 or more)
 \$ is the command end delimiter
 NL are the line termination characters - carriage return, line feed.

Note:

Parameters required are command dependent

1.4.2: Solicited Response Overview

When replies are made an initial response and final response are issued. It is unwise for the host to issue further commands until the final response has been received. Section 1.3: Identifier Considerations, describes the action of identifiers on these replies and specifies rules which may also suppress the replies.

1.4.2.1: Initial Response

This will be given on receipt of a valid command and for a positive acknowledge will be of the form:

```
(Source_ID) (Group_ID) (Destination_ID) !
```

In this way, the host quickly knows that the destination has received and understood the command.

1.4.2.1.1: Initial Response Failure

This will be given on receipt of an invalid command and will be of the form:

```
(Source_ID) (Group_ID) (Destination_ID) !$FAIL sc fn$
```

Where 'sc' is a status code (see section 2.4.1.1: Status Codes) specifying why the task could not be completed, and 'fn' specifies which field was responsible.

Note: *There is no final response.*

1.4.2.2: Final Response

This will be given on completion of the task and will be of the form:

```
(Source_ID) (Group_ID) (Destination_ID) !$Status_String$
```

The status string will be a unique response to the originating command.

1.4.2.2.1: Final Response Failure

This will be given where a task could not be completed and will be of the form:

```
(Source_ID) (Group_ID) (Destination_ID) !$FAIL sc fn$
```

Where 'sc' is a status code (see section 2.4.1.1: Status Codes) specifying why the task could not be completed, and 'fn' specifies which field was responsible.

Note:

¹ *In all cases, identifiers will only be returned as part of the response if supplied as part of the command (refer to section 1.3: Identifier Considerations for further details).*

² *Fields are numbered from left to right, starting at 1.*

1.4.3: Unsolicited Response Overview

Unsolicited responses are an addition to the RS232 protocol, and are generated automatically by the product to inform the host of a change to the products status.

1.4.3.1: Unsolicited Response

This will be given at any time during the operation of the product and will be of the form:

`(Source_ID) $Status_String$`

The major differences between solicited and unsolicited responses are as follows:

1. Unsolicited messages can occur at any time (if activated).
2. Source identifier, if present within product settings, will always form part of the message.
3. No exclamation mark is included before the command delimiter.

Note:

Refer to user guide regarding activation of unsolicited responses (user option '*Enable RS232 Events*').

1.4.3.2: Unsolicited Response Events

Should the following events occur, and the Majik Kontrol change have it's parameters changed for any of the following events, then an unsolicited response should be generated.

1. Volume increment/decrement from IR / RC5 command or front panel keypad.
2. Input Source profile selection from IR / RC5 command or front panel keypad.
3. Record Path changes from IR / RC5 or from front panel keypad.
4. Decode algorithm change from IR / RC5 / front panel keypad or stream change.
5. Balance adjustment increment/decrement from IR / RC5 command or front panel keypad.
6. Speaker trim adjustment increment/decrement from IR / RC5 command or front panel keypad.




2: System Commands




The following commands allow the Majik Kontrol to be part of a system driven through an RS232 interface.




2.1: Identity Commands

2.1.1: ID

Configure the product on a one to one basis




 <code>\$ID identifier\$</code>
 <code>\$ID identifier\$</code>
 <i>Write product identifier</i>




 <code>\$ID ~identifier\$</code>
 <code>\$ID\$</code>
 <i>Remove product identifier</i>




 <code>\$ID ?\$</code>
 <code>\$ID identifier\$</code>
 <i>Return product identifier</i>

2.1.2: GID

Configures a product as part of a group so that it can be accessed a number of ways

 <code>\$GID identifier\$</code>
 <code>\$GID identifier\$</code>
 <i>Write group identifier (product now becomes part of a group of products)</i>

 <code>\$GID ~identifier\$</code>
 <code>\$GID identifier [identifier [...]]\$</code>
 <i>Remove a product from a particular group</i>

 <code>\$GID ?\$</code>
 <code>\$GID identifier [identifier [...]]\$</code>
 <i>Return list of currently defined group identifiers from product</i>

Notes on Groups:

A product can be a member of at most 5 groups to allow it to be addressed in a variety of ways.




