



Technical Documentation

RemoteMNOPL

Product series: mc² series, Nova73 HD

Version: 4.16.0

Revision 08.02.2012

Copyright

All rights reserved. Permission to reprint or electronically reproduce any document or graphic in whole or in part for any reason is expressly prohibited, unless prior written consent is obtained from the Lawo AG.

All trademarks and registered trademarks belong to their respective owners. It cannot be guaranteed that all product names, products, trademarks, requisitions, regulations, guidelines, specifications and norms are free from trade mark rights of third parties.

All entries in this document have been thoroughly checked, however no guarantee for correctness can be given. Lawo AG cannot be held responsible for any misleading or incorrect information provided throughout this manual.

Lawo AG reserves the right to change specifications at any time without notice

© Lawo AG, February 2012

Document History:

	Date	Version
Basic principle for a discussion about a TCP/IP based protocol.	11.05.2001	0.1
Introduction and philosophy.	20.08.2001	0.2
Connecting and controlling crosspoints.	10.10.2001	0.3
Minor changes and completions.	19.10.2001	0.4
Login message contains ProjectID.	06.11.2001	1.0
Port number changed due to Kerberos conflicts. Login and login reply messages are also using MNOPL headers now.	21.11.2001	1.1
Description for redundant server communication and a takeover message added.	11.03.2002	1.2
Manufacturer and device IDs added. New special HLSD "UNKNOWN_SIGNAL" introduced.	26.11.2002	1.3
Streaming of error messages now included.	24.01.2003	1.4
Message head description for SA_SYSTEMMESSAGE added.	06.02.2003	1.5
Routing replies will now be sent even if the connection already exists.	13.06.2003	1.6
Device-Id NovaControl for manufacturer Lawo added.	20.11.2003	1.7
Description for double star topology, star switchover, retrieval of all system HLSDs, label retrieval, label modification and next generation system messages added.	05.01.2004	2.0
Error in the description of SA_ROUTERMESSAGE corrected.	15.01.2004	2.0.1
MicLine parameter control.	29.01.2004	2.1
DSP main level control added.	29.03.2004	2.2
DSP panning and positioning added.	15.04.2004	2.3
IO board DSP parameter controlling added.	22.06.2004	2.4
Signal mapping for ext. controllers, control of GPIO, silence detect and crossfades added.	26.01.2005	
Table of error classes and error types added.	24.03.2005	2.5
Device type GENERIC, replies for existing routings, mono mixers in outputs, IO board balance added.	28.10.2005	2.6
Control of units for stereo/surround monitoring and audio-follows-video events added.	20.12.2005	2.7
Matrix DSP board unit control added.	27.01.2006	2.8
Bugfix: MX_DSP_CHANNEL_MODULES delivers unsigned short instead of byte.	13.02.2006	2.8.1
HD/SD SDI control and mixing units parameter query included.	30.05.2006	2.9
Complete layout rework. Integration of Matrix DSP metering, AGC, gate, parametric and graphic EQ, 64 ² mixing unit, compact state query, compact state sequence, single commands for I/O board mixing units, protect level, request level and additional device types.	19.10.2006	3.0 pre
Protect and request levels omitted. Version number is now following the Nova73HD software version number.	19.06.2007	3.0
Minor layout corrections	21.06.2007	3.0.1
Matrix DSP input mixer, timed fader mute, signal condition monitor and general purpose channels (for mixing consoles) added.	02.01.2008	3.0.2
New boundaries for maximum number of matrix-DSP-channels. Client access to metering limited to a maximum amount of 5 clients. Number of concurrent silence detects increased.	07.02.2008	4.0
Table DSP Audio Follows Video included, Information Element table updated, Manufacturer and Device ID table updated, table Working with Audio-Follows-Video updated, some minor corrections.	17.03.2009	4.4

Working with signal labels, Audio follows Video some minor corrections, Protection Levels	05.06..2009	4.4.2
Server and Client Version Support in Login and Login Reply Messages, Working with Extended Device Information, Working with DSP Master Channels (for mixing consoles)	18.06.2009	4.6.0
DALLIS IP Codec section added.	07.05.2010	4.8.0
DSP Snap Isolate section added.	17.06.2010	4.10.0
Corrected some typos and copy/paste mistakes, added new 4.12 features: SDI 3G, Static Connect, PFL & AUX Send settings, changed HLSD struct entries to SignalAddressType (CSA).	13.12.2010	4.12.0
Appendix A - Mic/Line card dynamic range design limits added.	15.12.2010	4.12.0
Signal generator support, master (VCA, Surround) HLSD behaviour and mixing console access change added. Corrected some typos.	29.07.2011	4.14.0
Ported document to new Technical Documentation template, added Snapshot and Preset handling	24.01.2012	4.16.0

Contents

Document History:	3
1 Introduction	9
1.1 General Overview	9
1.2 Establishing a Connection	9
1.3 Login Sequence	9
1.4 Streams	9
1.5 Polling	9
1.6 Terminating the Session	9
2 MNOPL protocol	10
2.1 General Data Types	10
2.2 High Level Signal Definitions	10
2.3 Global Device Addresses (GDA)	11
2.4 Signal Mapping	11
2.5 Message Structure	11
2.6 Message Types	12
2.6.1 General remarks about message types and data format	12
2.6.2 PK_EVENT	13
2.6.3 PK_CHANGE	13
2.6.4 PK_QUERY	13
2.6.5 PK_QUERY_REPLY	13
2.6.6 PK_QUERY_NOREPLY	14
2.6.7 PK_ACCESS_DENIED	14
3 Detailed Description	15
3.1 Topology	15
3.1.1 Standalone Server	15
3.1.2 Redundant Servers	15
3.1.3 Dual Star Configuration	15
3.2 Connecting to a Server	15
3.2.1 Login Sequence	15
3.2.2 Connection Timing Constraints	15
4 General Messages	16
4.1 Login Message	16
4.2 Login Reply Message	16
4.3 Subscription	17
4.4 Unsubscription	17
4.5 Polling	17
4.6 Router Activity	18
4.7 Server Activity	18
5 Client Configuration in Service	19
5.1 Getting the List of Signal Definitions	19
5.2 Working with Signal Labels	19
5.3 Retrieval of Device Types	20
5.4 Retrieval of Extended Device Type Information	20
5.5 Compact State Query	22
5.6 Iterating Mapping Tables	23
6 Parameter Control	25
6.1 Microphone/Line Inputs Parameter Control	25
6.2 IO Board DSP Parameter Control	27
6.2.1 IO Board DSP Level Change Request	27
6.3 IO Board DSP Phase Inversion Request	27
6.4 IO Board DSP Mono Mix Request	27
6.5 IO Board DSP Mono Mixer Mode Settings	28
6.6 IO Board DSP Balance	28

6.7	I/O Board Mixing Matrix	28
6.7.1	Retrieving Unit Information	29
6.7.2	Configuring Mixing Unit Inputs (DEVICETYPE_SUM_IN)	30
6.7.3	Configuring Mixing Unit Outputs (DEVICETYPE_SUM_OUT)	31
6.7.4	Fading	32
6.7.5	Single Parameter Control	32
6.8	Stereo Monitoring Units (valid for mc ² HD)	35
6.9	Surround Monitoring Units	39
6.10	Silence Detect	44
6.11	DALLIS SDI HD/SD Units.....	46
6.11.1	Input and output sections	47
6.11.2	Input section	47
6.11.3	Output section	48
6.11.4	Device section	49
6.12	DALLIS SDI 3G Units.....	50
6.12.1	Output section	51
6.12.2	Device section	51
6.13	DALLIS IP Codec Units.....	55
6.13.1	IP Codec Control	55
6.13.2	IP Codec Status	56
7	Generic Control	59
7.1	Working with Audio-Follows-Video.....	59
7.2	Working with GPIO	59
7.2.1	GPI State (DEVICETYPE_GPI).....	59
7.2.2	GPO State (DEVICETYPE_GPO)	60
7.3	Setting two Signals to Stereo Mode	60
8	Working with Protections.....	61
8.1	Request Level.....	62
8.1.1	Request Level Event	62
8.1.2	Request Level Change	62
8.1.3	Querying a Request Level	62
8.1.4	Query Reply for a Request Level	62
8.1.5	Reply for Unsuccessful Request Level Query	63
8.2	Protect Level.....	63
8.2.1	Protect Level Request Message.....	63
8.2.2	Protect Level change Reply Message	64
8.2.3	Querying a Protect Level	64
8.2.4	Query Reply for a Protect Level	64
8.2.5	Reply for Unsuccessful Protect Level Query	64
8.3	Behaviour on protection violations	65
9	Matrix DSP Units.....	66
9.1	Matrix DSP Short Delay Module	66
9.2	Matrix DSP Fader Module.....	67
9.3	Matrix DSP Compressor Module.....	68
9.4	Matrix DSP Limiter Module	70
9.5	Matrix DSP Parametric EQ Module.....	71
9.5.1	Filter Slope and Type	71
9.5.2	Valid types per band.....	71
9.5.3	Slope and Quality per Type	71
9.6	Matrix DSP Graphic EQ Module	72
9.7	Matrix DSP Gate Module	73
9.8	Matrix DSP Automatic Gain Control Module	73
9.8.1	Side Chain Filter Types	74
9.9	Matrix DSP Input Mixer	74
9.10	Matrix DSP Timed Fader.....	75
9.11	Matrix DSP Signal Condition Monitor	75

9.12	Matrix DSP Channel Metering.....	76
9.13	Matrix DSP Correlation Metering.....	77
9.14	Matrix DSP Mixing Matrix.....	78
10	DSP Parameter Control (valid for mc ² series only)	80
10.1	DSP Main Level	80
10.2	DSP Channel Cut.....	81
10.3	DSP Panpot Balance	81
10.4	DSP Panpot Frontback	81
10.5	DSP Access Channel.....	81
10.6	DSP Audio Follows Video	81
10.7	DSP PFL On/Off, PFL 1 Clear, Aux Send On/Off, PEQ, PF, AF	83
11	Snapshots	84
11.1	Loading and Saving Snapshots.....	84
11.2	Using Snapshot Filters	85
11.3	Working with Snapshot Isolate	86
12	Access Channel Presets (valid for mc ² series only)	88
12.1	Querying a Channel Preset.....	88
12.2	Recalling a Channel Preset.....	88
13	Working with Routings	90
13.1	Basic Routing Messages.....	90
13.1.1	Routing Request Message	90
13.1.2	Routing Reply Message	90
13.1.3	Querying a Routing	91
13.1.4	Query Reply for a Routing	91
13.1.5	Reply for Unsuccessful Routing Query.....	91
13.2	Static Routing.....	91
13.2.1	Static Routing Request.....	91
13.2.2	Static Routing Change	92
13.2.3	Static Routing Query	92
13.2.4	Static Routing Reply	92
13.2.5	Unsuccessful Static Routing Query	92
14	Using signal generators (noise & tone)	93
14.1	Signal generators on DALLIS I/O units	93
14.2	Signal generators on Router Card 980/33.....	94
14.2.1	Automatic generation of HLSDs	94
14.2.2	Controlling the core sine generators.....	94
14.2.3	Controlling the core noise generators.....	95
15	Receiving Error Messages and Warnings	96
15.1	Errorinfo Struct	96
16	Manufacturer and Device ID Table.....	97
17	Information Element Table	98
18	Error classes and error types	99
19	Appendix A – Mic/Line card dynamic range design limits	101
19.1	Basic Condition	101
19.2	Adjustment Range.....	101

1 Introduction

The remote control protocol RemoteMNOPL is a LAN based client-server network byte order protocol to enable third party systems to control Lawo's digital mixing consoles or standalone routers. It is based on TCP/IP and implements the philosophy of categorized streams of control data for communication. The functionality described here is valid for mc² HD series and Nova 73.

1.1 General Overview

Digital mixing consoles or routers from Lawo AG are servers for third party equipment. They provide services for external systems to remotely control Lawo products. In the further description we will refer to Lawo products as the "server" and third party equipment as the "client".

1.2 Establishing a Connection

Connection establishment is similar to any other TCP/IP based client server architecture. The client is asking the server for a connection via a TCP socket with a known port number. After the connection has been accepted by the server the login sequence is the first communication flow.

1.3 Login Sequence

The client has to send a login message with an identification structure. The server is responding to this message with its own identification.

1.4 Streams

The server is using various messages and data types but only a limited number is in the interest of a client. Therefore a possibility to filter all these information elements has been created. RemoteMNOPL is defining streams as sequences of messages and data types of a specific family of information elements. The client can subscribe to these streams to receive the respective information.

1.5 Polling

During the communication session the server is sending *keepalive* messages to the client. These messages have to be answered immediately. If the client does not answer on more than 4 keepalive queries, the server will close the communication socket. A new connection has to be established by the client passing through a new login sequence.

1.6 Terminating the Session

A remote control session is terminated by closing the communication socket.

2 MNOPL protocol

The used MNOPL protocol is an event based communication protocol. Events can be sent at any time except during the start-up synchronization where the login sequence and the subscription to streams are done.

2.1 General Data Types

Data Type	Size
byte	1 octet unsigned
unsigned short	2 octets unsigned
unsigned int	4 octets unsigned
signed short	2 octets signed
signed int	4 octets signed
string[x]	sequence of x bytes
Client Signal Address Type (CSA)	unsigned int
High Level Signal Definition (HLSD)	unsigned int
Global Device Addresses (GDA)	unsigned int
head.m message types (M_NUM)	unsigned short
STATE8	byte
STATE16	unsigned short
STATE32	unsigned int
KEYS	unsigned short
DEVICETYPE	unsigned short
Extended Device Type (MCX_DEVICETYPE)	unsigned int
LEVEL	signed short
RANGE	signed short
TIME	unsigned int
STIME	unsigned short
RATIO	unsigned short
FREQ	unsigned short
QUALITY	unsigned short

The order in data types containing multiple octets is big endian (network byte order).

2.2 High Level Signal Definitions

The High Level Signal Definitions (HLSD) are special entities of four octets (unsigned int) each describing one level of the signal definition hierarchy. This is the addressing scheme for Lawo systems. Every signal in a system can be addressed by an HLSD.

bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	Signal class								Subclass								Component								Subcomponent							

The most significant bit (bit 31) is also called the 'target flag' since it can be used to determine sources and targets. If this bit is set, the signal definition points to a target otherwise to a source. The signal class is used to group signals by type. The subclass (sub signal) and component fields identify the signal mostly but not necessarily hardware oriented. The subcomponent field identifies single signals of multi-channel formats. Currently only the formats mono (subcomponent 0) and stereo (subcomponent 0 = left or 1 = right) are supported.

DSP signal HLSDs in mixing consoles use a different meaning for the subcomponent field: if the signal index which is normally represented in the component field exceeds 256 the subcomponent field will contain the high byte of the signal index.

2.3 Global Device Addresses (GDA)

The Global Device Address is used for a tree-based addressing of devices in Lawo systems. Descending from an unnamed root node the GDA tree levels are like follows:

- HD Core level
- HD Core IO card level (MADI card, ATM card, DSP card etc.)
- IO card port level (MADI ports, ATM ports)
- DALLIS IO card level (Microphone & Line interfaces, AES interfaces etc.)
- Signal index level

The tree is represented by a 32-bit number with the following structure:

bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	HD Core				HD Core IO card				IO port				DALLIS IO card				S	T	Signal index													

“S” in the table above marks the subdevice bit (which is always set to 0 for standard audio signals), “T” marks the target bit. This bit is set to 1 if the signal is a matrix destination and to 0 if it is a matrix source.

Global Device Addresses are currently used for the Extended Device Type Information telegram. Using the GDA information the client is able to determine the physical position of a signal.

2.4 Signal Mapping

Some clients are not able to use HLSD addresses for a couple of reasons. These clients can use the signal mapping feature of RemoteMNOPL. With signal mapping any addressing scheme with signal addresses of up to 31 bit in size can be mapped to values compatible with this protocol.

bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	T	External Address																														

The client can use the 32 bit frame above to send any client specific address to the server. Bit 32 is still used as the ‘target flag’ to distinguish source addresses from target addresses. In the server maps can be configured for up to 15 different clients to associate the client specific numbers to the HLSD addresses the server uses. Note: A client that uses signal mapping has to send the index of the appropriate map to the server in the login message.

Example: A client is using 10 bit addresses for sources and targets. To send a connect request source 17 to target 431 in signal mapping mode the client has to write the small addresses in unsigned int fields and set bit 31 for the target address. These unsigned int values then will be used for the RemoteMNOPL message datagram.

Source 17:

bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1

Target 431:

bit	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0	1	1	1	1

2.5 Message Structure

The message structure of a protocol message is as follows:

Message ID	unsigned short	head.m
Message Sub ID	unsigned short	head.n
Origin ID	byte	head.o
Message Type	byte	head.p
Message Length	unsigned short	head.l
[Data]	0...x octets	data

Explanation:

- Message ID (m): Defines the message type. Please refer to chapter 17 for a list of available message types.
- Message Sub ID (n): Contains further information depending on the message id. Must be set to 0 until otherwise stated in the message description.
- Origin ID (o): The unique ID for external controllers. This ID is assigned to the client by the server in the login sequence. The client has to use this ID in every message it is sending to the server.
- Message Type (p): Messages are divided into the following package types.

Request for data change	PK_EVENT
Report of a data change	PK_CHANGE
Query for data	PK_QUERY
Successful query reply	PK_QUERY_REPLY
Unsuccessful query reply	PK_QUERY_NOREPLY
Access denied	PK_ACCESS_DENIED

Note: A query will be replied only to the inquirer – even if the client did not subscribe to the corresponding message stream. Server changes are broadcasted in the system and will be transmitted to the client if the corresponding message stream is subscribed.

- Message Length (l): Number of octets of the message head (8 octets) plus the number of data octets contained.
- [Data]: The number of octets in the data field depends on the message ID and the message type. Some messages have no data.

2.6 Message Types

The RemoteMNOPL protocol contains six different types of messages:

- Event (PK_EVENT): the event message is a request for a data change in the system's database.
- Change (PK_CHANGE): a change message is the result of an action taken on the system's database. It is the reply to an event message (see above).
- Query (PK_QUERY): a query for certain parameters in the system.
- Query reply (PK_QUERY_REPLY): the query reply contains the information requested from the system by a query message.
- Unsuccessful query reply (PK_QUERY_NOREPLY): if unavailable or non-existent data has been requested the system's response will be an unsuccessful query reply.
- Access denied reply (PK_ACCESS_DENIED): Access denied due to restrictions for the client.

The following sections contain examples for every type of message possible. As a demonstration the messages will work with matrix connects.

2.6.1 General remarks about message types and data format

In most cases the data format of a Remote MNOPL message depends on its message type and there is a common concept for deducing the data format of any other message type if one data format is known. This concept for the data format is as follows:

PK_EVENT	contains one or more data elements including an address and a parameter
PK_CHANGE	same as PK_EVENT
PK_QUERY	contains one or more addresses (and no parameter)
PK_QUERY_REPLY	same as PK_EVENT
PK_QUERY_NOREPLY	same as PK_QUERY
PK_ACCESS_DENIED	same as PK_QUERY

Please note that all query and event messages support a single address only unless otherwise stated.

The following chapters give an example for a routing message.

2.6.2 PK_EVENT

The PK_EVENT message encapsulates requests for data changes in the system database. In this example the target 0x9B000200 shall be connected to the source 0x23000100.

