



LINN

Linn Majik CD™ - RS232 ASCII Interface Specification and Commands

Revision 1.00

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
2: SYSTEM COMMANDS	7
2.1: Identity Commands	7
2.1.1: ID	7
2.1.2: GID	7
2.2: Communication Commands	8
2.2.1: BAUD	8
2.2.2: RESET	8
2.2.3: ECHO	8
2.2.3.1: Power_Up Message	8
2.3: Polling Command	9
2.3.1: POLL	9
2.3.2: Polling Explained	10
2.4: Status Command	11
2.4.1: STATUS	11
2.4.1.1: Status Codes	11
2.5: IR	12
2.6: INIT	12
2.7: CHECKSUM	12
2.8: COUNTER	12
2.9: VERSION	13
3: MAJIK CD COMMANDS	15

3.1: Command Help	15
3.2: System Commands	15
3.3: DISC COMMANDS	16
3.3.1: OPEN	16
3.3.2: CLOSE	16
3.3.3: PLAY	16
3.3.4: PAUSE	16
3.3.5: STOP	16
3.3.6: MODE	17
3.3.7: TRACK	18
3.3.8: DISCINFO	19
3.3.9: SEARCH	19
3.3.10: TIME	20
3.3.11: REPEAT	21
3.3.12: SKIP	22
3.3.13: KEY	22
3.4: OTHER COMMANDS	23
3.4.1: STANDBY	23
3.4.2: SETUP	23
3.4.3: SPDIFOUTPUT	24
3.4.4: VFD_TEST	24
APPENDICE A : FORMAT OF COMMAND TABLE	25
APPENDICE B : ESCAPE SEQUENCES	26
APPENDICE C : COMMUNICATIONS SETTINGS	26
APPENDICE D : IGNORED DISC COMMANDS	27

Introduction

This document describes how to control the MAJIK CD product 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 CD to be used as part of a system driven through an RS232 interface.

3: Majik CD Commands

- This section defines a list of commands for controlling MAJIK CD. This section is further subdivided into the subsets of the MAJIK CD commands.
- These are :-
 - Disc Commands
 - Other Commands

1: Message Protocol

1.1: Overview

The RS232 interface on the MAJIK CD allows it to be controlled by a touch screen, PC or any computer with an RS232 port. The MAJIK CD 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 CD 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 CD however, will transmit unsolicited messages when something within the product changes, e.g. the disc stops playing.

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*

**Not
e** **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.

Note:

Command help is product dependent and is implemented on the MAJIK CD.

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:

Please refer to user options with regards to unsolicited responses (events).

2: System Commands

The following commands allow the MAJIK CD 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 2400 baud rate not supported by this product.

2.2.2: RESET

 \$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 CD is:

!\$MAJIK_CD\$

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

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

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

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

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

Important:

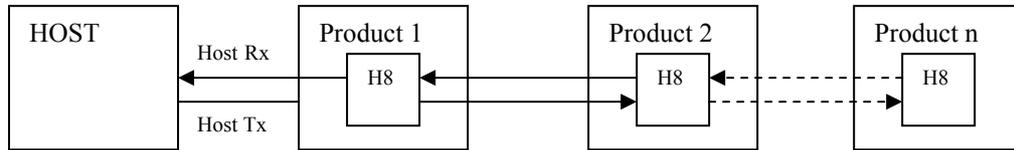
The '**POLL SLEEP**' command should be used with the product identifier returned by '**POLL ID**'.

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.

<i>Code</i>	<i>Description</i>
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)	- Reserved
up to	
47 (0x3F)	

2.5: IR

 \$IR ?\$
 !\$IR ON\$
 !\$IR OFF\$
 <i>Return current IR control status</i>

 \$IR [Y ON]\$
 !\$IR ON\$
 <i>Enable IR control of product</i>

 \$IR [N OFF]\$
 !\$IR OFF\$
 <i>Disable IR control of product</i>

2.6: INIT

 \$INIT\$
 !\$INIT\$
 <i>Resets product back to factory defaults</i>

2.7: CHECKSUM

 \$CHECKSUM ?\$
 !\$CHECKSUM hhhh\$
 <i>Return current software checksum</i>
 <i>Where: 'hhh' is a four digit hexadecimal value</i>

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>

 \$COUNTER WDT ?\$
 !\$COUNTER WDT value\$
 <i>Returns total number of watchdog timer resets.</i>

 \$COUNTER STACK ?\$
 !\$COUNTER STACK value\$
 <i>Returns total number of stack events.</i>

2.9: VERSION

 \$VERSION SOFTWARE ?\$
 !\$VERSION SOFTWARE H8 s version ESS s version MECH m version\$
 Return current versions of system software.