While in group mode, products with the same group ID will react in the same way to product specific commands sent to them using the Group_ID syntax (&group_id&).




In addition, products in Group Mode will not acknowledge receipt of commands from the host. This is to avoid all products in the group potentially responding at the same time.

Each product can be polled individually at the end of a group mode command to check they have all been updated correctly.

2.2: Communication Commands

2.2.1: BAUD

 \$BAUD baudrate\$
 !\$BAUD baudrate\$
 <i>Select new baud rate from the following: 4800, 9600, 14400³, 19200, 28800³, 38400, 57600³, 115200³, 230400³</i>

 \$BAUD ?\$
 !\$BAUD baudrate\$
 <i>Returns current baud rate (see above)</i>

Note:

¹ Initial and final responses will be at the current baud rate, before the new baud rate is implemented.




² Baud rate defaults to 9600 when the product is initialised.

³ New baud rates supported by this product.




⁴ 1200 & 2400 baud rate not supported by this product.

2.2.2: RESET

Return product comms buffers to a known state

 \$RESET\$
 !\$RESET\$
 <i>Clear communications buffer on product</i>

2.2.3: ECHO

 \$ECHO text\$
 !\$ECHO <text>\$
 <i>Echo's the text back enclosed in < and ></i>

This command is used ease the burden of initial set-up of host-product communications, the product will echo the parameter provided back to the host.

Note:

If no identifiers are supplied with this command, then all devices connected to a system will respond, which may result in a comms clash.

2.2.3.1: Power_Up Message

A power up message is provided which is transmitted to the host in order to verify that the host / product link is working.




The power up message on the Majik Kontrol is as follows: **!\$MAJIK_KONTROL\$**




Note: Please refer to user options with regards to Power Up Message




2.3: Polling Command




2.3.1: POLL

Polling is used to extract details of all products connected to the host

 <code>\$POLL START\$</code>
 <code>!\$POLL START\$</code>
 <i>Marks the start of polling</i>

 <code>\$POLL ID\$</code>
 <code>!\$POLL ID product_identifier\$</code>
 <i>Returns product identifier</i>

 <code>\$POLL SLEEP\$</code>
 <code>!\$POLL SLEEP\$</code>
 <i>Product responding to this will ignore all further commands until 'POLL DONE' is received</i>

 <code>\$POLL DONE\$</code>
 <code>No response to this command</code>
 <i>All products will now return to active operation</i>

Important

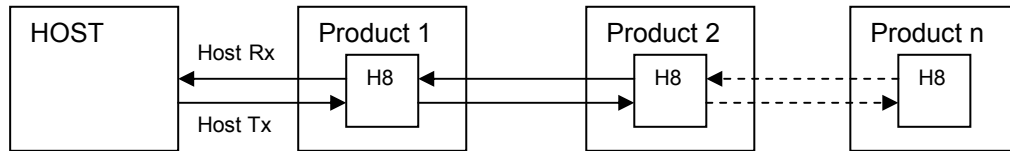
The 'POLL SLEEP' command should be used with the product identifier returned by 'POLL ID'.

If this is not done then all the products will stop responding and the polling sequence will fail.

2.3.2: Polling Explained

The RS232 interface hardware, via the `POLL` command, allows communication to daisy-chained RS232 controlled devices. The devices must be capable of buffering data for transmission as required.

Using this feature allows the host to 'auto-detect' the slave products on the RS232 link.



By taking advantage of this, it is possible to identify what is on the link using the following type of algorithm:

`$POLL START$`

- opens return-path switches in all devices, so only first device in chain can respond

`$POLL ID$`

- all devices respond but only response from first device reaches host

`@dest_1_id@$POLL SLEEP$`

- where 'dest_1_id' is the result of the previous 'POLL ID'
- matching product closes its switch
- product will not respond to any command now until 'POLL DONE' command received.

`$POLL ID$`

- second device can now respond with it's ID

`@dest_2_id@$POLL SLEEP$`

- where 'dest_2_id' is the result of the previous 'POLL ID'
- matching product closes its switch
- product will not respond to any command now until 'POLL DONE' command received.