Field	Value	Description
head.m	MX_CONNECT	Matrix routing.
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_EVENT	Request for data change.
head.l	E.g. 16	8 octets for the message head and 8 octets of data for 2 Client Signal Addresses (source and target)
Data	E.g. 0x23000100	Client Signal Address (CSA) of an audio source
	E.g. 0x9B000200	Client Signal Address (CSA) of an audio target

2.6.3 PK_CHANGE

The server would reply to the routing request message in the example above with:

Field	Value	Description
head.m	MX_CONNECT	Matrix routing.
head.n	0	Don't care.
head.o	OriginID	Server's Origin ID.
head.p	PK_CHANGE	Report of data change.
head.l	16	8 octets for the message head and 8 octets of data for 2 Client Signal Addresses (source and target).
Data	0x23000100	Client Signal Address (CSA) of an audio source.
	0x9B000200	Client Signal Address (CSA) of an audio target.

2.6.4 PK_QUERY

The source connected to a target can be queried by a MX_CONNECT message with packet type PK_QUERY.

Field	Value	Description
head.m	MX_CONNECT	Matrix routing.
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_QUERY	Query for data.
head.l	12	8 octets for the message head and 4 octets of data for a Client Signal Address (target).
Data	E.g. 0x9B000200	Client Signal Address (CSA) of the audio target.

This request will be answered with one of the following replies:

2.6.5 PK_QUERY_REPLY

If the query can be processed successfully the system responds with a PK_QUERY_REPLY message.

Field	Value	Description
head.m	MX_CONNECT	Matrix routing.
head.n	0	Don't care.
head.o	OriginID	Server's OriginID.
head.p	PK_QUERY_REPLY	Query reply.

Field	Value	Description
head.l	16	8 octets for the message head and 8 octets of data for two Client Signal Addresses (source and target).
Data	0x23000100	Client Signal Address (CSA) of the audio source.
	0x9B000200	Client Signal Address (CSA) of the audio target.

2.6.6 PK_QUERY_NOREPLY

In case a problem occurs during the processing of the query the system will inform the client with a PK_QUERY_NOREPLY message.

Field	Value	Description
head.m	MX_CONNECT	Matrix routing.
head.n	0	Don't care.
head.o	OriginID	Server's OriginID.
head.p	PK_QUERY_NOREPLY	Query unsuccessful.
head.l	12	8 octets for the message head and 4 octets of data for a Client Signal Address.
data	0x9B000200	Client Signal Address (CSA) of the audio target.

2.6.7 PK_ACCESS_DENIED

Due to configuration restrictions it is possible that the client is not allowed to access the audio target. The server will respond with a PK_ACCESS_DENIED message.

Field	Value	Description
head.m	MX_CONNECT	Matrix routing.
head.n	0	Don't care.
head.o	OriginID	Server's OriginID.
head.p	PK_ACCESS_DENIED	Access to this audio target is prohibited.
head.l	12	8 octets for the message head and 4 octets of data for a Client Signal Address.
data	0x9B000200	Client Signal Address (CSA) of the audio target.

3 Detailed Description

3.1 Topology

3.1.1 Standalone Server

In a standalone server configuration the client only has to connect to the server as described in chapter 1 to control the router and automatically keep in sync with the server.

3.1.2 Redundant Servers

For high availability reasons it is possible to have two servers in hot standby mode. A client has to connect to both servers at the same time. One of these servers is running in active mode, the other in passive or “standby” mode.

The client must send requests to both servers in parallel. The server in active mode is working the same way as a standalone server. Only the active server is sending responses to the client’s requests. The server in passive mode responds to login messages and stream subscriptions only and expects polling replies on its polling messages. All other messages to the passive server will be accepted and ignored.

When a passive server is becoming active it is sending an activity message to inform all clients that a redundancy takeover has taken place. The clients must reply to this message by sending all outstanding connection data to the server to resynchronize with the router.

3.1.3 Dual Star Configuration

In a dual star configuration each star is providing its own server. Both servers are working in parallel and audio data will be routed in each star. Therefore a client also has to connect to both servers at the same time. All connection requests have to be sent to both servers.

The client can request a switchover to the redundant star. The server of the star that has become active is sending an activity message to inform all clients that a switchover has taken place. This activity message is also sent unsolicited when e.g. a manual switchover has been initiated.

Of course each star can also be built with redundant servers. In that case the client has to connect to each star the way described above.

3.2 Connecting to a Server

The client is connecting to the server by opening a socket to TCP port 55555 at the server side. If a connection cannot be established to the server, the client must wait 10 seconds before the next try to connect again. If a connection is established successfully the client has to send the login message first.

3.2.1 Login Sequence

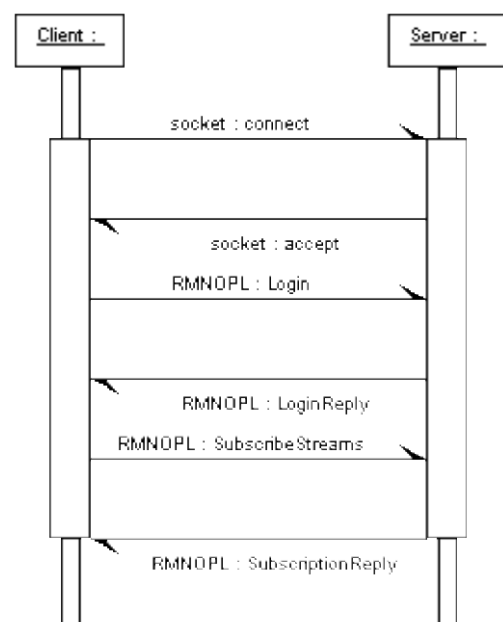
The login message contains a set of configuration information needed to initialize the server. The server will respond with a login reply message.

As the next step the client has to subscribe to the data streams needed in order to take part of the internal communication flow. The subscription request message will be answered by a subscription reply message.

By receiving the subscription reply message the client is fully accepted by the server and the necessary configuration has taken place.

3.2.2 Connection Timing Constraints

Cf. MX_HELLO, chapter 4.5.



4 General Messages

4.1 Login Message

The login message and the login reply message are special messages used for the initial connection setup. These messages contain information necessary to start the MNOPL communication. Both messages have a size of 48 octets.

Field	Type	Description
head.m	REMOTE_LOGIN	Remote Login Request.
head.n	0	Don't care.
head.o	0	Don't care.
head.p	PK_EVENT	Request for data change.
head.l	56	8 octets for the message head and 48 octets of data.
Label	string[32]	A small text describing the client type.
ManufacturerID	unsigned short	Identification of the manufacturer of the connecting client. Valid ManufacturerIDs can be found in chapter 16. You can apply for an ID at Lawo AG.
DeviceID	unsigned short	Identification of the manufacturer's device. Valid DeviceIDs can be found in chapter 16. You can apply for an ID at Lawo AG.
MappingMode	byte	0 = off; the type of signal definitions is set to HLSD. This is the normal operating mode providing the full functionality. Values between 1 and 15 choose the appropriate map for signal mapping. Signal mapping requires the configuration of client specific addresses that the server can translate to HLSDs.
ProjectID	unsigned int	Supported for legacy systems.
ControlSystemID	byte	Supported for legacy systems.
Client Major Version	byte	Version of the documentation the client complies with.
Client Minor Version	byte	
Reserved	string[4]	Reserved for future use.

After the login message has been sent, the server is reading the appropriate configuration data for serving the client. The client must set a timeout on this process not shorter than 10 seconds.

4.2 Login Reply Message

Field	Type	Description
head.m	REMOTE_LOGIN	Remote Login Request
head.n	0	Don't care.
head.o	0	Don't care.
head.p	PK_CHANGE	Report of data change.
head.l	56	8 octets for the message head and 48 octets of data.
Label	string[32]	A small text describing the server type.
ManufacturerID	unsigned short	0, if the system is a server of Lawo AG.
DeviceID	unsigned short	Identification of the manufacturer's device.
DeviceNumber	byte	Needed to distinguish multiple servers.
OriginID	byte	The OriginID that has to be used by the client in further communication (see note below).
Server State	byte	Describes whether the server is in STATE_ACTIVE or in STATE_PASSIVE.

Field	Type	Description
Router State	byte	Describes whether the router is in STATE_ACTIVE or in STATE_PASSIVE.
Server Major Version	byte	Version of the MNOPL documentation the server complies with.
Server Minor Version	byte	
Reserved	string[6]	Reserved for future use.

Note: The OriginID sent by the server in the login reply must be used by the client in the head.o field of every message it is sending to the server later. The server will use an origin ID in the range of 0 to 31 depending on the configuration. The default server origin ID is 0.

4.3 Subscription

Note: From now on the message descriptions will be less detailed. The table headers state the RemoteMNOPL message ID (head.m) and the parameters for all message types are explained in the table body. For more detailed information about the basic message structure please refer to chapters 2.5 and 2.6. For constant values (which are mostly mentioned CAPITALIZED) please refer to chapter 17.

OPEN_STREAM		
Message Type	Parameter	Comment
PK_EVENT	[Sequence of] M_NUM	This subscription message is opening the stream of message replies in which head.m contains M_NUM. You can subscribe to more than one stream by sending this message with multiple M_NUMs in the data field. The length of the message (head.l) must be set to the appropriate value.
PK_CHANGE	[Sequence of] M_NUM	The server will respond with a change message where the data field contains all the streams that were successfully registered for the client. The length of the message (head.l) is set to the appropriate value.
PK_QUERY	None	Not implemented.
PK_QUERY_REPLY	None	Not implemented.

4.4 Unsubscription

CLOSE_STREAM		
Message Type	Parameter	Comment
PK_EVENT	[Sequence of] M_NUM	This unsubscription request message is closing the stream of message replies in which head.m contains M_NUM. You can close more than one stream by sending this message with multiple M_NUMs in the data field.
PK_CHANGE	[Sequence of] M_NUM	The server will respond with a change message where the data field contains all the streams that were successfully closed by the server. The length of the message (head.l) is set to the appropriate value.
PK_QUERY	None	Not implemented.
PK_QUERY_REPLY	None	Not implemented.

4.5 Polling

The server will supervise the availability of the client by sending polling messages frequently. Server polling has to be answered at once sending a polling response back to the server.

Note: If neither a polling message nor any other message was sent by the server for longer than 30 seconds the client should close the socket and try to connect to the server again.

MX_HELLO		
Message Type	Parameter	Comment
PK_EVENT	None	The client has to respond with the same message (PK_EVENT) containing its own OriginID.
PK_CHANGE	None	Not implemented.
PK_QUERY	None	Not implemented.
PK_QUERY_REPLY	None	Not implemented.

4.6 Router Activity

ER_BECOME_ACTIVE_ROUTER		
Message Type	Parameter	Comment
PK_EVENT	None	In configurations with two redundant routers ('dual star' configuration) only audio data of one routing facility is used at a time. The client can switch over to the redundant system by sending a router activity request to the server of the standby system. Initially the client receives the router state information in the login reply message where the field 'router state' contains the value STATE_ACTIVE for the active system or STATE_PASSIVE for the redundant system.
PK_CHANGE	None	If the server has set the routing facility as the active system it is sending a router activity reply message to all connected clients.
PK_QUERY	None	Not implemented.
PK_QUERY_REPLY	None	Not implemented.

4.7 Server Activity

MX_SERVER_ACTIVE						
Message Type	Parameter	Comment				
PK_EVENT	None	The client is able to switch the standby server to active mode by sending the server activity change request.				
PK_CHANGE	None	The server will reply with an appropriate server activity change message after having switched to active mode. The second server which is switching from active to standby mode will block server activity change requests for the next 10 seconds.				
PK_QUERY	None	The client may query the current server and router state using the MX_SERVER_ACTIVE query.				
PK_QUERY_REPLY	<pre>struct { byte routerState; byte serverState; }</pre>	<div>If the server activity is queried the system's reply contains two bytes reporting the router state and the server state.</div> <table><tr><td>routerState</td><td>Current router state, valid states are STATE_ACTIVE and STATE_PASSIVE.</td></tr><tr><td>serverState</td><td>Current server state, valid states are STATE_ACTIVE and STATE_PASSIVE.</td></tr></table>	routerState	Current router state, valid states are STATE_ACTIVE and STATE_PASSIVE.	serverState	Current server state, valid states are STATE_ACTIVE and STATE_PASSIVE.
routerState	Current router state, valid states are STATE_ACTIVE and STATE_PASSIVE.					
serverState	Current server state, valid states are STATE_ACTIVE and STATE_PASSIVE.					

5 Client Configuration in Service

A client can request the router's configuration data dynamically at runtime by querying the complete list of high level signal definitions and the labeling information for each CSA received.

5.1 Getting the List of Signal Definitions

ER_SYSTEM_HLSDS		
Message Type	Parameter	Comment
PK_EVENT	None	Not implemented.
PK_CHANGE	None	Not implemented.
PK_QUERY	None	The client can request the complete list of signal definitions available in the system by sending the ER_SYSTEM_HLSD query.
PK_QUERY_REPLY	[Sequence of] CSA	The reply to the query is a complete list of signal definitions available in the mapping table selected via the login message.

Note: The list of CSAs contains signal definitions for all sources and targets available to the client. Due to the nature of CSAs, sources can be distinguished from targets by looking at the most significant bit (the 'target flag'). If the most significant bit is set the signal is a target.

If the number of CSAs in the system exceeds 15000 the query reply will be segmented. The segment index will be counted backwards in the head.n field. E.g. if we have 16000 CSAs, two messages will be sent: segment 1 (head.n = 1) containing 15000 CSAs and segment 2 (head.n = 0) containing 1000 CSAs.

5.2 Working with Signal Labels

Each signal can have up to five labels: LABEL_SIGNALNAME, LABEL_STARTUP, LABEL_USER, LABEL_INHERITED and LABEL_GROUP. The label selection has to be defined in the head.n field of the message by using the appropriate value according to the following table:

ER_LABEL	0xF2A6
LABEL_SIGNALNAME	0
LABEL_STARTUP	1
LABEL_USER	2
LABEL_INHERITED	3
LABEL_GROUP	4

Labels always have a fixed size of 8 octets. A label does not necessarily contain a NULL byte at the end. Therefore it cannot be seen as a string at a "C" point of view.

Any label can be changed by sending a new label value.

It is strongly recommended to change only the user label by using LABEL_USER the head.n field of the message.

Please note that labels of DSP master channels cannot be changed.

ER_LABEL		
Message Type	Parameter	Comment
PK_EVENT	struct { CSA address; byte label[8]; }	Request to change a label.
PK_CHANGE	struct { CSA address; byte label[8]; }	Automatically generated message when a label has been changed.

ER_LABEL		
Message Type	Parameter	Comment
PK_QUERY	[Sequence of] CSA	The client can ask for signal labels with the ER_LABEL query. The label that shall be included in the response has to be defined by setting the appropriate index in the head.n field.
PK_QUERY_REPLY	<pre>struct { CSA address; byte label[8]; }</pre>	

5.3 Retrieval of Device Types

The physical device that hosts the signal identified by a CSA can have special parameters to be controlled. To use the right parameter set it is first necessary to identify the type of a device by sending the following query.

IO_DEVICE_TYPE		
Message Type	Parameter	Comment
PK_EVENT	None	Not implemented.
PK_CHANGE	None	Not implemented.
PK_QUERY	[Sequence of] CSA	The parameter query is referenced by the CSA of the audio signal.
PK_QUERY_REPLY	<pre>struct { CSA address; DEVICETYPE type; }</pre>	The returned DEVICETYPE value specifies the type of the signal. For a table of available device types please refer to chapter 17.

5.4 Retrieval of Extended Device Type Information

In special appliances the simple device information as described in chapter 5.3 provides not enough information. Therefore, the Extended Device Type offers a more detailed view including the physical device position and the hardware device types of all parent devices.

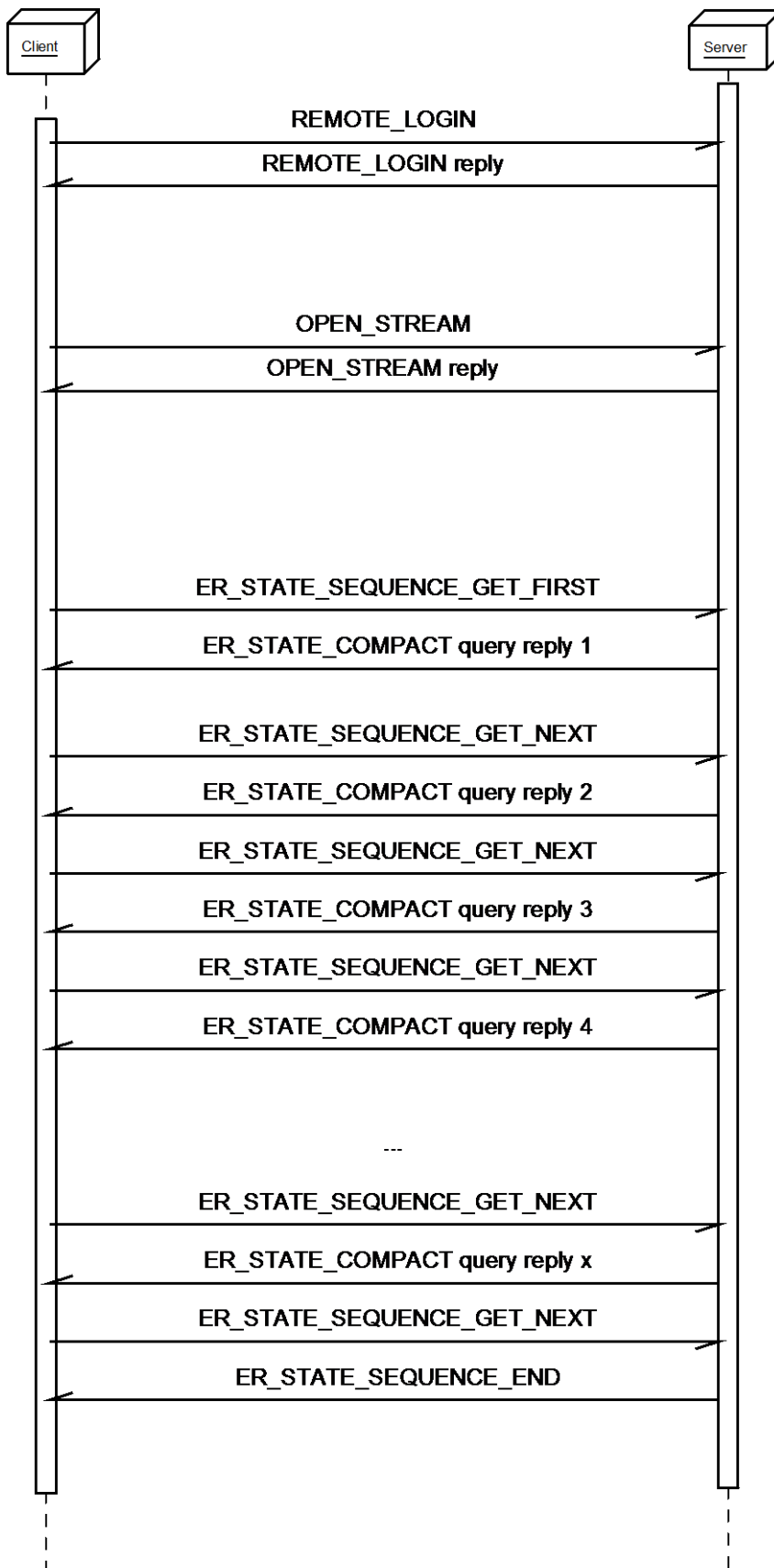
IO_EXTENDED_DEVICE_TYPE		
Message Type	Parameter	Comment
PK_EVENT	None	Not implemented.
PK_CHANGE	None	Not implemented.
PK_QUERY	[Sequence of] CSA	The parameter query is referenced by the CSA of the audio signal.