Notes:

The format of 's_version' is 'tpppvvvv'

where: 't' is the software release type ('P' = prototype and 'S' = release)
'ppp' is the software identifier (3 digits)
'vvvv' is the software version (4 digits)

The format of 'm_version' is 'dd.dd.dd.dd'

where: 'dd' is a 2 digit field (four fields in total)

VERSION continued ...

 \$VERSION HARDWARE ?\$
 !\$VERSION HARDWARE b_version serial\$
 !\$VERSION HARDWARE UNKNOWN\$
 Return current hardware board version(s).
 Multiple responses may be received if more than one board is present.

Notes:

The format of 'b_version' is 'PCASb_tm_{Rn}'

where

'PCAS'	is the version header
'b'	is the board identifier (1-16383)
't'	is the major board release type ('P' = prototype and 'L' = release)
'm'	is the major board revision number (1-255)
'R'	is the minor board release type (always 'R')
'n'	is the minor board revision number (1-255)

and 'serial' is a 16 digit hex value.

 \$VERSION HARDWARE PCAS (b (t (m (Rn (serial))))))\$
 !\$VERSION HARDWARE b_version\$
 !\$VERSION HARDWARE UNKNOWN\$
 Return current hardware board version(s), which match search criteria.

Notes:

This version of the command allows for the interrogation of the hardware, using increasingly specific search criteria, in order to return only the most relevant information.

The search criteria is one of the following (see above for details of format) ...

'PCAS'
'PCASb'
'PCASbt'
'PCASb_tm'
'PCASb_tm_{Rn}'
'PCASb_tm_{Rn} serial'

The 'b', 'm' and 'n' fields can be specified as '0', which acts as a wildcard character matching any value.

The 't' field can be specified as '?', which acts as a wildcard character matching any value.

Example 1: \$VERSION HARDWARE PCAS270?\$

Will return all boards, which are PCAS270's regardless of whether they are prototype or release versions.

Example 2: \$VERSION HARDWARE PCAS0L1R0\$

Will return all boards, which are release versions but regardless of board type etc.

3: Majik CD Commands

The following pages contain the command set for the MAJIK CD product.

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: Command Help

Command help is implemented by the MAJIK CD and will give the host details for any given command.

for example: `$? SEARCH$`

replies with: `!$? SEARCH [?| [<|>] speed|STOP]$`

Refer to section 1.4.1.1: Command Help for further information.

3.2: System Commands

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

3.3: DISC COMMANDS

Where **!\$IGNORED command reason\$** is shown in the command response, please refer to 'IGNORED DISC COMMANDS' for further information.

3.3.1: OPEN

 \$OPEN\$
 !\$OPEN OPENING\$
 !\$OPEN OPENED\$
 !\$IGNORED OPEN reason\$
 <i>Open the drawer</i>

3.3.2: CLOSE

 \$CLOSE\$
 !\$CLOSE CLOSING\$
 !\$CLOSE CLOSED\$
 !\$IGNORED CLOSE reason\$
 <i>Close the drawer</i>

3.3.3: PLAY

 \$PLAY\$
 !\$PLAY PLAYING\$
 !\$IGNORED PLAY reason\$
 <i>Start playing disc or continue from current position if paused</i>

3.3.4: PAUSE

 \$PAUSE\$
 !\$PAUSE PAUSED\$
 !\$IGNORED PAUSE reason\$
 <i>Pause playing at current position</i>

3.3.5: STOP

 \$STOP\$
 !\$STOP STOPPED\$
 !\$IGNORED STOP reason\$
 <i>Stop playing the disc.</i>

3.3.6: MODE

 \$MODE\$
 !\$MODE INSTANDBY\$
 !\$MODE POWEREDUP\$
 !\$MODE SETUPMENU\$
 !\$MODE TRAY_UNDEFINED\$
 !\$MODE OPENING\$
 !\$MODE OPENED\$
 !\$MODE CLOSING\$
 !\$MODE CLOSED\$
 !\$MODE DISC_UNDEFINED\$
 !\$MODE LOADING\$
 !\$MODE NODISC\$
 !\$MODE UNKNOWN\$
 !\$MODE CDDA\$
 !\$MODE DATA\$
 !\$MODE PLAY_UNDEFINED\$
 !\$MODE PLAYING\$
 !\$MODE PAUSED\$
 !\$MODE STOPPED\$
 !\$MODE SEARCHING\$
 !\$MODE SCANNING\$
 <i>Return current operational status</i>

Note:

The status list above is essentially a copy of the 'IGNORED Disc Commands' reason list, with the 'UNIT_', 'TRAY_' and 'DISC_' removed (in most cases).