The 'POLL ID' and 'POLL SLEEP' commands are issued repeatedly until all products have been queried and there is no response from the last 'POLL ID' command.

`$POLL ID$`

- no response since all product id's read, so time-out




`$POLL DONE$`

- resync all products on link again

2.4: Status Command

The status command has been provided as a debugging aid, i.e. the host can find out why a command was not processed.

2.4.1: STATUS

 \$STATUS\$
 !\$STATUS sc (sv)\$
 <i>Return the status of the last command</i>





Where 'sc' is the returned status code and 'sv' is the status value (only used with code 25 for now). Codes are allocated on a block basis for each product with the first 48 codes reserved for general use.




2.4.1.1: Status Codes




The following table lists the General Status Codes which all products support.

Code	Description
00 (0x00)	- No error
01 (0x01)	- Unexpected termination of command line
02 (0x02)	- Unrecognised or misplaced character in command line
03 (0x03)	- Corrupted command message (within \$...\$)
04 (0x04)	- Start of another source identifier, identifier has already been supplied
05 (0x05)	- Start of another group identifier, identifier has already been supplied
06 (0x06)	- Start of another destination identifier, identifier has already been supplied
07 (0x07)	- Source identifier is too large, maximum of 20 characters
08 (0x08)	- Group identifier is too large, maximum of 20 characters
09 (0x09)	- Destination identifier is too large, maximum of 20 characters
10 (0x0A)	- Source identifier corrupted
11 (0x0B)	- Group identifier corrupted
12 (0x0C)	- Destination identifier corrupted
13 (0x0D)	- Unknown group identity
14 (0x0E)	- Unknown destination identity
15 (0x0F)	- Unknown command
16 (0x10)	- Unknown command parameter
17 (0x11)	- Parameter missing from ID command
18 (0x12)	- Unknown product identifier, cannot delete
19 (0x13)	- Parameter missing from GID command
20 (0x14)	- Cannot delete group identifier, unknown
21 (0x15)	- Cannot add new group identifier, already exists
22 (0x16)	- Cannot add new group identifier, list full
23 (0x17)	- Polling must be activated by the POLL START command
24 (0x18)	- Only POLL ID , SLEEP or DONE commands accepted during polling
25 (0x19)	- Message exceeded maximum allowable length - 'sv' defines maximum length (upto and including CR, and excluding LF)
26 (0x1A) up to	- Reserved
47 (0x3F)	




2.5: IR

 \$IR ?\$
 !\$IR ON\$
 !\$IR OFF\$
 Return current IR control status




 \$IR [Y ON]\$
 !\$IR ON\$
 Enable IR control of product

 \$IR [N OFF]\$
 !\$IR OFF\$
 Disable IR control of product

2.6: INIT

 \$INIT\$
 !\$INIT\$
 Resets all product parameters back to factory defaults

2.7: VERSION




 \$VERSION SOFTWARE ?\$
 !\$VERSION SOFTWARE H8\$
 Return current versions of system software.

Notes:

The software versions returned are as follows ...

Field ...	Software for ...	Current version ...
H8	H8 Processor	S0408xxyy

where: 'xx' is the major software revision number
 'yy' is the minor software revision number

 \$VERSION HARDWARE ?\$
 !\$VERSION HARDWARE Mainboard=m Display=d Phono=p\$
 Return current versions of system hardware.




Notes:




The hardware versions returned are as follows ...

where: m, d and p are in the following format:

PCASxxxyyyy and xxx is the board type and yyyy is the board version

2.8: COUNTER

 \$COUNTER POWER ?\$
 !\$COUNTER POWER days:hours:minutes:seconds\$
 <i>Returns total powered up (operational) time.</i>

 \$COUNTER MAINS ?\$
 !\$COUNTER MAINS days:hours:minutes:seconds\$
 <i>Returns total mains connected time</i>

3: Majik Kontrol Commands

The following pages contain the command set for the Majik Kontrol Preamplifier.

Important:

¹ Parameters must be separated from commands and each other by at least one space character

² Where a command can be enabled or disabled then




Y or **ON** will enable (turn on) the setting and **N** or **OFF** will disable (turn off) the setting




3.1: System Commands




The system commands supported by the Majik Kontrol are **ID**, **GID**, **BAUD**, **RESET**, **ECHO**, **POLL**, **STATUS**, **INIT**, **IR**, **VERSION**, **COUNTER** and Power_Up Message. These are all explained in section 2: System Commands of this document.