IO_EXTENDED_DEVICE_TYPE								
Message Type	Parameter	Comment						
PK_QUERY_REPLY	<pre>struct { CSA address; DEVICETYPE type; unsigned int gda; unsigned int devicetype_box; unsigned int devicetype_slot; unsigned int devicetype_port; unsigned int devicetype_dallisslot; string[42] reserved; }</pre>	<p>The query reply for an Extended Device Type Information query also contains the reply for a standard Device Type query in the first 6 data octets.</p>						
		<table><tr><td>address</td><td>Signal CSA.</td></tr><tr><td>type</td><td>Device Type (see 5.3)</td></tr><tr><td>gda</td><td>Physical device position (see 2.3).</td></tr></table>	address	Signal CSA.	type	Device Type (see 5.3)	gda	Physical device position (see 2.3).
		address	Signal CSA.					
type	Device Type (see 5.3)							
gda	Physical device position (see 2.3).							
<p>The following elements device-type_box, _slot, _port and _dallisslot can be used to determine the device types of the signal's parent devices. Please note that the values of these elements are different from DEVICE-TYPE values.</p> <p>Please refer to chapter 17 for more details.</p>								

5.5 Compact State Query

ER_COMPACT_STATE																
Message Type	Parameter	Comment														
PK_EVENT	None	Not implemented.														
PK_CHANGE	None	Not implemented.														
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.														
PK_QUERY_REPLY	<pre>struct { CSA address; CSA conn_source; byte lbl_user[8]; byte lbl_group[8]; byte lbl_inhrt[8]; DEVICETYPE type; byte protect_level; }</pre>	<table><tr><td>address</td><td>The CSA of the queried signal.</td></tr><tr><td>conn_source</td><td>If the queried signal is a source - always NO_SIGNAL (0xFFFFFFFF). If the queried signal is a target: A source CSA If a source is connected to the target. NO_SIGNAL (0xFFFFFFFF) If no source is connected to the target. UNKNOWN_SIGNAL (0xFFFEFFFF) If a source is connected to a target which is unknown to the client.</td></tr><tr><td>lbl_user</td><td>The LABEL_USER of the queried signal.</td></tr><tr><td>lbl_group</td><td>The LABEL_GROUP of the queried signal.</td></tr><tr><td>lbl_inhrt</td><td>If the queried signal is a source the LABEL_INHERITED is always its own LABEL_USER. If the queried signal is a target: LABEL_USER of a source If a source is connected to the target. Its own LABEL_USER If no source is connected to the target.</td></tr><tr><td>type</td><td>The device type of the queried signal.</td></tr><tr><td>protect_level</td><td>The protect level for routing modifications. If protection is not supported for the queried signal, this field will contain PROTECT_NO_SUCCESS (0xFF).</td></tr></table>	address	The CSA of the queried signal.	conn_source	If the queried signal is a source - always NO_SIGNAL (0xFFFFFFFF). If the queried signal is a target: A source CSA If a source is connected to the target. NO_SIGNAL (0xFFFFFFFF) If no source is connected to the target. UNKNOWN_SIGNAL (0xFFFEFFFF) If a source is connected to a target which is unknown to the client.	lbl_user	The LABEL_USER of the queried signal.	lbl_group	The LABEL_GROUP of the queried signal.	lbl_inhrt	If the queried signal is a source the LABEL_INHERITED is always its own LABEL_USER. If the queried signal is a target: LABEL_USER of a source If a source is connected to the target. Its own LABEL_USER If no source is connected to the target.	type	The device type of the queried signal.	protect_level	The protect level for routing modifications. If protection is not supported for the queried signal, this field will contain PROTECT_NO_SUCCESS (0xFF).
address	The CSA of the queried signal.															
conn_source	If the queried signal is a source - always NO_SIGNAL (0xFFFFFFFF). If the queried signal is a target: A source CSA If a source is connected to the target. NO_SIGNAL (0xFFFFFFFF) If no source is connected to the target. UNKNOWN_SIGNAL (0xFFFEFFFF) If a source is connected to a target which is unknown to the client.															
lbl_user	The LABEL_USER of the queried signal.															
lbl_group	The LABEL_GROUP of the queried signal.															
lbl_inhrt	If the queried signal is a source the LABEL_INHERITED is always its own LABEL_USER. If the queried signal is a target: LABEL_USER of a source If a source is connected to the target. Its own LABEL_USER If no source is connected to the target.															
type	The device type of the queried signal.															
protect_level	The protect level for routing modifications. If protection is not supported for the queried signal, this field will contain PROTECT_NO_SUCCESS (0xFF).															

5.6 Iterating Mapping Tables



ER_STATE_SEQUENCE_GET_FIRST		
<i>Message Type</i>	<i>Parameter</i>	<i>Comment</i>
PK_EVENT	None	The server will set its sequence pointer to the first mapping table entry, irrespective of the current position of the sequence pointer. The server will reply to ER_STATE_SEQUENCE_GET_FIRST with one of the following messages: ER_COMPACT_STATE - Query Reply of the first signal in the mapping table. ER_STATE_SEQUENCE_END - If the mapping table is empty.
PK_CHANGE	None	Not implemented.
PK_QUERY	None	Not implemented.
PK_QUERY_REPLY	None	Not implemented.

ER_STATE_SEQUENCE_GET_NEXT		
<i>Message Type</i>	<i>Parameter</i>	<i>Comment</i>
PK_EVENT	None	The server will set the sequence pointer to the next mapping table entry, if the sequence pointer doesn't point to the last mapping table entry. If the sequence pointer points to the last mapping table entry, it will not be set to the first entry of the mapping table. The server will reply to ER_STATE_SEQUENCE_GET_NEXT with one of the following messages: ER_COMPACT_STATE - Query Reply of the next signal in the mapping table. ER_STATE_SEQUENCE_END - If the mapping table is empty.
PK_CHANGE	None	Not implemented.
PK_QUERY	None	Not implemented.
PK_QUERY_REPLY	None	Not implemented.

ER_STATE_SEQUENCE_END		
<i>Message Type</i>	<i>Parameter</i>	<i>Comment</i>
PK_EVENT	None	Not implemented.
PK_CHANGE	None	Not implemented.
PK_QUERY	None	Not implemented.
PK_QUERY_REPLY	None	Table pointer reached the end of the compact state query sequence.

6 Parameter Control

6.1 Microphone/Line Inputs Parameter Control

IO_MICLINE_BLOCK			
Message Type	Parameter	Comment	
PK_EVENT	<pre>struct { CSA address; byte mode; LEVEL amp; byte phantom; byte lowcutmode; byte pad; }</pre>	address	The CSA of the mic/line input.
		mode	Set to MODE_MIC or MODE_LINE. Please note that LINE only IO cards do not support switching to MODE_MIC. Nevertheless the switching will be tried – so there will be a success message first. After a short moment the actual hardware will deny the request by sending another PK_CHANGE with the MODE_LINE value.
		amp	Contains the amplification in dB in steps of 1 dB with a resolution of 1/32 dB (e.g. 32 = 1 dB, 64 = 2 dB). Please note that this 1 dB step restriction applies to Remote MNOPL clients only and other clients might send different values. The general range is -20 to +70 dB, which yields LEVEL values of -640 to +2240. Depending on the hardware type, the hardware mode (MIC, LINE), the system headroom and the system reference level, the valid range might be limited. In such a case the full control range is not available. If you try to set a value exceeding the range the hardware is capable of, the hardware will deny the request and send a PK_CHANGE message containing the corrected value which has been set.
		phantom	Set this field to ON if you want to switch phantom power on for microphone inputs. Otherwise set it to OFF. Phantom power switch requests will be ignored for line inputs.
		lowcutmode	Switches the low cut filter. Defined values are OFF, LOWCUT_40HZ, LOWCUT_80HZ, LOWCUT_140HZ.
		pad	Switches pad ON or OFF. Pad will not be available for line inputs.

IO_MICLINE_BLOCK		
Message Type	Parameter	Comment
PK_CHANGE	<pre>struct { CSA address; byte mode; LEVEL amp; byte phantom; byte lowcutmode; byte pad; }</pre>	Contains the values that have been set by the hardware.
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.
PK_QUERY_REPLY	<pre>struct { CSA address; byte mode; LEVEL amp; byte phantom; byte lowcutmode; byte pad; }</pre>	See PK_EVENT above.

6.2 IO Board DSP Parameter Control

6.2.1 IO Board DSP Level Change Request

The digital amplification level on I/O boards can be controlled by using the following RemoteMNOPL messages.

IO_BOARD_DSP_GAIN			
Message Type	Parameter	Comment	
PK_EVENT	<pre>struct { CSA address; LEVEL amp; }</pre>	address	The CSA of the input or output signal.
		amp	Contains the amplification in dB in steps of 1/32. The range is <i>-128 to +15 dB</i> . This yields LEVEL values of <i>-4096 to +480</i> .
PK_CHANGE	<pre>struct { CSA address; LEVEL amp; }</pre>	Contains the values that have been set by the hardware.	
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.	
PK_QUERY_REPLY	<pre>struct { CSA address; LEVEL amp; }</pre>	See PK_EVENT above.	

6.3 IO Board DSP Phase Inversion Request

Phase inversion with DSPs on I/O boards is controlled similar to the IO board DSP gain by using IO_BOARD_DSP_INV_PHASE in the head.m field.

The following structure can be used as the parameter set.

```
struct
{
    CSA address;
    byte inversion;
}
```

The parameter "inversion" contains ON or OFF.

6.4 IO Board DSP Mono Mix Request

Mono mix for matrix outputs on I/O boards is controlled similar to the IO board DSP gain by using IO_BOARD_DSP_OUT_MONOMIX in the head.m field.

The following structure can be used as the parameter set.

```
struct
{
    CSA address;
    byte monomix;
}
```

The parameter "monomix" contains ON or OFF.

6.5 IO Board DSP Mono Mixer Mode Settings

The configuration of mono mixers for matrix outputs on I/O boards is controlled similar to the above by using `IO_BOARD_DSP_OUT_MONOMODE` in the `head.m` field. A mono mixer is always using two physical outputs with sequent resources. These resources depend on the configuration of the matrix and are used in a project specific way.

The following structure can be used as the parameter set.

```
struct
{
    CSA    address;
    byte   mixermode;
}
```

The parameter "mixermode" can contain one of the following modes for the mono mixer:

- | | |
|---|-------------------------------|
| 0 | Mono mix |
| 1 | Left channel to both outputs |
| 2 | Right channel to both outputs |
| 3 | Left Right Swap |

6.6 IO Board DSP Balance

The control of the balance parameter of stereo inputs and outputs on I/O boards is controlled similar to the above by using `IO_BOARD_DSP_BALANCE` in the `head.m` field. Balance is always used with two physical inputs or outputs with sequent resources. These resources depend on the configuration of the matrix and are used in a project specific way.

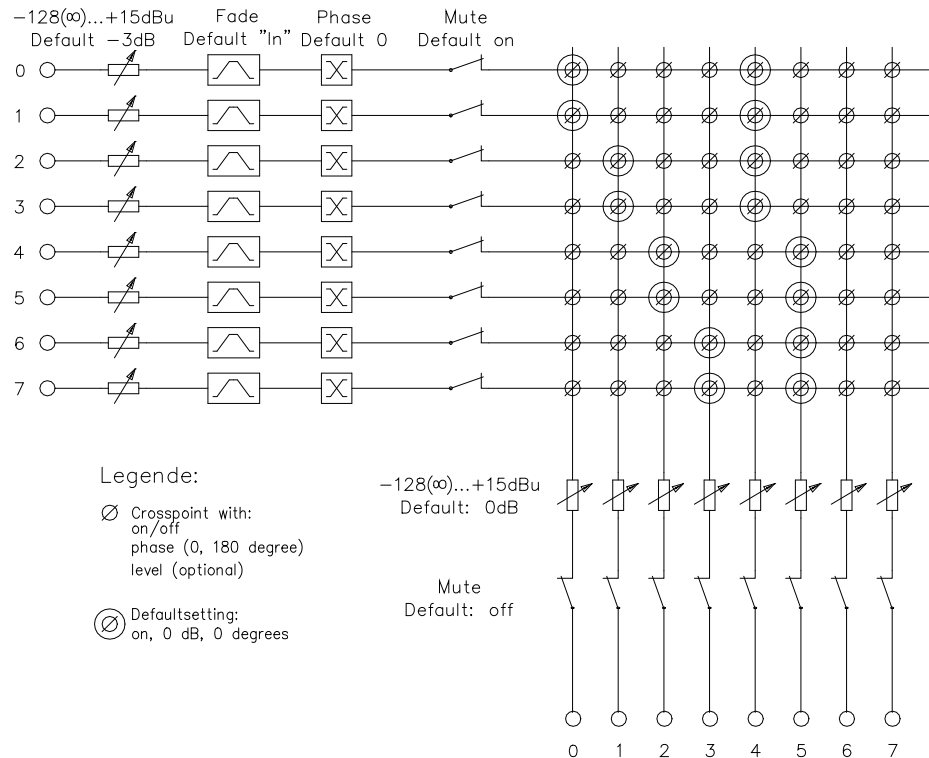
The following structure can be used as the parameter set.

```
struct
{
    CSA    address;
    RANGE  balance;
}
```

The range of the balance value is -20...+20.

6.7 I/O Board Mixing Matrix

A matrix system can have multiple small mixing units that can be inserted into a signal path. With these mixing units it is possible to implement signal fading or small mixing entities. The structure of these devices is shown in the following diagram:



The inputs and outputs of the mixing units are internal targets and sources of the matrix and identified by CSAs. Therefore the units can be inserted into a signal path by standard connect procedures.

To identify the sources and targets of a mixing unit the IO_DEVICE_TYPE query message can be used. The query result will be DEVICE_TYPE_SUM_IN for the inputs of the mixing units (target CSAs from the matrix point of view) and DEVICE_TYPE_SUM_OUT for the outputs (source CSAs for the matrix).

There are two types of mixing units available: 8x8 as shown in the diagram and 4x4 with the same functionality but smaller in size. The unit size can be queried with the ER_UNIT_INFO message. The ER_UNIT_INFO query delivers the dimension of the mixing unit (4 or 8), the ID of the unit in the matrix system and the ID of the channel within the mixing unit per CSA.

6.7.1 Retrieving Unit Information

ER_UNIT_INFO										
Message Type	Parameter	Comment								
PK_EVENT	None	Not implemented.								
PK_CHANGE	None	Not implemented.								
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.								
PK_QUERY_REPLY	<pre>struct { CSA address; byte dimension; byte unitID; byte channelID; }</pre>	<table><tr><td>address</td><td>CSA of the input or output of a mixing unit.</td></tr><tr><td>dimension</td><td>Dimension of the mixing unit (4 or 8).</td></tr><tr><td>unitID</td><td>Unit ID of the mixing unit.</td></tr><tr><td>channelID</td><td>Channel ID of the mixing unit.</td></tr></table>	address	CSA of the input or output of a mixing unit.	dimension	Dimension of the mixing unit (4 or 8).	unitID	Unit ID of the mixing unit.	channelID	Channel ID of the mixing unit.
address	CSA of the input or output of a mixing unit.									
dimension	Dimension of the mixing unit (4 or 8).									
unitID	Unit ID of the mixing unit.									
channelID	Channel ID of the mixing unit.									

6.7.2 Configuring Mixing Unit Inputs (DEVICETYPE_SUM_IN)

IO_BOARD_SUM_IN_CONFIG																
Message Type	Parameter	Comment														
PK_EVENT	<pre>struct { CSA address; byte mute; byte invPhase; LEVEL gain; STIME fadeInTime; STIME fadeInWaitTime; STIME fadeOutTime; }</pre>															
		<table><tr><td>address</td><td>CSA of the mixing unit input.</td></tr><tr><td>mute</td><td>Input mute (ON/OFF).</td></tr><tr><td>invPhase</td><td>Input phase inversion (ON/OFF).</td></tr><tr><td>gain</td><td>Contains the input amplification in dB in steps of 1/32. The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.</td></tr><tr><td>fadeInTime</td><td>Fade-in time in milliseconds. Recommended time settings are 20 to 65535 ms.</td></tr><tr><td>fadeInWaitTime</td><td>Fade-in-wait time in milliseconds (fade in starts after [fadeWaitTime] ms). Recommended time settings are 20 to 65535 ms.</td></tr><tr><td>fadeOutTime</td><td>Fade-out time in milliseconds. Recommended time settings are 20 to 65535 ms.</td></tr></table>	address	CSA of the mixing unit input.	mute	Input mute (ON/OFF).	invPhase	Input phase inversion (ON/OFF).	gain	Contains the input amplification in dB in steps of 1/32. The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.	fadeInTime	Fade-in time in milliseconds. Recommended time settings are 20 to 65535 ms.	fadeInWaitTime	Fade-in-wait time in milliseconds (fade in starts after [fadeWaitTime] ms). Recommended time settings are 20 to 65535 ms.	fadeOutTime	Fade-out time in milliseconds. Recommended time settings are 20 to 65535 ms.
		address	CSA of the mixing unit input.													
		mute	Input mute (ON/OFF).													
		invPhase	Input phase inversion (ON/OFF).													
		gain	Contains the input amplification in dB in steps of 1/32. The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.													
		fadeInTime	Fade-in time in milliseconds. Recommended time settings are 20 to 65535 ms.													
		fadeInWaitTime	Fade-in-wait time in milliseconds (fade in starts after [fadeWaitTime] ms). Recommended time settings are 20 to 65535 ms.													
fadeOutTime	Fade-out time in milliseconds. Recommended time settings are 20 to 65535 ms.															
PK_CHANGE	<pre>struct { CSA address; byte mute; byte invPhase; LEVEL gain; STIME fadeInTime; STIME fadeInWaitTime; STIME fadeOutTime; }</pre>	Contains the values that have been set by the hardware.														
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.														
PK_QUERY_REPLY	<pre>struct { CSA address; byte mute; byte invPhase; LEVEL gain; STIME fadeInTime; STIME fadeInWaitTime; STIME fadeOutTime; }</pre>	See PK_EVENT above.														

6.7.3 Configuring Mixing Unit Outputs (DEVICETYPE_SUM_OUT)

IO_BOARD_SUM_OUT_CONFIG			
Message Type	Parameter	Comment	
PK_EVENT	<pre>struct { CSA address; byte mute; byte invPhase; LEVEL gain; byte XPSet[8]; byte invXPPhase[8]; LEVEL XPGain[8]; }</pre>	address	CSA of the mixing unit output.
		mute	Output mute (ON/OFF).
		invPhase	Output phase inversion (ON/OFF).
		gain	Contains the output amplification in dB in steps of 1/32. The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.
		XPSet[8]	Set to ON if the corresponding input shall be contained in the output, otherwise OFF.
		invXPPhase[8]	Set to ON if the corresponding input shall be contained in the output with inverted phase, otherwise OFF.
		XPGain[8]	Contains the amplification at the crosspoint in dB in steps of 1/32. The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.
PK_CHANGE	<pre>struct { CSA address; byte mute; byte invPhase; LEVEL gain; byte XPSet[8]; byte invXPPhase[8]; LEVEL XPGain[8]; }</pre>	Contains the values that have been set by the hardware.	
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.	
PK_QUERY_REPLY	<pre>struct { CSA address; byte mute; byte invPhase; LEVEL gain; byte XPSet[8]; byte invXPPhase[8]; LEVEL XPGain[8]; }</pre>	See PK_EVENT above.	

With this message it is possible to configure a complete output as shown in the block diagram above. To configure the whole mixing unit it is necessary to send four/eight messages, one for each output of the unit.