Please refer to 'IGNORED Disc Commands' for further details.

3.3.7: TRACK

 \$TRACK +\$
 !\$TRACK number\$
 !\$IGNORED TRACK reason\$
 <i>Select next (available) track.</i>

 \$TRACK -\$
 !\$TRACK number\$
 !\$IGNORED TRACK reason\$
 <i>Select previous (available) track.</i>

 \$TRACK number\$
 !\$TRACK number\$
 !\$TRACK BADTRACK\$
 !\$IGNORED TRACK reason\$
 <i>Select track number</i>

 \$TRACK ?\$
 !\$TRACK number\$
 !\$IGNORED TRACK reason\$
 <i>Return current track number</i>

 \$TRACK TOT\$
 !\$TRACK TOT number\$
 !\$IGNORED TRACK reason\$
 <i>Return total number of tracks.</i>

3.3.8: DISCINFO

 \$DISCINFO ?\$
 !\$DISCINFO disc_type stream_type
 Return disc type and stream type.

Note:

Where disc type is: *DISC_*
 followed by: *CDDA, HDCD, DATA, UNKNOWN, LOADING, NODISC or UNDEFINED.*

And stream type is: *STREAM_*
 followed by: *MP3, CDDA, DTS, PCM or UNKNOWN.*

3.3.9: SEARCH

 \$SEARCH < speed\$
 !\$SEARCH < speed\$
 !\$IGNORED SEARCH reason\$
 Search backwards through disc until STOP command is received.
 Where speed is 2X, 4X, 6X or 8X

 \$SEARCH > speed\$
 !\$SEARCH > speed\$
 !\$IGNORED SEARCH reason\$
 Search forwards through disc until STOP command is received.
 Where speed is 2X, 4X, 6X or 8X

 \$SEARCH STOP\$
 !\$SEARCH STOP\$
 !\$IGNORED SEARCH reason\$
 Stop searching disc

 \$SEARCH ?\$	Not Implemented
 !\$SEARCH STOP\$	
 !\$SEARCH <\$	
 !\$SEARCH >\$	
 !\$IGNORED SEARCH reason\$	
 Return current search status	

3.3.10: TIME

 \$TIME DISC BEG\$
 !\$TIME DISC BEG minutes seconds\$
 !\$IGNORED TIME reason\$
 <i>Set the time mode to return the elapsed time of the disc</i>

 \$TIME DISC END\$
 !\$TIME DISC END minutes seconds\$
 !\$IGNORED TIME reason\$
 <i>Set the time mode to return the remaining time of the disc</i>

 \$TIME DISC TOT\$
 !\$TIME DISC TOT minutes seconds\$
 !\$IGNORED TIME reason\$
 <i>Return the total time of the disc</i>

 \$TIME TRACK BEG\$
 !\$TIME TRACK BEG minutes seconds\$
 !\$IGNORED TIME reason\$
 <i>Set the time mode to return the elapsed time of the track</i>

 \$TIME TRACK END\$
 !\$TIME TRACK END minutes seconds\$
 !\$IGNORED TIME reason\$
 <i>Set the time mode to return the remaining time of the track</i>

 \$TIME TRACK TOT\$
 !\$TIME TRACK TOT minutes seconds\$
 !\$IGNORED TIME reason\$
 <i>Return the total time of the track</i>

 \$TIME OFF\$
 !\$TIME OFF\$
 !\$IGNORED TIME reason\$
 <i>Set the time mode to off.</i>

 \$TIME ?\$
 !\$TIME [DISC TRACK] BEG minutes seconds\$
 !\$TIME [DISC TRACK] END minutes seconds\$
 !\$TIME [DISC TRACK] TOT minutes seconds\$
 !\$IGNORED TIME reason\$
 <i>Return the time elapsed/remaining/total as set by commands above or by the handset</i>

3.3.11: REPEAT

 \$REPEAT [Y ON]\$
 !\$REPEAT ON\$
 !\$IGNORED REPEAT reason\$
 Turn repeat on.
 If a program is currently active then the program will be repeated, otherwise the entire disc will be repeated.