3.2: Other Commands

3.2.1: STANDBY




 \$STANDBY ?\$
 !\$STANDBY [ON OFF]\$
 <i>Return current standby status.</i>




 \$STANDBY [Y ON]\$
 !\$STANDBY [ON OFF]\$
 <i>Enter standby.</i>




 \$STANDBY [N OFF]\$
 !\$STANDBY [ON OFF]\$
 <i>Exit standby.</i>




 \$STANDBY TOGGLE\$
 !\$STANDBY [ON OFF]\$
 <i>Toggle standby.</i>

3.2.2: MUTE




 \$MUTE ?\$
 !\$MUTE [ON OFF]\$
 <i>Return current mute status.</i>




 \$MUTE [Y ON]\$
 !\$MUTE [ON OFF]\$
 <i>Mute on.</i>




 \$MUTE [N OFF]\$
 !\$MUTE [ON OFF]\$
 <i>Mute off.</i>




 \$MUTE TOGGLE\$
 !\$MUTE [ON OFF]\$
 <i>Toggle mute.</i>




3.2.3: VOLUME

 <code>\$VOLUME ?\$</code>
 <code>!\$VOLUME value\$</code>
 <i>Return current setting.</i>

 <code>\$VOLUME [+ -]\$</code>
 <code>!\$VOLUME value\$</code>
 <i>Increase or decrease current setting by one.</i>

 <code>\$VOLUME [+ -]value\$</code>
 <code>!\$VOLUME value\$</code>
 <i>Increase or decrease current setting by supplied value.</i>




 <code>\$VOLUME = [+]value\$</code>
 <code>!\$VOLUME value\$</code>
 <i>Set to absolute value supplied.</i>




 <code>\$VOLUME LIMITS\$</code>
 <code>!\$VOLUME LIMITS min_value max_value\$</code>
 <i>Return minimum and maximum settings (0 to 100 inclusive).</i>




Note: The Majik Kontrol volume can be specified in half steps, for example, a volume of 75.5 is now permissible.




3.2.4: BALANCE



3.2.4.1: BALANCE / BALANCE_LR (left/right)

 <code>\$(BALANCE BALANCE_LR) ?\$</code>
 <code>!\$(BALANCE BALANCE_LR) value\$</code>
 <i>Return current setting (-ve is left biased, +ve is right biased).</i>

 <code>\$(BALANCE BALANCE_LR) [+ -]\$</code>
 <code>!\$(BALANCE BALANCE_LR) value\$</code>
 <i>Increase or decrease current setting by one.</i>

 <code>\$(BALANCE BALANCE_LR) [+ -]value\$</code>
 <code>!\$(BALANCE BALANCE_LR) value\$</code>
 <i>Increase or decrease current setting by supplied value.</i>

 <code>\$(BALANCE BALANCE_LR) = [+ -]value\$</code>
 <code>!\$(BALANCE BALANCE_LR) value\$</code>
 <i>Set to absolute value supplied.</i>

 <code>\$(BALANCE BALANCE_LR) LIMITS\$</code>
 <code>!\$(BALANCE BALANCE_LR) LIMITS min_value max_value\$</code>

<i>Return minimum and maximum settings (-10 to +10 inclusive).</i>
--

3.2.5: INPUT

3.2.5.1: AUDIO

\$INPUT AUDIO ?\$
!\$INPUT AUDIO name\$
<i>Return currently selected audio input</i>

\$INPUT AUDIO name\$
!\$INPUT AUDIO name\$
<i>Select audio input</i>

where **name** is one of the following

NONE	Select no audio input
INPUT [1...6]	Select analogue audio input 1-6
ANALOGKNEKT	Select knekt analogue audio input

3.2.6: RECORD

\$RECORD ?\$
!\$RECORD [OFF NONE input TO output]\$
<i>Return current record path details.</i>

\$RECORD OFF\$
!\$RECORD OFF\$
<i>Disable current record path.</i>

\$RECORD ON\$
!\$RECORD [NONE input TO output]\$
<i>Enable current (last) record path.</i>

\$RECORD input TO output\$
!\$RECORD [INVALID_INPUT INVALID_OUTPUT input TO output]\$
<i>Set up a record connection from specified input(s) to record output.</i>

Where **input** is...