6.7.4 Fading

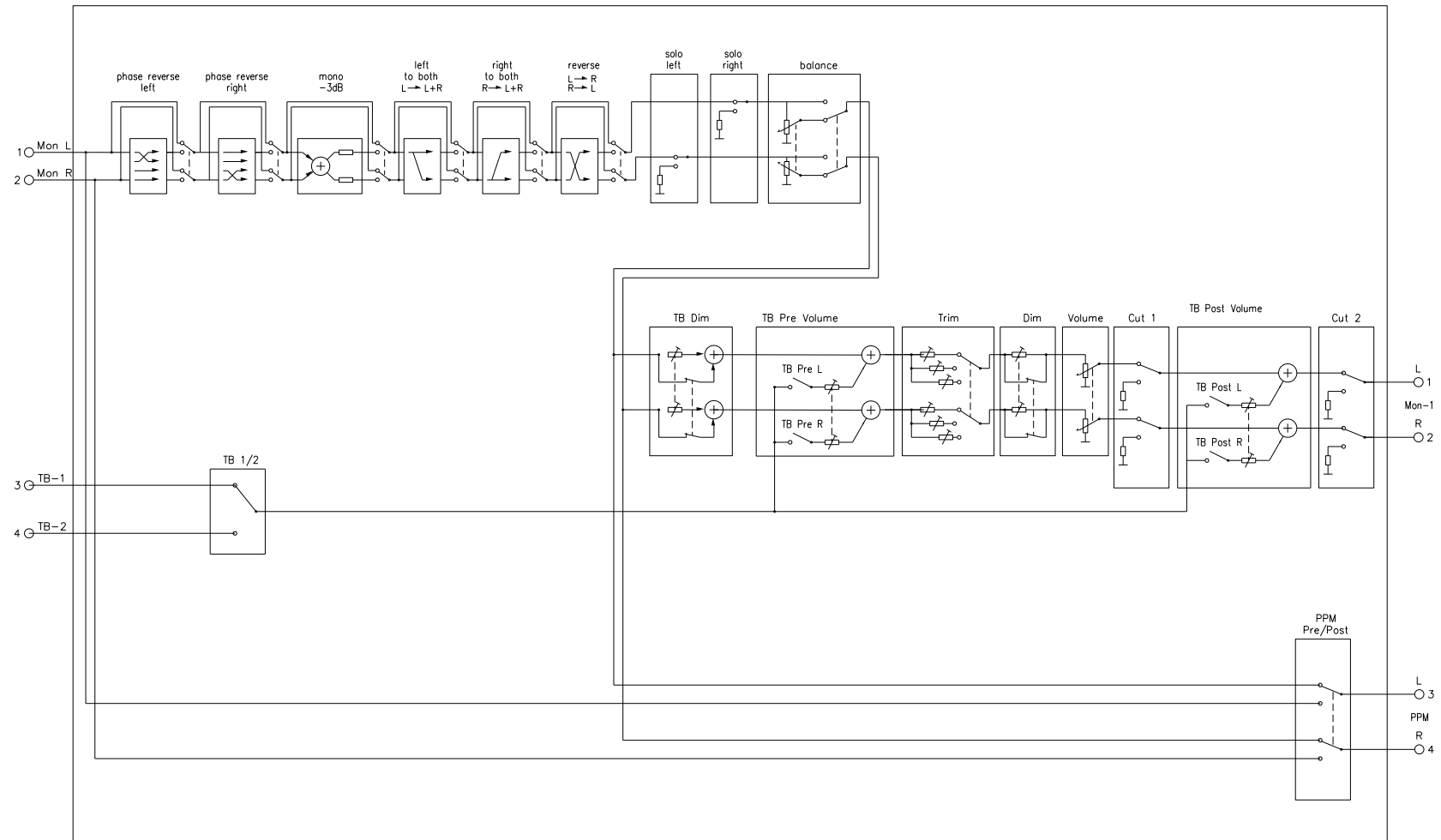
To use the fading function of the mixing units it is necessary to configure the inputs and outputs appropriately. For example a crossfade in 100ms can be done by configuring the FadeInTime and the FadeOutTime of two inputs to 100ms and the FadeWaitTime to 0ms each. An output is mixing both channels together. The fading procedure itself is started with the following message:

IO_BOARD_SUM_IN_FADE						
Message Type	Parameter	Comment				
PK_EVENT	<pre>struct { CSA address; byte fadingtype; }</pre>	With this message it is possible to start fade procedures. Every fade procedure is identified by the CSA of the unit's input channel plus the procedure type (i.e. "fade in" or "fade out").				
		<table><tr><td>address</td><td>CSA of the mixing input.</td></tr><tr><td>fadingtype</td><td>Specifies whether the signal should be faded in (FADE_IN) or faded out (FADE_OUT).</td></tr></table>	address	CSA of the mixing input.	fadingtype	Specifies whether the signal should be faded in (FADE_IN) or faded out (FADE_OUT).
		address	CSA of the mixing input.			
fadingtype	Specifies whether the signal should be faded in (FADE_IN) or faded out (FADE_OUT).					
PK_CHANGE	<pre>struct { CSA address; byte fadingtype; }</pre>	<p>Sending this PK_CHANGE the server confirms a fade request.</p> <p>Note that receiving this message does not mean that the fade has already been carried out. If a wait time of e.g. 1 second is set, the change will probably arrive before the fade starts.</p>				
PK_QUERY	None	Not implemented.				
PK_QUERY_REPLY	None	Not implemented.				

6.7.5 Single Parameter Control

head.m	Parameter Set	Comments
IO_BOARD_SUM_IN_GAIN IO_BOARD_SUM_OUT_GAIN	<pre>struct { CSA address; LEVEL gain; }</pre>	<p>The parameter "gain" contains the amplification in dB in steps of 1/32. The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.</p>
IO_BOARD_SUM_IN_MUTE IO_BOARD_SUM_OUT_MUTE	<pre>struct { CSA address; byte mute; }</pre>	<p>The parameter "mute" contains ON or OFF depending on whether the signal shall be muted or not.</p>
IO_BOARD_SUM_IN_PHASE IO_BOARD_SUM_OUT_PHASE	<pre>struct { CSA address; byte invPhase; }</pre>	<p>The parameter "invPhase" contains ON or OFF depending on whether the signal shall be phase inverted or not.</p>

head.m	Parameter Set	Comments
IO_BOARD_SUM_IN_FADE_IN_TIME	<pre>struct { CSA address; STIME fadeInTime; }</pre>	<p>The fadeInTime contains the fade-in time in milliseconds.</p> <p>Recommend settings are 20 to 65535 ms.</p>
IO_BOARD_SUM_IN_FADE_IN_WAIT_TIME	<pre>struct { CSA address; STIME fadeInWaitTime; }</pre>	<p>The fadeInWaitTime contains the fade-in-wait time in milliseconds.</p> <p>Recommend settings are 20 to 65535 ms.</p>
IO_BOARD_SUM_IN_FADE_OUT_TIME	<pre>struct { CSA address; STIME fadeOutTime; }</pre>	<p>The fadeOutTime contains the fade-out time in milliseconds.</p> <p>Recommend settings are 20 to 65535 ms.</p>
IO_BOARD_SUM_OUT_XPOINT_SET	<pre>struct { CSA address; byte XPSet; }</pre>	<p>The index of the mixing unit input for the crosspoint setting is determined in the head.n field in the range of 0 to 3 or 0 to 7, depending on the dimension of the mixing unit (4 or 8).</p> <p>The parameter "XPSet" contains ON or OFF depending on whether the given input shall be mixed to the output or not.</p>
IO_BOARD_SUM_OUT_XPOINT_PHASE	<pre>struct { CSA address; byte invXPPhase; }</pre>	<p>The index of the mixing unit input is determined in the head.n field in the range of 0 to 3 or 0 to 7, depending on the dimension of the mixing unit (4 or 8).</p> <p>The parameter "invXPPhase" contains ON or OFF depending on whether the corresponding signal shall be phase inverted in the crosspoint or not.</p>
IO_BOARD_SUM_OUT_XPOINT_GAIN	<pre>struct { CSA address; LEVEL XPGain; }</pre>	<p>The index of the mixing unit input is determined in the head.n field in the range of 0 to 3 or 0 to 7, depending on the dimension of the mixing unit (4 or 8).</p> <p>The parameter "XPGain" contains the amplification at the crosspoint in dB in steps of 1/32.</p> <p>The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.</p>



6.8 Stereo Monitoring Units (valid for mc² HD)

A system can include multiple stereo monitoring units. With these stereo monitoring units it is possible to realize a comfortable stereo monitoring. The structure of these devices is shown in the diagram.

The inputs and outputs of the stereo monitoring unit are internal targets and sources of the matrix and identified by CSAs. Therefore the units can be inserted into a signal path by standard routing procedures.

Please note that the maximum output level is a result of a computation including all parameters of the levelling chain (e.g. Talkback Dimlevel, Trimlevels, etc.) and is altogether limited to +15 dB. The same procedure applies to the minimum output level (limit: -128 dB). The “input parameters” for these computations, i.e. the levelling chain parameters, are not limited in range, i.e. the full (signed) 16-bit range of the LEVEL type may be used.

IO_BOARD_STEREO_MON_VOLUME						
Message Type	Parameter	Comment				
PK_EVENT	<pre>struct { CSA address; LEVEL volume; }</pre>	<table><tr><td>address</td><td>CSA of the monitoring input or output signal.</td></tr><tr><td>volume</td><td>Contains the amplification in dB in steps of 1/32. The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.</td></tr></table>	address	CSA of the monitoring input or output signal.	volume	Contains the amplification in dB in steps of 1/32. The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.
		address	CSA of the monitoring input or output signal.			
volume	Contains the amplification in dB in steps of 1/32. The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.					
PK_CHANGE	<pre>struct { CSA address; LEVEL volume; }</pre>	Contains the values that have been set by the hardware.				
PK_QUERY	None	Not implemented.				
PK_QUERY_REPLY	<pre>struct { CSA address; LEVEL volume; }</pre>	See PK_EVENT above.				

All further parameters are controlled similar to the stereo monitoring volume by using the appropriate message type in the head.m field and its control structure. All parameters are written down in the following table:

head.m	Parameter Set	Comments
IO_BOARD_STEREO_DIMLEVEL	<pre>struct { CSA address; LEVEL dimLevel; }</pre>	The parameter “dimLevel” determines the volume decrease when dim is active. (IO_BOARD_STEREO_DIM).
IO_BOARD_STEREO_TB_DIMLEVEL	<pre>struct { CSA address; LEVEL talkbackDimLevel; }</pre>	The parameter “talkbackDimLevel” determines the volume decrease when talkback dim (IO_BOARD_STEREO_TB_DIM) is active.

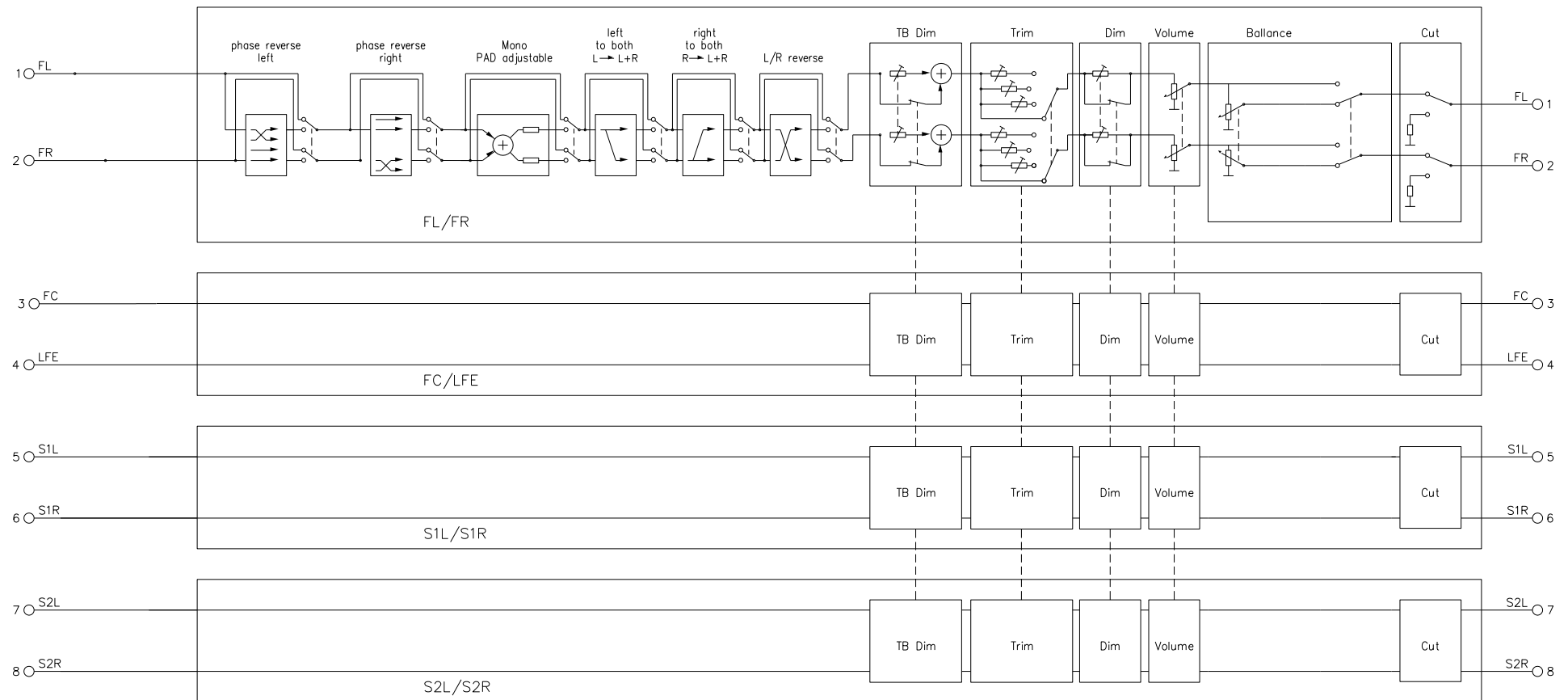
IO_BOARD_STEREO_TB_PRELEVEL	<pre>struct { CSA address; LEVEL talkbackPreLevel; }</pre>	The talkback pre level determines the amplification of the talkback signal if talkback pre left (IO_BOARD_STEREO_TB_PRE_L) or talkback pre right (IO_BOARD_STEREO_TB_PRE_R) is active.
IO_BOARD_STEREO_TB_POSTLEVEL	<pre>struct { CSA address; LEVEL talkbackPostLevel; }</pre>	The talkback post level determines the amplification of the talkback signal if talkback post left (IO_BOARD_STEREO_TB_POST_L) or talkback post right (IO_BOARD_STEREO_TB_POST_R) is active.
IO_BOARD_STEREO_TRIMLEVEL1, IO_BOARD_STEREO_TRIMLEVEL2 or IO_BOARD_STEREO_TRIMLEVEL3	<pre>struct { CSA address; LEVEL trimLevel; }</pre>	Trim levels are preset level offsets which can e.g. be used for adjusting different sets of loudspeakers to the same level.
IO_BOARD_STEREO_BALANCE	<pre>struct { CSA address; RANGE balance; }</pre>	The valid range of the balance value is -20...+20.
IO_BOARD_STEREO_PHASE_ INVERSION_L IO_BOARD_STEREO_PHASE_ INVERSION_R	<pre>struct { CSA address; byte phaseInversion; }</pre>	The parameter "phaseInversion" contains ON or OFF depending on whether the phase inversion is used or not.
IO_BOARD_STEREO_SWAP	<pre>struct { CSA address; byte swap; }</pre>	The parameter "swap" contains ON or OFF depending on whether the swap left right is used or not.
IO_BOARD_STEREO_MONOMIX	<pre>struct { CSA address; byte monomix; }</pre>	The parameter "monomix" contains ON or OFF depending on whether the mono mixer is used or not.
IO_BOARD_STEREO_LEFT_TO_BOTH	<pre>struct { CSA address; byte leftToBoth; }</pre>	The parameter "leftToBoth" contains ON or OFF depending on whether left-to-both is used or not.

IO_BOARD_STEREO_RIGHT_TO_BOTH	<pre>struct { CSA address; byte rightToBoth; }</pre>	The parameter “rightToBoth” contains ON or OFF depending on whether right-to-both is used or not.
IO_BOARD_STEREO_SOLO_L	<pre>struct { CSA address; byte soloLeft; }</pre>	The parameter “soloLeft” contains ON or OFF depending on whether solo-left is used or not.
IO_BOARD_STEREO_SOLO_R	<pre>struct { CSA address; byte soloRight; }</pre>	The parameter “soloRight” contains ON or OFF depending on whether solo-right is used or not.
IO_BOARD_STEREO_DIM	<pre>struct { CSA address; byte dim; }</pre>	The parameter “dim” contains ON or OFF depending on whether the monitoring is dimmed or not.
IO_BOARD_STEREO_TB_DIM	<pre>struct { CSA address; byte talkbackDim; }</pre>	The parameter “talkbackDim” contains ON or OFF depending on whether talkback is dimmed or not.
IO_BOARD_STEREO_BALANCE_OFF	<pre>struct { CSA address; byte balanceOff; }</pre>	The parameter “balanceOff” contains ON or OFF depending on whether balance is switched on or off.
IO_BOARD_STEREO_CUT_L IO_BOARD_STEREO_CUT_R	<pre>struct { CSA address; byte cut; }</pre>	The parameter “cut” contains ON or OFF depending on whether the left (respective the right) channel is cut or not.
IO_BOARD_STEREO_CUT_PRETBPOST_L IO_BOARD_STEREO_CUT_PRETBPOST_R	<pre>struct { CSA address; byte cut; }</pre>	The parameter “cut” contains ON or OFF depending on whether cut left pre talkback or cut right pre talkback is used or not.
IO_BOARD_STEREO_TB_PRE_L IO_BOARD_STEREO_TB_PRE_R	<pre>struct { CSA address; byte talkback; }</pre>	The parameter “talkback” contains ON or OFF depending on whether talkback pre left or talkback pre right is used or not.

IO_BOARD_STEREO_TB_POST_L IO_BOARD_STEREO_TB_POST_R	<pre>struct { CSA address; byte talkback; }</pre>	The parameter "talkback" contains ON or OFF depending on whether talkback post left or talkback post right is used or not.
IO_BOARD_STEREO_TB_SELECT_2	<pre>struct { CSA address; byte talkbackSelect2; }</pre>	The parameter "talkbackSelect2" contains ON if the talkback input 2 is used or OFF if the talkback input 1 is used.
IO_BOARD_STEREO_PPM_PRE	<pre>struct { CSA address; byte ppmPre; }</pre>	The parameter "ppmPre" contains ON if the PPM output is connected to the input signals of the monitoring device or OFF if the PPM output is connected to the output of the balance section of the monitoring device.