 \$REPEAT [N OFF]\$
 !\$REPEAT OFF\$
 !\$IGNORED REPEAT reason\$
 Turn repeat off.

 \$REPEAT BEG\$
 !\$REPEAT BEG\$
 !\$IGNORED REPEAT reason\$
 Mark start of repeat section.

 \$REPEAT END\$
 !\$REPEAT END\$
 !\$REPEAT BADREPEAT\$
 !\$IGNORED REPEAT reason\$
 Mark end of repeat section and start to repeat.
 This command must be preceded at some point by a \$REPEAT BEG\$ command.

 \$REPEAT TRACK\$
 !\$REPEAT TRACK\$
 !\$IGNORED REPEAT reason\$
 Repeat current track.

 \$REPEAT ?\$
 !\$REPEAT ON\$
 !\$REPEAT OFF\$
 !\$REPEAT TRACK\$
 !\$REPEAT A\$
 !\$REPEAT A-B\$
 !\$IGNORED REPEAT reason\$
 Return current repeat status.

3.3.12: SKIP

 \$SKIP +\$
 !\$SKIP +\$
 !\$IGNORED SKIP reason\$
 <i>Select next track.</i>

 \$SKIP -\$
 !\$SKIP -\$
 !\$IGNORED SKIP reason\$
 <i>Select previous track.</i>

3.3.13: KEY

 \$KEY [UP DOWN]\$
 !\$KEY [UP DOWN]\$
 !\$IGNORED [UP DOWN] reason\$
 <i>Navigate menu up/down.</i>

 \$KEY [LEFT RIGHT]\$
 !\$KEY [LEFT RIGHT]\$
 !\$IGNORED [LEFT RIGHT] reason\$
 <i>Navigate menu left/right.</i>

 \$KEY ENTER\$
 !\$KEY ENTER\$
 !\$IGNORED ENTER reason\$
 <i>Select menu item etc.</i>

Note: Although the key command will be sent and a response returned, there is no guarantee that the command will be executed (depends on disc type and mode of operation).

3.4: OTHER COMMANDS

3.4.1: 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>

In certain cases, processing of commands will not be possible, in which case the response will be of the form ...

\$IGNORED *command reason*\$

where IGNORED is the ignored field
 command is the command being ignored
 DISABLED is the reason the command was ignored

For example, **\$LISTEN ?\$**

Could respond, **\$IGNORED LISTEN UNIT_INSTANDBY\$**

3.4.2: SETUP

 \$SETUP [ON OFF]\$
 !\$SETUP [ON OFF]\$
 <i>Turn setup menu off or on.</i>

 \$SETUP ?\$
 !\$SETUP [ON OFF]\$
 <i>Return current setup menu status.</i>

3.4.3: SPDIFOUTPUT

 \$SPDIFOUTPUT [OFF RAW LTRTPCM] \$
 !\$SPDIFOUTPUT [OFF RAW LTRTPCM] \$
 <i>Set SPDIF output to supplied value.</i>

 \$SPDIFOUTPUT ?\$
 !\$SPDIFOUTPUT [OFF RAW LTRTPCM] \$
 <i>Return current SPDIF output setting.</i>

3.4.4: VFD_TEST

 \$VFD_TEST [FULL OFF ?] \$
 !\$VFD_TEST [FULL OFF] \$
 <i>Set and/or return current VFD test screen status.</i>
 <i>Used during production to ensure VFD is functioning correctly</i>

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
!\$FAIL sc fn\$	-	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)`	-	means the value is optional
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) | # | - source identifier delimiter |
| - 36 (0x24) | \$ | - command delimiter |
| - 38 (0x36) | & | - group identifier delimiter |
| - 64 (0x40) | @ | - destination identifier delimiter |
| - 92 (0x5C) | \ | - escape sequence |
- Additionally, top-bit set (ASCII codes 128-255) characters can now also be included, using the same method.

Note:

¹ The MAJIK CD supports the ISO 8859-1 character set (with some exceptions, see ²).