[audio_input] ... name of an audio input

refer to *INPUT AUDIO* commands for a list of relevant input names.

and **output** is one of the following ...




ANALOG ... analog output




3.2.7: NORMALISE

 <code>\$NORMALISE\$</code>
 <code>!\$NORMALISE\$</code>
 <i>Normalise the audio settings</i>




3.2.8: SYSTEM

3.2.8.1: VOLUME

 <code>\$\$SYSTEM VOLUME ?\$</code>
 <code>!\$SYSTEM VOLUME level\$</code>
 <i>Return current maximum system volume.</i>

 <code>\$\$SYSTEM VOLUME level\$</code>
 <code>!\$SYSTEM VOLUME level\$</code>
 <i>Set maximum system volume.</i>






3.2.8.2: STATUS

 <code>\$\$SYSTEM STATUS\$</code>
 <code>!\$SYSTEM STATUS volume source mute surround\$</code>
 <i>Return current system status.</i>

Where ...	volume	is the current volume
	source	is the current source (audio/video/profile)
	mute	is the current mute status [MUTED UNMUTED]
	surround	is the current surround mode

Appendice A : Format Of Command Table

Commands are described using the following format:

 \$COMMAND parameters\$	- actual command
 !\$COMMAND response 1\$	- list of possible responses
	
 !\$COMMAND response n\$	
 <i>Description</i>	- brief description of command

Each table describes one variation of the command, therefore, for a command with five variations there will be five tables.

In cases of a command where there may be more than one form of response, all forms of the response will be listed.

The following conventions apply:

\$COMMAND parameters\$	- is the command variation
!\$COMMAND response\$	- is the response to a command
\$response\$	- is an unsolicited response
!\$FAIL number field\$	- is the response to a failed command
All uppercase words are keywords	- all commands and system parameters must be supplied in uppercase
All lowercase words represent a parameter	- ie. number means supply a numeric value
Parameter's shown as, '[p1 p2 p3]'	- means use one of these values
Parameter's shown as, 'p1 [p2 [...]]'	- means supply one or more values

Appendice B : Escape Sequences

Previous implementations of the RS232 protocol, excluded the use of specific characters within identifiers (#, \$, &, @ and spaces) and the command itself. These characters may now be included by using the escape sequence `\xHH`, where `HH` is a two digit hexadecimal code representing the actual ASCII code of the character.

This, for example, allows identifiers and command field data to contain spaces, which would otherwise be treated as field separators.

For example,	<code>#Record Deck#</code>
now becomes	<code>#Record\x20Deck#</code>
and	<code>!\$ARTIST name of artist\$</code>
becomes	<code>!\$ARTIST name\x20of\x20artist\$</code>

The following (ASCII) characters must be encoded, if they are to be included as part of an identifier or as part of a command.

- | | | |
|-------------|----------------------|------------------------------------|
| - 32 (0x20) | space | - field separator |
| - 35 (0x23) | # hash sign | - source identifier delimiter |
| - 36 (0x24) | \$ dollar sign | - command delimiter |
| - 38 (0x36) | & ampersand | - group identifier delimiter |
| - 64 (0x40) | @ commercial at sign | - destination identifier delimiter |
| - 92 (0x5C) | \ backslash | - escape sequence |
- Additionally, top-bit set (ASCII codes 128-255) characters can now also be included, using the same method.

Note:

¹ *Characters within the ranges 0 to 31, and 128 to 159 should not be used.*

² *The main use of escape sequences will be to output Album, Artist and Track names. It is therefore suggested that hosts do not use escape sequences unless absolutely necessary, thereby limiting any possible problems when this product is connected to a daisy chain of products which includes previous incarnations of Linn equipment (software in these products will simply treat them as raw ASCII data).*

Appendice C : Communications Settings

The Majik Kontrol uses the following communications settings:

- 7 bits data
- 1 stop bit
- even parity
- baud rate specified by host (initially 9600)