6.9 Surround Monitoring Units

A system can include multiple surround monitoring units. With these monitoring units it is possible to realize a comfortable surround monitoring. The structure of these devices is shown in the following diagram:



The inputs and outputs of the surround monitoring unit are internal targets and sources of the matrix and identified by CSAs. Therefore the units can be inserted into a signal path by standard connect procedures. All parameters of the surround monitoring units are controlled similar to the stereo monitoring units.

head.m	Parameter Set	Comments
IO_BOARD_SURROUND_MON_VOLUME	<pre>struct { CSA address; LEVEL volume; }</pre>	<p>The volume parameter contains the amplification in dB in steps of 1/32.</p> <p>The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.</p>
IO_BOARD_SURROUND_DIMLEVEL	<pre>struct { CSA address; LEVEL dimLevel; }</pre>	<p>The dim level determines the lowering of the volume if dim (IO_BOARD_SURROUND_DIM) is active.</p>
IO_BOARD_SURROUND_TB_DIMLEVEL	<pre>struct { CSA address; LEVEL dimLevel; }</pre>	<p>The talkback dim level determines the lowering of the volume if talkback dim (IO_BOARD_SURROUND_TB_DIM) is active.</p>
IO_BOARD_SURROUND_MONO_DIMLEVEL	<pre>struct { CSA address; LEVEL monoDimLevel; }</pre>	<p>The mono dim level determines the lowering of the volume if the mono matrix (IO_BOARD_SURROUND_MONO) is active.</p>
IO_BOARD_SURROUND_TRIM1_FL, IO_BOARD_SURROUND_TRIM2_FL or IO_BOARD_SURROUND_TRIM3_FL for front left channel	<pre>struct { CSA address; LEVEL trimLevel; }</pre>	<p>Trim levels are preset level offsets which can e.g. be used for adjusting different sets of loudspeakers to the same level.</p>
IO_BOARD_SURROUND_TRIM1_FR, IO_BOARD_SURROUND_TRIM2_FR or IO_BOARD_SURROUND_TRIM3_FR for front right channel	<pre>struct { CSA address; LEVEL trimLevel; }</pre>	<p>Trim levels are preset level offsets which can e.g. be used for adjusting different sets of loudspeakers to the same level.</p>
IO_BOARD_SURROUND_TRIM1_FC, IO_BOARD_SURROUND_TRIM2_FC or IO_BOARD_SURROUND_TRIM3_FC for front center channel	<pre>struct { CSA address; LEVEL trimLevel; }</pre>	<p>Trim levels are preset level offsets which can e.g. be used for adjusting different sets of loudspeakers to the same level.</p>
IO_BOARD_SURROUND_TRIM1_LFE, IO_BOARD_SURROUND_TRIM2_LFE or IO_BOARD_SURROUND_TRIM3_LFE for LFE	<pre>struct { CSA address; LEVEL trimLevel; }</pre>	<p>Trim levels are preset level offsets which can e.g. be used for adjusting different sets of loudspeakers to the same level.</p>

IO_BOARD_SURROUND_TRIM1_S1L, IO_BOARD_SURROUND_TRIM2_S1L or IO_BOARD_SURROUND_TRIM3_S1L for surround 1 left channel	struct { CSA address; LEVEL trimLevel; }	Trim levels are preset level offsets which can e.g. be used for adjusting different sets of loudspeakers to the same level.
IO_BOARD_SURROUND_TRIM1_S1R, IO_BOARD_SURROUND_TRIM2_S1R or IO_BOARD_SURROUND_TRIM3_S1R for surround 1 right channel	struct { CSA address; LEVEL trimLevel; }	Trim levels are preset level offsets which can e.g. be used for adjusting different sets of loudspeakers to the same level.
IO_BOARD_SURROUND_TRIM1_S2L, IO_BOARD_SURROUND_TRIM2_S2L or IO_BOARD_SURROUND_TRIM3_S2L for surround 2 left channel	struct { CSA address; LEVEL trimLevel; }	Trim levels are preset level offsets which can e.g. be used for adjusting different sets of loudspeakers to the same level.
IO_BOARD_SURROUND_TRIM1_S2R, IO_BOARD_SURROUND_TRIM2_S2R or IO_BOARD_SURROUND_TRIM3_S2R for surround 2 right channel	struct { CSA address; LEVEL trimLevel; }	Trim levels are preset level offsets which can e.g. be used for adjusting different sets of loudspeakers to the same level.
IO_BOARD_SURROUND_BALANCE_LR for Balance Front Left – Front Right IO_BOARD_SURROUND_BALANCE_FB for Balance Front – Back	struct { CSA address; RANGE balance; }	The range of the balance value is -20...+20
IO_BOARD_SURROUND_TB_DIM	struct { CSA address; byte talkbackDim; }	The parameter “talkbackDim” contains ON or OFF depending on whether talkback dim is used or not.
IO_BOARD_SURROUND_ PHASE_INVERSION_L	struct { CSA address; byte phaseInversion; }	The parameter “phaseInversion” contains ON or OFF depending on whether the phase inversion is used or not.
IO_BOARD_SURROUND_ PHASE_INVERSION_R	struct { CSA address; byte phaseInversion; }	The parameter “phaseInversion” contains ON or OFF depending on whether the phase inversion is used or not.

IO_BOARD_SURROUND_MONOMIX	<pre>struct { CSA address; byte monomix; }</pre>	The parameter “monomix” contains ON or OFF depending on whether the mono mixer is used or not.
IO_BOARD_SURROUND_LEFT_TO_BOTH	<pre>struct { CSA address; byte leftToBoth; }</pre>	The parameter “leftToBoth” contains ON or OFF depending on whether left-to-both is used or not.
IO_BOARD_SURROUND_RIGHT_TO_BOTH	<pre>struct { CSA address; byte rightToBoth; }</pre>	The parameter “rightToBoth” contains ON or OFF depending on whether right-to-both is used or not.
IO_BOARD_SURROUND_SWAP	<pre>struct { CSA address; byte swap; }</pre>	The parameter “swap” contains ON or OFF depending on whether the swap-left-right is used or not.
IO_BOARD_SURROUND_DIM	<pre>struct { CSA address; byte dim; }</pre>	The parameter “dim” contains ON or OFF depending on whether dim is used or not.
IO_BOARD_SURROUND_BALANCE_OFF	<pre>struct { CSA address; byte balanceOff; }</pre>	The parameter “balanceOff” contains ON or OFF depending on whether balance is switched off or not.

Please note that the maximum output level is a result of a computation including all parameters of the levelling chain (e.g. Talkback Dimlevel, Trimlevels, etc.) and is altogether limited to +15 dB. The same procedure applies to the minimum output level (limit: -128 dB). The “input parameters” for these computations, i.e. the levelling chain parameters, are not limited in range, i.e. the full (signed) 16-bit range of the LEVEL type may be used.

6.10 Silence Detect

Silence detect units can be configured in inputs and/or in outputs of the matrix. If a signal is detected as silent or as available again the following message is sent:

IO_SILENCE_DETECT			
Message Type	Parameter	Comment	
PK_EVENT	None	Not implemented.	
PK_CHANGE	struct { CSA address; byte silence; }	address	CSA of the input or output.
		silence	The detected state of the signal. Specifies whether the signal has changed from silence to signal (SIGNAL) or from signal to silence (SILENCE).
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.	
PK_QUERY_REPLY	struct { CSA address; byte silence; }	See PK_CHANGE above.	

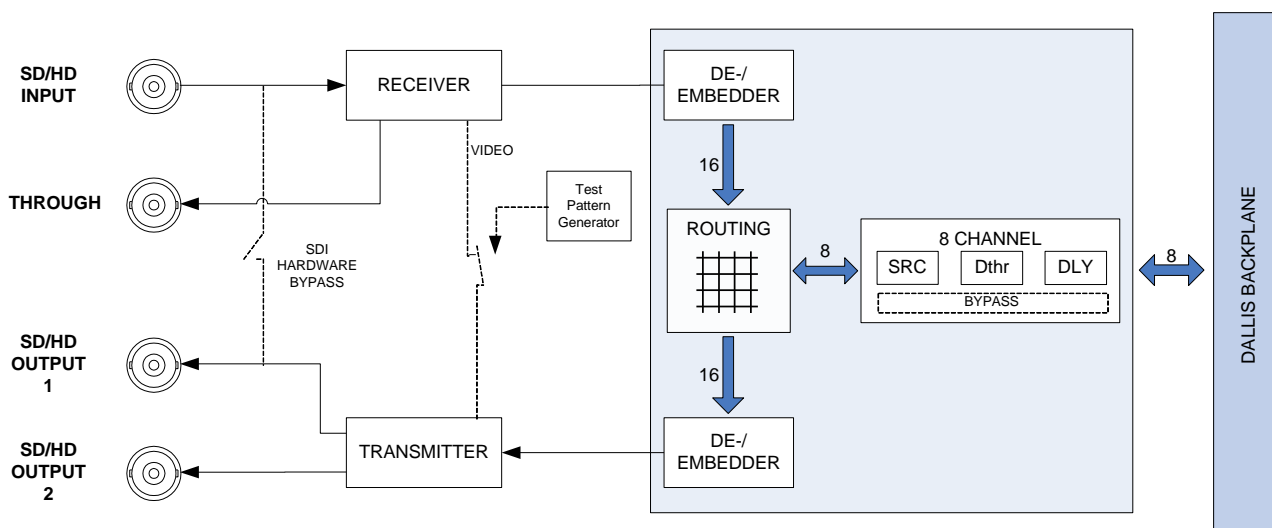
Note: The system can handle a maximum of 512 simultaneous (mono or stereo linked) silence detects.

IO_SILENCE_DETECT_CONFIG			
Message Type	Parameter	Comment	
PK_EVENT	<pre> struct { CSA address; byte mode; LEVEL thresholdLeft; LEVEL thresholdRight; byte silenceDetTime_L; byte silenceDetTime_R; byte signalDetTime_L; byte signalDetTime_R; } </pre>	address	CSA of the input or output.
		mode	<p>A bitwise coded flag for the silence detector mode:</p> <p>Bit 0 = 1: Left Channel on</p> <p>Bit 1 = 1: Right Channel on</p> <p>Bit 2 = 1: Stereo Link</p> <p>Stereo Link means that all configuration settings are mirrored for the left and the right channel and both channels will be treated as a unit. Therefore, a silent signal has to be silent on both channels for the configured duration.</p> <p>As a precondition for the Stereo Link feature, the Left Channel On and Right Channel On bits have to be set.</p>
		thresholdLeft	<p>Threshold level for the left channel.</p> <p>The range is <i>-128 to +15 dB</i>. This yields LEVEL values of <i>-4096 to +480</i>.</p>
		thresholdRight	<p>Threshold level for the right channel.</p> <p>The range is <i>-128 to +15 dB</i>. This yields LEVEL values of <i>-4096 to +480</i>.</p>
		silenceDetTime_L	Determines the timespan in seconds used to detect silence (0-255s).
		silenceDetTime_R	Determines the timespan in seconds used to detect silence (0-255s).
		signalDetTime_L	Determines the timespan in seconds used to detect a signal (0-255s).
		signalDetTime_R	Determines the timespan in seconds used to detect a signal (0-255s).

IO_SILENCE_DETECT_CONFIG		
Message Type	Parameter	Comment
PK_CHANGE	<pre>struct { CSA address; byte Mode; LEVEL ThresholdLeft; LEVEL ThresholdRight; byte SilenceDetectT_L; byte SilenceDetectT_R; byte SignalDetectT_L; byte SignalDetectT_R; }</pre>	Contains the values that have been set by the hardware.
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.
PK_QUERY_REPLY	<pre>struct { CSA address; byte Mode; LEVEL ThresholdLeft; LEVEL ThresholdRight; byte SilenceDetectT_L; byte SilenceDetectT_R; byte SignalDetectT_L; byte SignalDetectT_R; }</pre>	See PK_EVENT above.

6.11 DALLIS SDI HD/SD Units

BLOCK DIAGRAM HD/SD SDI-INTERFACE



6.11.1 Input and output sections

Inputs and outputs of SDI embedder/deembedder boards can be controlled via the commands listed in the following paragraphs. The parameters of the SDI units are generally in stereo format. These parameters are indexed by the CSA of the left input and left output signal respectively. SDI inputs are identified by DEVICETYPE_SDI_IN, outputs are identified by DEVICETYPE_SDI_OUT.

6.11.2 Input section

head.m	Parameter Set	Comments																
IO_SDI_DEEMB_GROUP_CHAN_SELECT	<pre>struct { CSA address; byte select; }</pre>	<p>The select parameter selects the SDI channel pair for deembedding into this matrix source.</p> <p>The valid range of select is 1 to 8.</p> <table><tr><td>group 1, channel 1 and 2</td><td>1</td></tr><tr><td>group 1, channel 3 and 4</td><td>2</td></tr><tr><td>group 2, channel 1 and 2</td><td>3</td></tr><tr><td>group 2, channel 3 and 4</td><td>4</td></tr><tr><td>group 3, channel 1 and 2</td><td>5</td></tr><tr><td>group 3, channel 3 and 4</td><td>6</td></tr><tr><td>group 4, channel 1 and 2</td><td>7</td></tr><tr><td>group 4, channel 3 and 4</td><td>8</td></tr></table>	group 1, channel 1 and 2	1	group 1, channel 3 and 4	2	group 2, channel 1 and 2	3	group 2, channel 3 and 4	4	group 3, channel 1 and 2	5	group 3, channel 3 and 4	6	group 4, channel 1 and 2	7	group 4, channel 3 and 4	8
group 1, channel 1 and 2	1																	
group 1, channel 3 and 4	2																	
group 2, channel 1 and 2	3																	
group 2, channel 3 and 4	4																	
group 3, channel 1 and 2	5																	
group 3, channel 3 and 4	6																	
group 4, channel 1 and 2	7																	
group 4, channel 3 and 4	8																	
IO_SDI_DEEMB_DELAY	<pre>struct { CSA address; TIME delayTime; }</pre>	<p>The delayTime determines the delay of the deembedded audio signal in milliseconds in steps of 1 millisecond. The valid range is 0 to 240.</p>																
IO_SDI_DEEMB_DELAY_ON	<pre>struct { CSA address; byte delayOn; }</pre>	<p>The parameter delayOn contains ON or OFF depending on whether the delay is active or not.</p>																

6.11.3 Output section

head.m	Parameter Set	Comments																		
IO_SDI_EMB_GROUP_CHAN_SELECT	<pre>struct { CSA address; byte select; }</pre>	<p>The select parameter selects the SDI channel pair for embedding from this matrix target.</p> <p>The range of select is 0 to 8.</p> <table><tr><td>no embedding</td><td>0</td></tr><tr><td>group 1, channel 1 and 2</td><td>1</td></tr><tr><td>group 1, channel 3 and 4</td><td>2</td></tr><tr><td>group 2, channel 1 and 2</td><td>3</td></tr><tr><td>group 2, channel 3 and 4</td><td>4</td></tr><tr><td>group 3, channel 1 and 2</td><td>5</td></tr><tr><td>group 3, channel 3 and 4</td><td>6</td></tr><tr><td>group 4, channel 1 and 2</td><td>7</td></tr><tr><td>group 4, channel 3 and 4</td><td>8</td></tr></table>	no embedding	0	group 1, channel 1 and 2	1	group 1, channel 3 and 4	2	group 2, channel 1 and 2	3	group 2, channel 3 and 4	4	group 3, channel 1 and 2	5	group 3, channel 3 and 4	6	group 4, channel 1 and 2	7	group 4, channel 3 and 4	8
no embedding	0																			
group 1, channel 1 and 2	1																			
group 1, channel 3 and 4	2																			
group 2, channel 1 and 2	3																			
group 2, channel 3 and 4	4																			
group 3, channel 1 and 2	5																			
group 3, channel 3 and 4	6																			
group 4, channel 1 and 2	7																			
group 4, channel 3 and 4	8																			
IO_SDI_EMB_DELAY	<pre>struct { CSA address; TIME delayTime; }</pre>	<p>The delayTime determines the delay of the embedded audio signal in milliseconds in steps of 1 millisecond.</p> <p>The valid range is 0 to 240.</p>																		
IO_SDI_EMB_DELAY_ON	<pre>struct { CSA address; byte delayOn; }</pre>	<p>The parameter delayOn contains ON or OFF depending on whether the delay is active or not.</p>																		
IO_SDI_EMB_WORDLENGTH	<pre>struct { CSA address; byte wordLength; }</pre>	<p>The parameter wordLength determines the wordlength of the embedded audio signal.</p> <table><tr><td>16 bit</td><td>0</td></tr><tr><td>20 bit</td><td>4</td></tr><tr><td>24 bit</td><td>8</td></tr></table>	16 bit	0	20 bit	4	24 bit	8												
16 bit	0																			
20 bit	4																			
24 bit	8																			

6.11.4 Device section

The device section of the SDI HD/SD unit contains all device control elements. Since the device section has no input and output signals, it is indexed by the CSA of any left input signal or left output signal respectively. Parameter changes in the device section will be reported with the CSA of the first left input signal or the first left output signal, depending on whether the operating mode defines input signals or not.

Message Type		Indication
Request for data change	PK_EVENT	Any left input signal or left output signal.
Query for data	PK_QUERY	
Successful query reply	PK_QUERY_REPLY	
Unsuccessful query reply	PK_QUERY_NOREPLY	
Access denied	PK_ACCESS_DENIED	
Report of a data change	PK_CHANGE	The first left input signal or the first left output signal, if the operating mode defines no input signals

head.m	Parameter Set	Comments
IO_SDI_SRC	<pre>struct { CSA address; byte src; }</pre>	The parameter src contains ON or OFF depending on whether the sample rate converter is active or not.
IO_SDI_DELAY_SELECT	<pre>struct { CSA address; byte delaySelect; }</pre>	<p>The parameter delaySelect determines the assignment of the delays. The delays can be assigned to the input section (deembedder) or to the output section (embedder)</p> <p>Delays are assigned to the 0 input section (deembedder)</p> <p>Delays are assigned to the 1 output section (embedder)</p>
IO_SDI_EMBEDDER_MODE	<pre>struct { CSA address; byte embedderMode; }</pre>	<p>The parameter embedderMode determines the operation mode of the embedders.</p> <p>On 0 Clean 1 Off 2</p>
IO_SDI_GENERATOR_MODE	<pre>struct { CSA address; byte generatorMode; }</pre>	<p>The parameter generatorMode determines the operation mode of the video generator.</p> <p>automatic (if lock fails) 0 On 1</p>

IO_SDI_GENERATOR_COLOR	<pre>struct { CSA address; byte generatorColor; }</pre>	<p>The parameter generatorColor determines the color of the video generator.</p> <table><tr><td>black</td><td>0</td></tr><tr><td>color bars</td><td>1</td></tr></table>	black	0	color bars	1																																				
black	0																																									
color bars	1																																									
IO_SDI_GENERATOR_FORMAT	<pre>struct { CSA address; byte generatorFormat; }</pre>	<p>The parameter generatorFormat determines the format of the video generator.</p> <p>The range of generatorFormat is 0 to 13.</p> <table><tr><td>automatic</td><td>0</td></tr><tr><td>720p50</td><td>1</td></tr><tr><td>720p59_94</td><td>2</td></tr><tr><td>720p60</td><td>3</td></tr><tr><td>1080i50</td><td>4</td></tr><tr><td>1080i59_94</td><td>5</td></tr><tr><td>1080i60</td><td>6</td></tr><tr><td>1080p23_98</td><td>7</td></tr><tr><td>1080p24</td><td>8</td></tr><tr><td>1080p25</td><td>9</td></tr><tr><td>1080p29_97</td><td>10</td></tr><tr><td>1080p30</td><td>11</td></tr><tr><td>SD NTSC</td><td>12</td></tr><tr><td>SD PAL</td><td>13</td></tr></table> <p>The abbreviations above are based on the following structure:</p> <p>lines – progressive/interlaced – frame rate</p> <p>Example:</p> <table><tr><td>720p50</td><td>720 lines</td></tr><tr><td></td><td>progressive</td></tr><tr><td></td><td>50 frames per second</td></tr></table> <p>Note:</p> <p>For SD-Board only the following values are valid:</p> <table><tr><td>automatic</td><td>0</td></tr><tr><td>SD NTSC</td><td>12</td></tr><tr><td>SD PAL</td><td>13</td></tr></table>	automatic	0	720p50	1	720p59_94	2	720p60	3	1080i50	4	1080i59_94	5	1080i60	6	1080p23_98	7	1080p24	8	1080p25	9	1080p29_97	10	1080p30	11	SD NTSC	12	SD PAL	13	720p50	720 lines		progressive		50 frames per second	automatic	0	SD NTSC	12	SD PAL	13
automatic	0																																									
720p50	1																																									
720p59_94	2																																									
720p60	3																																									
1080i50	4																																									
1080i59_94	5																																									
1080i60	6																																									
1080p23_98	7																																									
1080p24	8																																									
1080p25	9																																									
1080p29_97	10																																									
1080p30	11																																									
SD NTSC	12																																									
SD PAL	13																																									
720p50	720 lines																																									
	progressive																																									
	50 frames per second																																									
automatic	0																																									
SD NTSC	12																																									
SD PAL	13																																									

6.12 DALLIS SDI 3G Units

As of version 4.12 of the Remote MNOPL protocol, 3rd generation DALLIS SDI units are supported.