² Characters within the ranges 0 to 31, and 128 to 159 should not be used.

³ Character 127 (DELETE) will delete the last character received from the RS232 buffer.

⁴ The main use of escape sequences will be to output Album, Artist and Track names. It is therefore **highly recommended 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 CD uses the following communications settings:

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

Appendice D : IGNORED Disc Commands

While using the disc commands, there will be specific points at which the command does not make sense.

In these cases the response will be of the form ...

\$IGNORED *command* *reason*\$

where	IGNORED	is the ignored field
	<i>command</i>	is the command being ignored
	<i>reason</i>	is the reason the command was ignored.

For example, **\$SEARCH > 2X\$** when the tray is opened

Would respond, **\$IGNORED SEARCH TRAY_OPENED\$**

The following table details the *reason* for a command being ignored.

Reason	Details
UNIT_INSTANDBY	- Product in standby
UNIT_POWEREDUP	- Product powering up
UNIT_SETUPMENU	- Product in setup mode
TRAY_UNDEFINED	- Unknown tray state (product in standby or powering up)
TRAY_OPENING	- Tray in process of opening
TRAY_OPENED	- Tray opened
TRAY_CLOSING	- Tray in process of closing
TRAY_CLOSED	- Tray closed and in process of detecting disc
DISC_UNDEFINED	- Unknown disc state (product in standby, powering up or tray state unknown)
DISC_LOADING	- Determining disc type
DISC_NODISC	- No disc loaded or unknown disc type
DISC_UNKNOWN	- Unknown disc type
DISC_CDDA	- Disc type detected - not ready to process commands
DISC_DATA	- Disc type detected - not ready to process commands
PLAY_UNDEFINED	- Unknown play state (product in standby, powering up or tray/disc state unknown)
PLAY_PLAYING	- Playing - cannot process command in this mode
PLAY_PAUSED	- Paused - cannot process command in this mode
PLAY_PRESTOP	- Pre-Stopped - cannot process command in this mode
PLAY_STOPPED	- Stopped - cannot process command in this mode
PLAY_SEARCHING	- Searching - cannot process command in this mode
PLAY_SCANNING	- Scanning - cannot process command in this mode

Relationship Between Disc Commands And Product State

The following table details the outcome of a command in relation to the current state of the product.

For example:

Command: **\$PLAY\$** when current state is *DISC_NODISC*

Response: **\$IGNORED PLAY DISC_NODISC\$**

✓ denotes command will be ignored (digit denotes that certain variants of command are allowed)

	OPEN	CLOSE	PLAY	PAUSE	STOP	MODE	TRACK	DISCINFO	SEARCH	TIME	REPEAT	SKIP	KEY
UNIT_INSTANDBY	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓
UNIT_POWEREDUP	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	
UNIT_SETUPMENU	✓	✓	✓	✓	✓		✓	✓	✓	✓	✓	✓	
TRAY_UNDEFINED			✓	✓	✓		✓		✓	✓	✓	✓	
TRAY_OPENING			✓	✓	✓		✓		✓	✓	✓	✓	
TRAY_OPENED			✓	✓	✓		✓		✓	✓	✓	✓	
TRAY_CLOSING			✓	✓	✓		✓		✓	✓	✓	✓	
TRAY_CLOSED			✓	✓	✓		✓		✓	✓	✓	✓	
DISC_UNDEFINED			✓	✓	✓		✓		✓	✓	✓	✓	
DISC_LOADING			✓	✓	✓		✓		✓	✓	✓	✓	
DISC_NODISC			✓	✓	✓		✓		✓	✓	✓	✓	
DISC_UNKNOWN			✓	✓	✓		✓		✓	✓	✓	✓	
DISC_CDDA			✓	✓	✓		✓		✓	✓	✓	✓	
DISC_DATA			✓	✓	✓		✓		✓	✓	✓	✓	
PLAY_UNDEFINED			✓	✓	✓		✓		✓	✓	✓	✓	
PLAY_PLAYING													
PLAY_PAUSED										1	✓		
PLAY_PRESTOP				✓			✓		✓	1	✓	✓	
PLAY_STOPPED				✓			✓		✓	1	✓	✓	
PLAY_SEARCHING											✓		
PLAY_SCANNING											✓		

Note:

¹ All TIME commands apart from ?, DISC TOT and TRACK TOT commands will be ignored.