6.12.1 Output section

Inputs and outputs of SDI 3G embedder/deembedder boards can be controlled via the commands listed in the following paragraphs. The parameters of the SDI 3G units are generally in stereo format. These parameters are indexed by the CSA of the left input or output signal. SDI 3G inputs are identified by DEVICETYPE_3G_SDI_IN, outputs are identified by DEVICETYPE_3G_SDI_OUT when querying the device type.

Please note that SDI 3G input signals do not provide controllable elements in the signal layer. For controlling the device layer, please refer to chapter 6.12.2.

For information about the general data format structure of the different available message types, please refer to chapter 2.6.1.

head.m	Parameter Set	Comments						
IO_SDI_3G_EMB_DELAY	<pre>struct { CSA address; TIME time; }</pre>	<p>The delay time determines the delay of the embedded audio signal in milliseconds in steps of 1 millisecond.</p> <p>The valid range is 0 to 160.</p>						
IO_SDI_3G_EMB_DELAY_ON	<pre>struct { CSA address; byte on; }</pre>	<p>The parameter “on” contains ON or OFF depending on whether the delay shall be active or not.</p>						
IO_SDI_3G_EMB_WORDLENGTH	<pre>struct { CSA address; byte wordlength; }</pre>	<p>The parameter “wordlength” determines the wordlength of the embedded audio signal.</p> <table><tr><td>16 bit</td><td>0</td></tr><tr><td>20 bit</td><td>4</td></tr><tr><td>24 bit</td><td>8</td></tr></table>	16 bit	0	20 bit	4	24 bit	8
16 bit	0							
20 bit	4							
24 bit	8							

6.12.2 Device section

The device section of the SDI 3G unit contains all device control elements. Since the device section has no input and output signals, it is indexed by the CSA of any of its left input signals or left output signals respectively. Parameter changes in the device section will be reported with the CSA of the first left input signal or the first left output signal, depending on whether the operating mode defines input signals or not.

Message Type		Indication
Request for data change	PK_EVENT	Any left input signal or left output signal.
Query for data	PK_QUERY	
Successful query reply	PK_QUERY_REPLY	
Unsuccessful query reply	PK_QUERY_NOREPLY	
Access denied	PK_ACCESS_DENIED	
Report of a data change	PK_CHANGE	The first left input signal or the first left output signal, if the operating mode defines no input signals

head.m	Parameter Set	Comments
IO_SDI_3G_CTRL_SRC	<pre>struct { CSA address; byte src; }</pre>	The parameter "src" contains ON or OFF depending on whether the sample rate converter is active or not.
IO_SDI_3G_CTRL_CLEAN	<pre>struct { CSA address; byte clean; }</pre>	Valid settings for "clean" : ON / OFF If "clean" is switched to ON, the SDI embedder removes all upstream audio and metadata information and generates a new ancillary data space structure.
IO_SDI_3G_CTRL_VIDEO_DELAY_ENABLE	<pre>struct { CSA address; byte enable; }</pre>	The parameter "enable" contains ON or OFF depending on whether the video delay shall be used or not.
IO_SDI_3G_VIDEO_DELAY	<pre>struct { CSA address; byte frames; }</pre>	The parameter "frames" contains the number of frames for the video delay. The valid range is 0 to 8.
IO_SDI_3G_GEN_PATTERN	<pre>struct { CSA address; byte pattern; }</pre>	The parameter „pattern“ determines the pattern that shall be used for the video generator. Valid settings are: <div style="display: flex; justify-content: space-between; padding: 0 20px;"> <div>0</div> <div>Color Bars</div> </div> <div style="display: flex; justify-content: space-between; padding: 0 20px;"> <div>1</div> <div>Black</div> </div>

IO_SDI_3G_GEN_SELECT	<pre>struct { CSA address; byte select; }</pre>	<p>The parameter “select” determines the video format for the video generator signal.</p> <p>Valid settings are:</p> <table><tr><td>0</td><td>Auto</td></tr><tr><td>1</td><td>3G 1080p60</td></tr><tr><td>2</td><td>3G 1080p59.94</td></tr><tr><td>3</td><td>3G 1080p50</td></tr><tr><td>4</td><td>HD 1080i60</td></tr><tr><td>5</td><td>HD 1080i59.94</td></tr><tr><td>6</td><td>HD 1080i50</td></tr><tr><td>7</td><td>HD 1080p30</td></tr><tr><td>8</td><td>HD 1080p29.97</td></tr><tr><td>9</td><td>HD 1080p25</td></tr><tr><td>10</td><td>HD 1080p24</td></tr><tr><td>11</td><td>HD 1080p23.98</td></tr><tr><td>12</td><td>HD 720p60</td></tr><tr><td>13</td><td>HD 720p59.94</td></tr><tr><td>14</td><td>HD 720p50</td></tr><tr><td>15</td><td>SD 525</td></tr><tr><td>16</td><td>SD 625</td></tr></table> <p>The abbreviations above are based on the following structure:</p> <p>lines – progressive/interlaced – frame rate</p> <p>Example:</p> <table><tr><td>720p50</td><td>720 lines</td></tr><tr><td></td><td>progressive</td></tr><tr><td></td><td>50 frames per second</td></tr></table>	0	Auto	1	3G 1080p60	2	3G 1080p59.94	3	3G 1080p50	4	HD 1080i60	5	HD 1080i59.94	6	HD 1080i50	7	HD 1080p30	8	HD 1080p29.97	9	HD 1080p25	10	HD 1080p24	11	HD 1080p23.98	12	HD 720p60	13	HD 720p59.94	14	HD 720p50	15	SD 525	16	SD 625	720p50	720 lines		progressive		50 frames per second
0	Auto																																									
1	3G 1080p60																																									
2	3G 1080p59.94																																									
3	3G 1080p50																																									
4	HD 1080i60																																									
5	HD 1080i59.94																																									
6	HD 1080i50																																									
7	HD 1080p30																																									
8	HD 1080p29.97																																									
9	HD 1080p25																																									
10	HD 1080p24																																									
11	HD 1080p23.98																																									
12	HD 720p60																																									
13	HD 720p59.94																																									
14	HD 720p50																																									
15	SD 525																																									
16	SD 625																																									
720p50	720 lines																																									
	progressive																																									
	50 frames per second																																									
IO_SDI_3G_CTRL_VIDEOGEN_FORCE	<pre>struct { CSA address; byte force; }</pre>	<p>If the parameter “force” is set to OFF, the video generator will automatically use the detected frame format of the SDI video input signal (if available).</p> <p>If this behaviour is inappropriate, setting the parameter “force” to ON forces the generator to use the preselected video format (IO_SDI_3G_GEN_SELECT).</p>																																								
IO_SDI_3G_METADATA_EMBEDDER_ENABLE	<pre>struct { CSA address; byte enable; }</pre>	<p>If the parameter “enable” is set to ON, the SDI metadata embedder will be enabled, if set to OFF, the metadata embedder will be disabled.</p>																																								

IO_SDI_3G_METADATA_EMBEDDER_MODE	<pre>struct { CSA address; byte mode; }</pre>	<p>Using the “mode” parameter, the behaviour of the metadata embedder can be changed between:</p> <p>0 for automatic line selection</p> <p>1 for using the pre-selected video line (see IO_SDI_3G_METADATA_EMBEDDER_LINE_SELECT)</p>
IO_SDI_3G_METADATA_EMBEDDER_LINE_SELECT	<pre>struct { CSA address; byte select; }</pre>	<p>Using the “select” parameter, the metadata embedder line can be preselected.</p> <p>The valid range is 11 to 18.</p>
IO_SDI_3G_METADATA_EMBEDDER_SELECT	<pre>struct { CSA address; byte select; }</pre>	<p>For each of the two metadata embedders, this control parameter selects, into which stream the metadata shall be embedded.</p> <p>The index of the metadata embedder is specified in the head.n field (0 or 1).</p> <p>The “select” parameter specifies the VANC stream index for the embedding in a range of 1 to 9.</p>
IO_SDI_3G_METADATA_DEMBEDDER_SELECT	<pre>struct { CSA address; byte select; }</pre>	<p>For each of the two metadata deembedders, this control parameter selects, from which stream the metadata shall be deembedded.</p> <p>The index of the metadata deembedder is specified in the head.n field (0 or 1).</p> <p>The “select” parameter specifies the VANC stream index for the deembedding in a range of 1 to 9.</p>
IO_SDI_3G_METADATA_EMBEDDER_ROUTING	<pre>struct { CSA address; byte routing; }</pre>	<p>The embedder routing specifies the device from which the audio signal to be embedded is taken on a per-channel basis.</p> <p>The channel index is specified in the head.n field in a range from 0 to 15, where 0 means group 1/channel 1, 1 means group 1/channel 2, ..., 15 means group 4/channel 4.</p> <p>The “routing” parameter specifies the source for the embedder in a range of</p> <p>0 – 15 to take the audio from the linked deembedder</p> <p>16 – 31 to take the audio from the DALLIS output channel</p>

IO_SDI_3G_DEEMBEDDER_ROUTING	<pre>struct { CSA address; byte routing; }</pre>	<p>The deembedder routing specifies the assignment of the SDI audio information to the DALLIS input channels.</p> <p>The DALLIS input signal index is specified in the head.n field in a range from 0 to 15.</p> <p>The “routing” parameter specifies the source for the embedder in a range of 0 to 15, where 0 means group 1/channel 1, 1 means group 1/channel 2, ..., 15 means group 4/channel 4.</p>				
IO_SDI_3G_GROUP1_EMBEDDER_ON IO_SDI_3G_GROUP2_EMBEDDER_ON IO_SDI_3G_GROUP3_EMBEDDER_ON IO_SDI_3G_GROUP4_EMBEDDER_ON	<pre>struct { CSA address; byte on; }</pre>	<p>The parameter “on” dis-/enables the audio embedder for the specified SDI group. Valid settings for “on” are ON and OFF.</p>				
IO_SDI_3G_SDHD_LOCK_STATE	<pre>struct { CSA address; byte lock; }</pre>	<p>This is a read only parameter which determines the lock state of the SDI signal.</p> <p>Only PK_QUERY messages are allowed for this head.m value and the parameter “lock” has to be interpreted as follows:</p> <table><tr><td>0</td><td>SDI signal unlocked or invalid</td></tr><tr><td>1</td><td>SDI signal is locked.</td></tr></table>	0	SDI signal unlocked or invalid	1	SDI signal is locked.
0	SDI signal unlocked or invalid					
1	SDI signal is locked.					

6.13 DALLIS IP Codec Units

The DALLIS IP Codec control is only available for devices with device types “DEVICETYPE_IPCODEC_IN” and “DEVICETYPE_IPCODEC_OUT”. The control commands are accepted on all signal CSAs but the real control will be done by the IP Codec board (which has no own CSA). Therefore all returned values from the server will be addressed using the first available source signal CSA for the accessed board. Best practice is to also send the control commands using the first source signal CSA.

6.13.1 IP Codec Control

The DALLIS IP codec offers three GPO controls, providing the possibility to map some user functions via the IP codec's web browser setup utility. They are addressed using the Remote MNOPL head.n field zero based. This means that e.g. for addressing the 2nd GPO value the head.n field has to be set to “1”.

IO_DALLIS_IPCODEC_GPO_VALUE		
Message Type	Parameter	Comment
PK_EVENT	<pre>struct { CSA address; byte value; }</pre>	Sets the GPO indexed by the head.n field to the specified “value”. Valid “value” settings are 0 and 1.
PK_CHANGE	<pre>struct { CSA address; byte value; }</pre>	Informs whenever the state of value changes.
PK_QUERY	CSA	Queries the state of the GPO value.

IO_DALLIS_IPCODEC_GPO_VALUE		
Message Type	Parameter	Comment
PK_QUERY_REPLY	<pre>struct { CSA address; byte value; }</pre>	See PK_EVENT above.

6.13.2 IP Codec Status

There are four status information bits that can be retrieved from the system. The values are read only, so PK_EVENT messages are not processed. Please refer to the following tables:

IO_DALLIS_IPCODEC_ALARM_STATE		
Message Type	Parameter	Comment
PK_EVENT	None	Not implemented.
PK_CHANGE	<pre>struct { CSA address; byte value; }</pre>	<p>Notifies whenever the DALLIS IP Codec Alarm state changes. Valid values are:</p> <p>“0” for no alarm.</p> <p>“1” for alarm.</p>
PK_QUERY	CSA	Queries the state of the IP Codec alarm.
PK_QUERY_REPLY	<pre>struct { CSA address; byte value; }</pre>	See PK_EVENT above.

IO_DALLIS_IPCODEC_NETWORK_CONNECTION_STATE		
Message Type	Parameter	Comment
PK_EVENT	None	Not implemented.
PK_CHANGE	<pre>struct { CSA address; byte value; }</pre>	<p>Notifies whenever the DALLIS IP Codec network connection state changes. Valid values are:</p> <p>“0” for no network connection.</p> <p>“1” for established network connection.</p>
PK_QUERY	CSA	Queries the state of the IP Codec network connection.
PK_QUERY_REPLY	<pre>struct { CSA address; byte value; }</pre>	See PK_EVENT above.



IO_DALLIS_IPCODEC_AUDIO_CONNECTION_STATE		
<i>Message Type</i>	<i>Parameter</i>	<i>Comment</i>
PK_EVENT	None	Not implemented.
PK_CHANGE	<pre>struct { CSA address; byte value; }</pre>	<p>Notifies whenever the DALLIS IP Codec audio connection state changes. Valid values are:</p> <p>“0” for no audio connection.</p> <p>“1” for established audio connection.</p>
PK_QUERY	CSA	Queries the state of the IP Codec audio connection.
PK_QUERY_REPLY	<pre>struct { CSA address; byte value; }</pre>	See PK_EVENT above.

IO_DALLIS_IPCODEC_TRANSPARENT_MODE_STATE		
<i>Message Type</i>	<i>Parameter</i>	<i>Comment</i>
PK_EVENT	None	Not implemented.
PK_CHANGE	<pre>struct { CSA address; byte value; }</pre>	<p>Notifies whenever the DALLIS IP Codec transparent mode is switched on or off. Valid values are:</p> <p>“0” for non-transparent mode.</p> <p>“1” for transparent mode.</p>
PK_QUERY	CSA	Queries the state of the IP Codec transparent mode setting.
PK_QUERY_REPLY	<pre>struct { CSA address; byte value; }</pre>	See PK_EVENT above.

7 Generic Control

7.1 Working with Audio-Follows-Video

To work with audio-follows-video the client has to subscribe to the AFV_EVENT stream at the server. If the AFV_EVENT stream is enabled the client will get any changes concerning audio-follows-video events (PK_CHANGE messages). If the server creates a change for an audio-follows-video event requested by a client, the appropriate information is automatically mirrored to all clients. More detailed information about AFV configuration can be found in chapter 10.6.

AFV_EVENT		
Message Type	Parameter	Comment
PK_EVENT	byte switch;	The head.n field contains the audio-follows-video event number (1...127). The switch parameter defines whether the audio-follows-video event state shall be set to ON or OFF.
PK_CHANGE	byte switch;	Contains the value that has been set by the hardware.
PK_QUERY	None	Query for the audio-follows-video event state
PK_QUERY_REPLY	byte switch;	The audio-follows-video event state

7.2 Working with GPIO

General Purpose Inputs and Outputs are also addressed by CSAs. If necessary, the client can distinguish between GPIO signals and audio signals using a device type query. The result of such a device type query for a GPIO signal will be DEVICETYPE_GPI or DEVICETYPE_GPO depending on the type of signal.

7.2.1 GPI State (DEVICETYPE_GPI)

IO_GPIO_VALUE						
Message Type	Parameter	Comment				
PK_EVENT	None	Not implemented.				
PK_CHANGE	<pre>struct { CSA address; STATE8 state; }</pre>	<div>Notification of a state change for a GPI signal.</div> <table><tr><td>address</td><td>CSA of the changed GPI.</td></tr><tr><td>state</td><td>State for the GPI (ON/OFF).</td></tr></table>	address	CSA of the changed GPI.	state	State for the GPI (ON/OFF).
address	CSA of the changed GPI.					
state	State for the GPI (ON/OFF).					
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.				
PK_QUERY_REPLY	<pre>struct { CSA address; STATE8 state; }</pre>	See PK_CHANGE above.				

7.2.2 GPO State (DEVICETYPE_GPO)

IO_GPIO_VALUE						
Message Type	Parameter	Comment				
PK_EVENT	<pre>struct { CSA address; STATE8 state; }</pre>	The state of a GPO can be changed by sending a new state.				
		<table><tr><td>address</td><td>CSA of the changed GPI.</td></tr><tr><td>state</td><td>New state requested for the GPO (ON/OFF).</td></tr></table>	address	CSA of the changed GPI.	state	New state requested for the GPO (ON/OFF).
		address	CSA of the changed GPI.			
state	New state requested for the GPO (ON/OFF).					
PK_CHANGE	<pre>struct { CSA address; STATE8 state; }</pre>	Contains the value that has been set by the hardware.				
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.				
PK_QUERY_REPLY	<pre>struct { CSA address; STATE8 state; }</pre>	See PK_EVENT above.				

7.3 Setting two Signals to Stereo Mode

With this command two signals may be switched to stereo mode, e.g. matrix dsp channels. The second signal that will be combined with the signal identified by the used CSA will automatically be chosen by the matrix. In case of an even numbered signal it will be the next odd numbered signal. In case of an odd number the previous even numbered signal will be used. If the router is configured with user-defined CSAs please refer to your configuration.

SIGNAL_STEREO						
Message Type	Parameter	Comment				
PK_EVENT	<pre>struct { CSA address; byte stereo; }</pre>	Request to set a signal to stereo mode linking it with the corresponding partner signal.				
		<table><tr><td>address</td><td>CSA of the signal.</td></tr><tr><td>stereo</td><td>ON or OFF depending on whether the signal shall be in stereo mode or not.</td></tr></table>	address	CSA of the signal.	stereo	ON or OFF depending on whether the signal shall be in stereo mode or not.
		address	CSA of the signal.			
stereo	ON or OFF depending on whether the signal shall be in stereo mode or not.					
PK_CHANGE	<pre>struct { CSA address; byte stereo; }</pre>	Contains the value that has been set by the hardware.				
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.				
PK_QUERY_REPLY	<pre>struct { CSA address; byte stereo; }</pre>	See PK_CHANGE above.				

8 Working with Protections

To work with protections, the client has to subscribe to the `CLIENT_REQUEST_LEVEL` and the `DATA_PROTECT_LEVEL` stream at the server. If the `DATA_PROTECT_LEVEL` stream is enabled, the client will get any changes concerning Protect Levels (`PK_CHANGE` messages) the server is sending to any other client. If the `CLIENT_REQUEST_LEVEL` stream is enabled, the client will get any changes concerning Request Levels (`PK_CHANGE` messages) the server is sending to the inquirer. This way it can automatically stay in sync with the routing entity in the server. If the server creates changes for the Protect Levels of sources, targets or data areas for a client, the appropriate information is automatically mirrored to all clients.

Note: If the data field contains definitions of unknown signals, the client does not get any answer. This is usually a configuration problem. To prevent from such problems the dynamic client configuration should be used.

Request Level (RL)	Each client has a Request Level which indicates <ul style="list-style-type: none"> the level for the processing of all of its change requests the maximum protect level the client can set
Protect Level (PL)	Use with sources and targets of the router: The Protect Level indicates the protection level for routing request messages.
	Use with data areas: The Protect Level indicates the protection level for data modification request messages.

Interaction of Request Level and Protect Level

A client with Request Level...	... can read data with Protection Level	... can modify data with Protection Level	.. can read the Protect Level	..can set the Protect Level to
0	0 - 3	No modification	0 - 3	No modification
1	0 - 3	0	0 - 3	0 -1
2	0 - 3	0 – 1	0 - 3	0 -2
3	0 - 3	0 – 2	0 - 3	0 -3

Initializations

Request Level	After having received a Login Message from a client, the server sets the Request Level to an initial value of 1. The Request Level can be modified by the client with a Request Level request message.
Protect Level	During a coldstart procedure all Protect Levels are set to an initial value of 0. During a warmstart procedure all Protect Levels remain unaffected.

Example of a client with Request Level 2 and audio targets

The client is allowed to query (read) the routing	of all Targets with Protect Level 0 – 3 (all)
The client is allowed to modify (write) the routing	of all Targets with Protect Level 0 – 1
The client is allowed to query (read) the Protect Level	of all Targets with Protect Level 0 – 3 (all)
The client is allowed to modify (write) the Protect Level	of all Targets with Protect Level 0 – 2 to 0 – 2

Notes:

If a Protect Level has been set to 2, only clients with Request Level 3 can modify the routing. Clients with Request Level 2 have to decrease the Protect Level, before change requests will be accepted.

If a Protect Level has been set to 3, no clients can modify the routing. A Client with Request Level 3 has to decrease the Protect Level, before change requests will be accepted.

8.1 Request Level

8.1.1 Request Level Event

The Request Level of a client can be controlled by using the following RemoteMNOPL messages

Field	Value	Description
head.m	CLIENT_REQUEST_LEVEL	Request Level (0xFE30)
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_EVENT	Request for data change.
head.l	9	8 octets for the message head and 1 octet for data.
data	E.g. 2	Request Level, 1 octet [0, 1, 2, 3]

The following structure can be used as the parameter set. The range of requestLevel is 0 to 3.

```
struct RequestLevel
{
    byte requestLevel;
}
```

8.1.2 Request Level Change

Field	Value	Description
head.m	CLIENT_REQUEST_LEVEL	Request Level
head.n	0	Don't care.
head.o	OriginID	OriginID of the server.
head.p	PK_CHANGE	Report of data change.
head.l	9	8 octets for the message head and 1 octet for data.
data	E.g. 2	Request Level of the client.

8.1.3 Querying a Request Level

Field	Value	Description
head.m	CLIENT_REQUEST_LEVEL	Request Level
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_QUERY	Query for data.
head.l	8	8 octets for the message head.

This request will be answered with one of the following replies:

8.1.4 Query Reply for a Request Level

Field	Value	Description
head.m	CLIENT_REQUEST_LEVEL	Request Level
head.n	0	Don't care.
head.o	OriginID	OriginID of the server.
head.p	PK_QUERY_REPLY	Query reply.

Field	Value	Description
head.l	9	8 octets for the message head and 1 octet for data.
data	E.g. 2	Request Level of the client.

8.1.5 Reply for Unsuccessful Request Level Query

Field	Value	Description
head.m	CLIENT_REQUEST_LEVEL	Request Level
head.n	0	Don't care.
head.o	OriginID	OriginID of the server.
head.p	PK_QUERY_NOREPLY	Query unsuccessful.
head.l	8	8 octets for the message head.

8.2 Protect Level

8.2.1 Protect Level Request Message

The Protect Level of sources and targets of the router and data areas can be controlled by using the following RemoteMNOPL messages

Field	Value	Description
head.m	DATA_PROTECT_LEVEL	Protect Level (0xFE31)
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_EVENT	Request for data change.
head.l	E.g. 15	8 octets for the message head and 4 octets of data for a CSA, 2 octets for a Message ID and 1 octet of data for the Protect Level.
data	E.g. 0x9B000200	CSA of a source, target or a data area
	E.g. MX_CONNECT	To protect a source or target from routing modifications
	E.g. 1	Protect Level

The following structure can be used as the parameter set.

messageID	Determines the protection area	
	MX_CONNECT	To protect a source or target from routing modifications
	Any Parameter Message ID	To protect a complete data area from data modifications
protectLevel	In the range of 0 to 3	

```
struct ProtectLevel
{
    CSA                address;
    unsigned short     messageID;
    byte               protectLevel;
}
```

Multiple Protect Levels can be sent within a single Protect Level Request Message. The length of the message (head.l) must be set to the appropriate value.

8.2.2 Protect Level change Reply Message

Field	Value	Description
head.m	DATA_PROTECT_LEVEL	Protect Level
head.n	0	Don't care.
head.o	OriginID	OriginID of the server.
head.p	PK_CHANGE	Report of data change.
head.l	15	8 octets for the message head and 4 octets of data for a CSA, 2 octets for a Message ID and 1 octet of data for the Protect Level.
data	E.g. 0x9B000200	CSA of a source, target or a data area
	E.g. MX_CONNECT	To protect a source or target from routing modifications
	E.g. 1	Protect Level

8.2.3 Querying a Protect Level

Field	Value	Description
head.m	DATA_PROTECT_LEVEL	Protect Level
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_QUERY	Query for data.
head.l	14	8 octets for the message head and 4 octets of data for a CSA and 2 octets for a Message ID
data	E.g. 0x9B000200	CSA of a source, target or a data area.
	E.g. MX_CONNECT	To query the Protect Level against routing modifications

This request will be answered with one of the following replies:

8.2.4 Query Reply for a Protect Level

Field	Value	Description
head.m	DATA_PROTECT_LEVEL	Protect Level
head.n	0	Don't care.
head.o	OriginID	OriginID of the server.
head.p	PK_QUERY_REPLY	Query reply.
head.l	15	8 octets for the message head and 4 octets of data for a CSA, 2 octets for a Message ID and 1 octet of data for the Protect Level.
data	E.g. 0x9B000200	CSA of a source, target or a data area.
	E.g. MX_CONNECT	The Protect Level against routing modifications
	E.g. 1	Protect Level

8.2.5 Reply for Unsuccessful Protect Level Query

Field	Value	Description
head.m	DATA_PROTECT_LEVEL	Protect Level
head.n	0	Don't care.
head.o	OriginID	OriginID of the server.
head.p	PK_QUERY_NOREPLY	Query unsuccessful.

Field	Value	Description
head.l	12	8 octets for the message head and 4 octets of data for a CSA and 2 octets for a Message ID
data	E.g. 0x9B000200	CSA of a source, target or a data area.
	E.g. MX_CONNECT	The Protect Level against routing modifications

8.3 Behaviour on protection violations

There are two kinds of protection violations:

- A client requests a Protect Level modification higher than its own Request Level ($PL > RL$)
E.g. sending a Protect Level Request Message (head.m: DATA_PROTECT_LEVEL) with a Protect Level of 3, but the Request Level of the client is 2.
- A client requests a routing or data modification and the request target is protected by a Protect Level equal or higher than the client's Request Level ($PL \geq RL$)
E.g. sending a data modification request message (e.g. head.m: MX_CONNECT) and the Protect Level of the audio target (or source) is 2, but the Request Level of the client is 2.

In both cases the message will be resent to the inquirer with a PK_ACCESS_DENIED in the message type field (head.p). The message data will remain unaffected.

9 Matrix DSP Units

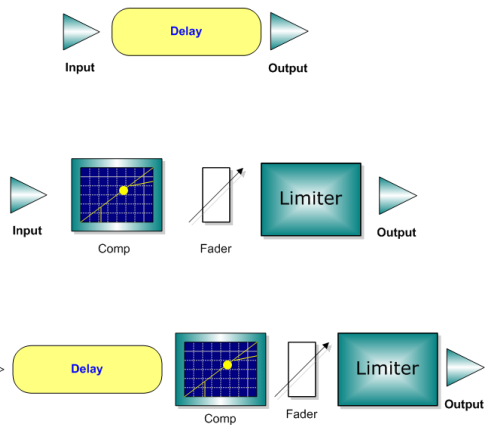
A system can have multiple matrix DSP units. These units contain various DSP modules which are combined to module chains.

The inputs and outputs of the DSP channels are internal targets and sources of the matrix and identified by CSAs. Therefore the DSP channels can be inserted into a signal path by standard connect procedures. Each parameter of the DSP modules in a DSP channel is indicated by the CSA of the DSP channel input or output signal.

When querying the device type of a DSP channel the query reply is DEVICETYPE_MXDSP_CHANNEL.

Each DSP channel can have one of the structures shown in the diagram to the right. The structure used in a DSP channel may be queried on demand similar to the device type query by a special query message of type MX_DSP_CHANNEL_MODULES. The reply is a list of unsigned short values identifying the available modules in the DSP channel (e.g. MXDSP_MODULE_LIMITER). Please refer to the information element table for a list of possible modules.

Please note: Due to the automatic generation of Matrix DSP HLSDs, the Signal Mapping feature as described in chapter 2.4 is not available for Matrix DSP signals. This means that if a client wants to control Matrix DSP settings, it currently has to use the HLSD addressing scheme (mapping mode 0).



9.1 Matrix DSP Short Delay Module

MXD_BOARD_SHORT_DELAY_TIME			
Message Type	Parameter	Comment	
PK_EVENT	<pre>struct { CSA address; TIME delay; }</pre>	address	CSA of the matrix input or output signal.
		delay	Contains the delay time in milliseconds in steps of 1/48. The range is 0 to 600 milliseconds, which yields TIME values of 0 to 28800.
PK_CHANGE	<pre>struct { CSA address; TIME delay; }</pre>	Contains the value that has been set by the hardware.	
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.	
PK_QUERY_REPLY	<pre>struct { CSA address; TIME delay; }</pre>	See PK_EVENT above.	

All further parameters are controlled similar to the delay time by using the appropriate message type in the head.m field and its control structure. Please refer to the following tables:

head.m	Parameter Set	Comments
MXD_BOARD_SHORT_DELAY_ON	<pre>struct { CSA address; byte on; }</pre>	The parameter on contains ON or OFF depending on whether the delay module is switched on or off.

9.2 Matrix DSP Fader Module

All parameters of the matrix DSP fader module units are controlled similar to the matrix DSP short delay module. A list of parameters is written down in the following table:

head.m	Parameter Set	Comments
MXD_BOARD_FADER_GAIN	<pre>struct { CSA address; LEVEL gain; }</pre>	<p>The gain parameter contains the amplification in dB in steps of 1/32.</p> <p>The range is <i>-128 to +15 dB</i>. This yields LEVEL values of <i>-4096 to +480</i>.</p>
MXD_BOARD_FADER_BYPASS	<pre>struct { CSA address; byte bypass; }</pre>	The parameter bypass contains ON or OFF depending on whether the bypass is switched on or off.
MXD_BOARD_FADER_MUTE	<pre>struct { CSA address; byte mute; }</pre>	The parameter mute contains ON or OFF depending on whether the signal is muted or not.

9.3 Matrix DSP Compressor Module

All parameters of the matrix DSP compressor module units are controlled similar to the matrix DSP short delay module. The list of parameters is written down in the following table:

head.m	Parameter Set	Comments
MXD_ BOARD_COMPRESSOR_ATTACK	struct { CSA address; TIME attack; }	The attack parameter contains the compressor attack time in milliseconds in steps of 1/48. The range is 0 to 250 milliseconds which yields TIME values of 0 to 12000.
MXD_ BOARD_COMPRESSOR_RELEASE	struct { CSA address; TIME release; }	The release parameter contains the compressor release time in milliseconds in steps of 1/48. The range is 0 to 10000 milliseconds which yields TIME values of 0 to 480000.
MXD_ BOARD_COMPRESSOR_DELAYTIME	struct { CSA address; TIME delay; }	The delay parameter contains the compressor delay time in milliseconds in steps of 1/48. The range is 0 to 10 milliseconds which yields TIME values of 0 to 480.
MXD_ BOARD_COMPRESSOR_ON	struct { CSA address; byte on; }	The parameter on contains ON or OFF depending on whether the compressor module is switched on or off.
MXD_ BOARD_COMPRESSOR_SOFTKNEE	struct { CSA address; byte softknee; }	The parameter softknee contains ON or OFF depending on whether the softknee mode is switched on or off.
MXD_ BOARD_COMPRESSOR_STEREO	struct { CSA address; byte stereo; }	The parameter stereo contains ON or OFF depending on whether the stereo mode is switched on or off.
MXD_ BOARD_COMPRESSOR_THRESHOLD	struct { CSA address; LEVEL threshold; }	The threshold parameter contains the compressor threshold in dB in steps of 1/32. The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.

head.m	Parameter Set	Comments										
MXD_BOARD_COMPRESSOR_GAIN	<pre>struct { CSA address; LEVEL gain; }</pre>	<p>The gain parameter contains the compressor gain in dB in steps of 1/32.</p> <p>The range is -20 to +20 dB which yields LEVEL values of -640 to +640.</p>										
MXD_BOARD_COMPRESSOR_RATIO	<pre>struct { CSA address; RATIO data; }</pre>	<p>The data parameter describes the compressor ratio and can be computed using the following equation:</p> <p>data = 2048 * log (ratio)</p> <p>The range is -2048 to 3898 (0.1 to 80).</p> <p>Examples:</p> <table><tr><th>data</th><th>ratio</th></tr><tr><td>- 2048</td><td>0.1</td></tr><tr><td>0</td><td>1</td></tr><tr><td>3898</td><td>80</td></tr></table>	data	ratio	- 2048	0.1	0	1	3898	80		
data	ratio											
- 2048	0.1											
0	1											
3898	80											
MXD_BOARD_COMPRESSOR_SCF_GAIN	<pre>struct { CSA address; LEVEL gain; }</pre>	<p>The gain parameter contains the compressor SCF gain in dB in steps of 1/32.</p> <p>The range is -24 to +24 dB which yields LEVEL values of -768 to +768.</p>										
MXD_BOARD_COMPRESSOR_SCF_FREQ	<pre>struct { CSA address; FREQ data; }</pre>	<p>The frequency parameter describes the compressor SCF frequency and can be computed using the following equation:</p> <p>data = 1638 * log (frequency/Hz)</p> <p>The range is 2131 to 7045 (20 Hz to 20.000 Hz).</p> <p>Examples:</p> <table><tr><th>data</th><th>frequency [Hz]</th></tr><tr><td>2131</td><td>20</td></tr><tr><td>4914</td><td>1.000</td></tr><tr><td>7045</td><td>20.000</td></tr></table>	data	frequency [Hz]	2131	20	4914	1.000	7045	20.000		
data	frequency [Hz]											
2131	20											
4914	1.000											
7045	20.000											
MXD_BOARD_COMPRESSOR_SCF_ON	<pre>struct { CSA address; byte on; }</pre>	<p>The parameter on contains ON or OFF depending on whether the compressor SCF is switched on or off.</p>										
MXD_BOARD_COMPRESSOR_SCF_TYPE	<pre>struct { CSA address; byte type; }</pre>	<p>The type parameter contains the compressor SCF type.</p> <table><tr><th>data</th><th>type</th></tr><tr><td>2</td><td>high pass</td></tr><tr><td>3</td><td>low pass</td></tr><tr><td>4</td><td>shelving high</td></tr><tr><td>5</td><td>shelving low</td></tr></table>	data	type	2	high pass	3	low pass	4	shelving high	5	shelving low
data	type											
2	high pass											
3	low pass											
4	shelving high											
5	shelving low											

9.4 Matrix DSP Limiter Module

All parameters of the Matrix DSP Limiter Module units are controlled similar to the matrix DSP short delay module. The list of parameters is written down in the following table:

head.m	Parameter Set	Comments
MXD_BOARD_LIMITER_ATTACK	<pre>struct { CSA address; TIME attack; }</pre>	<p>The attack parameter contains the limiter attack time in milliseconds in steps of 1/48.</p> <p>The range is 0 to 250 milliseconds which yields TIME values of 0 to 12000.</p>
MXD_BOARD_LIMITER_HOLD	<pre>struct { CSA address; TIME hold; }</pre>	<p>The hold parameter contains the limiter hold time in milliseconds in steps of 1/48.</p> <p>The range is 0 to 500 milliseconds which yields TIME values of 0 to 24000.</p>
MXD_BOARD_LIMITER_RELEASE	<pre>struct { CSA address; TIME release; }</pre>	<p>The release parameter contains the limiter release time in milliseconds in steps of 1/48.</p> <p>The range is 0 to 10000 milliseconds which yields TIME values of 0 to 480000.</p>
MXD_BOARD_LIMITER_DELAYTIME	<pre>struct { CSA address; TIME delay; }</pre>	<p>The delay parameter contains the limiter delay time in milliseconds in steps of 1/48.</p> <p>The range is 0 to 10 milliseconds which yields TIME values of 0 to 480.</p>
MXD_BOARD_LIMITER_ON	<pre>struct { CSA address; byte on; }</pre>	<p>The parameter on contains ON or OFF depending on whether the limiter module is switched on or off.</p>
MXD_BOARD_LIMITER_SOFTKNEE	<pre>struct { CSA address; byte softknee; }</pre>	<p>The parameter softknee contains ON or OFF depending on whether the softknee mode is switched on or off.</p>
MXD_BOARD_LIMITER_STEREO	<pre>struct { CSA address; byte stereo; }</pre>	<p>The parameter stereo contains ON or OFF depending on whether the Stereo mode is switched on or off.</p>

head.m	Parameter Set	Comments
MXD_ BOARD_LIMITER_THRESHOLD	<pre>struct { CSA address; LEVEL threshold; }</pre>	<p>The threshold parameter contains the limiter threshold in dB in steps of 1/32.</p> <p>The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.</p>
MXD_ BOARD_LIMITER_GAIN	<pre>struct { CSA address; LEVEL gain; }</pre>	<p>The gain parameter contains the limiter gain in dB in steps of 1/32.</p> <p>The range is -20 to +20 dB which yields LEVEL values of -640 to +640.</p>

9.5 Matrix DSP Parametric EQ Module

All parameters of the Matrix DSP Parametric EQ Module units are controlled similar to the matrix DSP short delay module. Additional parameters are explained in the tables below:

9.5.1 Filter Slope and Type

Slope	<p>0 = 6dB/Oct.(N=1)</p> <p>1 = 12dB/Oct.(N=2)</p> <p>2 = 18dB/Oct.(N=3)</p>
Type	<p>0 = Constant Q</p> <p>1 = Bell</p> <p>2 = High Pass</p> <p>3 = Low Pass</p> <p>4 = Shelving High</p> <p>5 = Shelving Low</p>

9.5.2 Valid types per band

Band	Types
1	0, 1, 2, 5
2	0, 1
3	0, 1
4	0, 1, 3, 4

9.5.3 Slope and Quality per Type

Type	Slope	Quality
0 = Constant Q	don't care (internally always 1 (N=2))	0.1 to 80 (* 64)
1 = Bell	don't care (internally always 1 (N=2))	0.1 to 80 (* 64)
2 = High Pass	0, 1, 2 (all)	don't care
3 = Low Pass	0, 1, 2 (all)	don't care
4 = Shelving High	0, 1, 2 (all)	don't care
5 = Shelving Low	0, 1, 2 (all)	don't care

Due to the similar handling all DSP modules, the list of parameters for the parametric EQ is documented in short form in the following table:

head.m	Comments
MXD_BOARD_PARAM_EQ_BAND_1_GAIN	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_PARAM_EQ_BAND_1_FREQ	Parameter: FREQUENCY - The parameter contains the frequency in Hz computed by data = $1638 * \log(\text{freq})$.
MXD_BOARD_PARAM_EQ_BAND_1_QUALITY	Parameter: QUALITY – The filter quality Q in steps of 1/64. See tables above.
MXD_BOARD_PARAM_EQ_BAND_1_ON	Parameter: byte - ON or OFF.
MXD_BOARD_PARAM_EQ_BAND_1_SLOPE	Parameter: byte – See tables above.
MXD_BOARD_PARAM_EQ_BAND_1_TYPE	Parameter: byte – See tables above.
MXD_BOARD_PARAM_EQ_BAND_2_GAIN	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_PARAM_EQ_BAND_2_FREQ	Parameter: FREQUENCY - The parameter contains the frequency in Hz computed by data = $1638 * \log(\text{freq})$.
MXD_BOARD_PARAM_EQ_BAND_2_QUALITY	Parameter: QUALITY – The filter quality Q in steps of 1/64. See tables above.
MXD_BOARD_PARAM_EQ_BAND_2_ON	Parameter: byte - ON or OFF.
MXD_BOARD_PARAM_EQ_BAND_2_SLOPE	Parameter: byte – See tables above.
MXD_BOARD_PARAM_EQ_BAND_2_TYPE	Parameter: byte – See tables above.
MXD_BOARD_PARAM_EQ_BAND_3_GAIN	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_PARAM_EQ_BAND_3_FREQ	Parameter: FREQUENCY - The parameter contains the frequency in Hz computed by data = $1638 * \log(\text{freq})$.
MXD_BOARD_PARAM_EQ_BAND_3_QUALITY	Parameter: QUALITY – The filter quality Q in steps of 1/64. See tables above.
MXD_BOARD_PARAM_EQ_BAND_3_ON	Parameter: byte - ON or OFF.
MXD_BOARD_PARAM_EQ_BAND_3_SLOPE	Parameter: byte – See tables above.
MXD_BOARD_PARAM_EQ_BAND_3_TYPE	Parameter: byte – See tables above.
MXD_BOARD_PARAM_EQ_BAND_4_GAIN	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_PARAM_EQ_BAND_4_FREQ	Parameter: FREQUENCY - The parameter contains the frequency in Hz computed by data = $1638 * \log(\text{freq})$.
MXD_BOARD_PARAM_EQ_BAND_4_QUALITY	Parameter: QUALITY – The filter quality Q in steps of 1/64. See tables above.
MXD_BOARD_PARAM_EQ_BAND_4_ON	Parameter: byte - ON or OFF.
MXD_BOARD_PARAM_EQ_BAND_4_SLOPE	Parameter: byte – See tables above.
MXD_BOARD_PARAM_EQ_BAND_4_TYPE	Parameter: byte – See tables above.
MXD_BOARD_PARAM_EQ_EFFECT_GAIN	Parameter: LEVEL – Level of the after EQ gain in dB in steps of 1/32.
MXD_BOARD_PARAM_EQ_ON	Parameter: byte – Switches the whole module ON or OFF.

9.6 Matrix DSP Graphic EQ Module

Due to the similar handling all DSP modules the list of parameters for the graphic EQ is documented in short form in the following table:

head.m	Comments
MXD_BOARD_GRAPHIC_EQ_BAND_1_GAIN	Parameter: LEVEL - The gain in dB in steps of 1/32.
...	
MXD_BOARD_GRAPHIC_EQ_BAND_31_GAIN	

head.m	Comments
MXD_BOARD_GRAPHIC_EQ_HIGHPASS_FREQ	Parameter: FREQUENCY - The parameter contains the frequency in Hz computed by data = $1638 * \log(\text{freq})$.
MXD_BOARD_GRAPHIC_EQ_LOWPASS_FREQ	Parameter: FREQUENCY - The parameter contains the frequency in Hz computed by data = $1638 * \log(\text{freq})$.
MXD_BOARD_GRAPHIC_EQ_NOTCH_1_FREQ	Parameter: FREQUENCY - The parameter contains the frequency in Hz computed by data = $1638 * \log(\text{freq})$.
MXD_BOARD_GRAPHIC_EQ_NOTCH_1_QUALITY	Parameter: QUALITY – The filter quality Q in steps of 1/64. See tables above.
MXD_BOARD_GRAPHIC_EQ_NOTCH_2_FREQ	Parameter: FREQUENCY - The parameter contains the frequency in Hz computed by data = $1638 * \log(\text{freq})$.
MXD_BOARD_GRAPHIC_EQ_NOTCH_2_QUALITY	Parameter: QUALITY – The filter quality Q in steps of 1/64. See tables above.
MXD_BOARD_GRAPHIC_EQ_EFFECT_GAIN	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_GRAPHIC_EQ_ON	Parameter: byte - ON or OFF.
MXD_BOARD_GRAPHIC_EQ_TRUECURVE_ON	Parameter: byte - ON or OFF.
MXD_BOARD_GRAPHIC_EQ_HIGHPASS_ON	Parameter: byte - ON or OFF.
MXD_BOARD_GRAPHIC_EQ_LOWPASS_ON	Parameter: byte - ON or OFF.
MXD_BOARD_GRAPHIC_EQ_NOTCH1_ON	Parameter: byte - ON or OFF.
MXD_BOARD_GRAPHIC_EQ_NOTCH2_ON	Parameter: byte - ON or OFF.

9.7 Matrix DSP Gate Module

Due to the similar handling all DSP modules the list of parameters for the gate module is documented in short form in the following table:

head.m	Comments
MXD_BOARD_GATE_ATTACK	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_GATE_HOLD	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_GATE_RELEASE	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_GATE_DELAYTIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_GATE_ON	Parameter: byte - ON or OFF.
MXD_BOARD_GATE_STEREO	Parameter: byte - ON or OFF.
MXD_BOARD_GATE_THRESHOLD	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_GATE_FLOOR	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_GATE_HYSTERESIS	Parameter: LEVEL - The gain in dB in steps of 1/32.

9.8 Matrix DSP Automatic Gain Control Module

Due to the similar handling all DSP modules the list of parameters for the AGC module is documented in short form in the following table:

head.m	Comments
MXD_BOARD_AGC_ATTACK	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.

head.m	Comments
MXD_BOARD_AGC_HOLD	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_AGC_RELEASE	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_AGC_DELAYTIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_AGC_ON	Parameter: byte - ON or OFF.
MXD_BOARD_AGC_STEREO	Parameter: byte - ON or OFF.
MXD_BOARD_AGC_EXP_THRESHOLD	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_AGC_EXP_RATIO	Parameter: RATIO - The parameter contains the ratio computed by data = $2048 * \log(\text{ratio})$.
MXD_BOARD_AGC_COMP_ROT_THRES	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_AGC_COMP_RATIO	Parameter: RATIO - The parameter contains the ratio computed by data = $2048 * \log(\text{ratio})$.
MXD_BOARD_AGC_MAX_GAIN	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_AGC_MIN_GAIN	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_AGC_SCF_GAIN	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_AGC_SCF_FREQ	Parameter: FREQUENCY - The parameter contains the frequency in Hz computed by data = $1638 * \log(\text{freq})$.
MXD_BOARD_AGC_SCF_ON	Parameter: byte - ON or OFF.
MXD_BOARD_AGC_SCF_TYPE	Parameter: byte – See table below.

9.8.1 Side Chain Filter Types

Type	Slope	Quality
2 = High Pass	don't care (internal always 0 (N=1))	don't care
3 = Low Pass	don't care (internal always 0 (N=1))	don't care
4 = Shelving High	don't care (internal always 0 (N=1))	don't care
5 = Shelving Low	don't care (internal always 0 (N=1))	don't care

9.9 Matrix DSP Input Mixer

Due to the similar handling all DSP modules the list of parameters for the input mixer module is documented in short form in the following table:

head.m	Comments
MXD_BOARD_INPUT_MIXER_GAIN	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_INPUT_MIXER_BALANCE	Parameter: RANGE – Contains -20 to 20 (from left to right)
MXD_BOARD_INPUT_MIXER_ON	Parameter: byte - ON or OFF.
MXD_BOARD_INPUT_MIXER_PHASE	Parameter: byte - ON or OFF.
MXD_BOARD_INPUT_MIXER_SIDE_EXCHANGE	Parameter: byte - ON or OFF.
MXD_BOARD_INPUT_MIXER_LEFT_TO_BOTH	Parameter: byte - ON or OFF.
MXD_BOARD_INPUT_MIXER_RIGHT_TO_BOTH	Parameter: byte - ON or OFF.
MXD_BOARD_INPUT_MIXER_MS	Parameter: byte - ON or OFF.
MXD_BOARD_INPUT_MIXER_MONO_TO_BOTH	Parameter: byte - ON or OFF.
MXD_BOARD_INPUT_MIXER_STEREO	Parameter: byte - ON or OFF.

9.10 Matrix DSP Timed Fader

Due to the similar handling all DSP modules the list of parameters for the input mixer module is documented in short form in the following table:

head.m	Comments
MXD_BOARD_TIMED_FADER_GAIN	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_TIMED_FADER_BYPASS	Parameter: byte - ON or OFF.
MXD_BOARD_TIMED_FADER_MUTE	Parameter: byte - ON or OFF.
MXD_BOARD_TIMED_FADER_FADE_BY_TIME	Parameter: byte - ON or OFF.
MXD_BOARD_TIMED_FADER_MUTE_BY_TIME	Parameter: byte - ON or OFF.
MXD_BOARD_TIMED_FADER_STEREO	Parameter: byte - ON or OFF.
MXD_BOARD_TIMED_FADER_FADE_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_TIMED_FADER_MUTE_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_TIMED_FADER_UNMUTE_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_TIMED_FADER_UNMUTE_DELAY_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.

9.11 Matrix DSP Signal Condition Monitor

Due to the similar handling all DSP modules the list of parameters for the input mixer module is documented in short form in the following table:

head.m	Comments
MXD_BOARD_SIGNAL_CONDITION_MONITOR_OVER_DURATION_ALARM_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_OVER_ALARM_HOLD_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_UNDER_DURATION_ALARM_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_UNDER_ALARM_HOLD_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_SILENCE_DURATION_ALARM_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_SILENCE_ALARM_HOLD_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_CORRELATION_DURATION_ALARM_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_CORRELATION_ALARM_HOLD_TIME	Parameter: TIME - The parameter contains the time in milliseconds in steps of 1/48.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_OVER_THRESHOLD	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_UNDER_THRESHOLD	Parameter: LEVEL - The gain in dB in steps of 1/32.

head.m	Comments
MXD_BOARD_SIGNAL_CONDITION_MONITOR_SILENCE_THRESHOLD	Parameter: LEVEL - The gain in dB in steps of 1/32.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_CORRELATION_THRESHOLD	Parameter: RANGE – Contains -100 (out-of-phase) to 100 (in-phase); 0 means incoherent signals.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_ALARM_ENABLE	Parameter: byte - ON or OFF.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_OVER_ALARM_ENABLE	Parameter: byte - ON or OFF.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_UNDER_ALARM_ENABLE	Parameter: byte - ON or OFF.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_SILENCE_ALARM_ENABLE	Parameter: byte - ON or OFF.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_CORRELATION_ALARM_ENABLE	Parameter: byte - ON or OFF.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_STEREO	Parameter: byte - ON or OFF.
MXD_BOARD_SIGNAL_CONDITION_MONITOR_LEVEL_ALARM_STATE	Parameter: 0 = Target range 1 = Silence 2 = Under Range 3 = Over Range
MXD_BOARD_SIGNAL_CONDITION_MONITOR_CORRELATION_ALARM	Parameter: byte - ON or OFF.

9.12 Matrix DSP Channel Metering

Note: Due to the required processing power, this functionality requires a matrix control server (router modules 980/31 and 980/32) or a router module type 980/33. The total number of metering subscriptions (level metering and/or correlation metering) is limited to 512 mono channels per client. The maximum number of metering channels per system is 1024 since at most four matrix-DSP-boards with DSP configurations containing metering may be used.

Only five clients may subscribe to metering data at the same time. Subscription requests of further clients will be rejected.

MXD_BOARD_METERING		
Message Type	Parameter	Comment
PK_EVENT	None	Not implemented.
PK_CHANGE	<pre>struct { CSA address; LEVEL meter_level; }</pre>	<p>The metering_level contains the signal level in dB in steps of 1/32.</p> <p>The range is -128 to 40 dB. This yields LEVEL values of -4096 to +1280.</p> <p>The metering information for all channels will be updated periodically.</p>
PK_QUERY	None	Not implemented.
PK_QUERY_REPLY	None	Not implemented.

MXD_BOARD_ADD_METERING		
Message Type	Parameter	Comment
PK_EVENT	[Sequence of] CSAs	Selection of DSP channels for which the client wishes to receive metering information. The channels will be added to a client specific list of metered DSP channels in the server. The maximum number of channels to meter simultaneously is 128.
PK_CHANGE	[Sequence of] CSAs	Includes the CSAs of all metering enabled DSP channels.
PK_QUERY	None	Query for all signals of the client specific list in the server.
PK_QUERY_REPLY	[Sequence of] CSAs	Includes all signals of the client specific list in the server.

MXD_BOARD_REMOVE_METERING		
Message Type	Parameter	Comment
PK_EVENT	[Sequence of] CSAs	Selection of DSP channels the client wishes to remove from the list of metered DSP channels.
PK_CHANGE	[Sequence of] CSAs	Includes the CSAs of the DSP channels that have been removed successfully from the list of metered channels.
PK_QUERY	None	Not implemented.
PK_QUERY_REPLY	None	Not implemented.

9.13 Matrix DSP Correlation Metering

MXD_BOARD_CORRELATION_METERING		
Message Type	Parameter	Comment
PK_EVENT	None	Not implemented.
PK_CHANGE	<pre>struct { CSA address; RANGE degree; }</pre>	<p>The degree contains the degree of correlation in %.</p> <p>The range is -100 to +100.</p> <p>The metering information for all channels in the list of metered DSP channels will be updated periodically.</p>
PK_QUERY	None	Not implemented.
PK_QUERY_REPLY	None	Not implemented.

MXD_BOARD_ADD_CORRELATION_METERING		
Message Type	Parameter	Comment
PK_EVENT	[Sequence of] CSAs	Selection of even outputs of DSP channels for which the client wishes to receive correlation information. The HLSDs will be added to a client specific list in the server. e.g. 0x730b0000, 730b0002, ...
PK_CHANGE	[Sequence of] CSAs	Includes the CSAs of all signals added to the client specific list in the server.
PK_QUERY	None	Query for all signals of the client specific list in the server.
PK_QUERY_REPLY	[Sequence of] CSAs	Includes all signals of the client specific list in the server.

MXD_BOARD_REMOVE_CORRELATION_METERING		
Message Type	Parameter	Comment
PK_EVENT	[Sequence of] CSAs	Selection of even outputs of DSP channels which the client wishes to remove from the specific list in the server. e.g. 0x730b0000, 730b0002, ...
PK_CHANGE	[Sequence of] CSAs	Includes the CSAs of the DSP channels that have been removed successfully from the list of metered channels.
PK_QUERY	None	Not implemented.
PK_QUERY_REPLY	None	Not implemented.

Note again: The total number of metering subscriptions (level metering and/or correlation metering) is limited to 512 mono channels per client. The maximum number of metering channels per system is 1024 since at most four matrix-DSP-boards with DSP configurations containing metering may be used.

Only five clients may subscribe to metering data at the same time. Subscription requests of further clients will be rejected.

9.14 Matrix DSP Mixing Matrix

The Matrix DSP boards may contain mixing units similar to the mixing units of I/O boards depending on the configuration. The appropriate input and output signals may be identified by the device type query (IO_DEVICE_TYPE). The query result will be DEVICETYPE_MXDSP_SUM_IN for the inputs of the matrix DSP mixing units (target CSAs from the matrix point of view) and DEVICETYPE_MXDSP_SUM_OUT for the outputs (source CSAs for the matrix). The unit size can be queried with the ER_UNIT_INFO message. The ER_UNIT_INFO query delivers the dimension of the mixing unit (64), the ID of the unit in the matrix system and the ID of the channel within the mixing unit per CSA.

head.m	Parameter Set	Comments
MXD_BOARD_SUM_IN_GAIN	<pre>struct { CSA address; LEVEL gain; }</pre>	<p>The gain parameter contains the amplification in dB in steps of 1/32.</p> <p>The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.</p>
MXD_BOARD_SUM_IN_MUTE	<pre>struct { CSA address; byte mute; }</pre>	<p>The parameter mute contains ON or OFF depending on whether mute is switched on or off.</p>
MXD_BOARD_SUM_IN_PHASE	<pre>struct { CSA address; byte invPhase; }</pre>	<p>The parameter invPhase contains ON or OFF depending on whether phase inversion is used or not.</p>
MXD_BOARD_SUM_OUT_GAIN	<pre>struct { CSA address; LEVEL gain; }</pre>	<p>The gain parameter contains the amplification in dB in steps of 1/32.</p> <p>The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.</p>

head.m	Parameter Set	Comments
MXD_BOARD_SUM_OUT_MUTE	<pre>struct { CSA address; byte mute; }</pre>	The parameter mute contains ON or OFF depending on whether mute is switched on or off.
MXD_BOARD_SUM_OUT_PHASE	<pre>struct { CSA address; byte invPhase; }</pre>	The parameter invPhase contains ON or OFF depending on whether phase inversion is used or not.
MXD_BOARD_SUM_OUT_XPOINT_GAIN	<pre>struct { CSA address; LEVEL gain; }</pre>	<p>The index of the mixing unit input is determined in the head.n field in the range of 0 to 63.</p> <p>The gain parameter contains the amplification in dB in steps of 1/32.</p> <p>The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480.</p>
MXD_BOARD_SUM_OUT_XPOINT_SET	<pre>struct { CSA address; byte crosspointSet; }</pre>	<p>The index of the mixing unit input is determined in the head.n field in the range of 0 to 63.</p> <p>The crosspointSet parameter contains ON or OFF depending on whether the given input is used or not.</p>
MXD_BOARD_SUM_OUT_XPOINT_PHASE	<pre>struct { CSA address; byte invPhase; }</pre>	<p>The index of the mixing unit input is determined in the head.n field in the range of 0 to 63.</p> <p>The invPhase parameter contains ON or OFF depending on whether phase inversion is used or not.</p>

10 DSP Parameter Control (valid for mc² series only)

Please note: Due to the automatic generation of DSP HLSDs, the Signal Mapping feature as described in chapter 2.4 is not available for DSP signals. This means that if a client wants to control DSP settings, it currently has to use the HLSD addressing scheme (mapping mode 0).

10.1 DSP Main Level

EU_DSP_MAINLEVEL			
Message Type	Parameter	Comment	
PK_EVENT	<pre>struct { CSA address; LEVEL amp; }</pre>	The main level of a DSP channel can be changed by sending a new amplification value.	
		address	The CSA of the DSP's direct out signal.
		amp	Contains the amplification in dB in steps of 1/32. The range is -128 to +15 dB. This yields LEVEL values of -4096 to +480. If you try to set a value out of the range the hardware is capable to, the corresponding PK_CHANGE message will contain the value that has really been set.
PK_CHANGE	<pre>struct { CSA address; LEVEL amp; }</pre>	Contains the value that has been set by the hardware.	
PK_QUERY	CSA	The parameter query is referenced by the CSA of the audio signal.	
PK_QUERY_REPLY	<pre>struct { CSA address; LEVEL amp; }</pre>	See PK_EVENT above.	

For controlling the DSP main level the direct out signal address of the DSP channel will be used. HLSDs for the direct out signal are built using the following values:

Signalclass	Subclass	Component	Subcomponent
SCL_DIROUT (0x01)	FU_INPUT (0x00)	0..255	0..2
	FU_MONITOR (0x01)	0..255	0..2
	FU_GROUP (0x02)	0..191	0
	FU_SUMM (0x03)	0..191	0
	FU_AUX (0x04)	0..31	0
	FU_MASTER (0x05)	0..127	0
	FU_GPC (0x06)	0..255	0
	FU_SURROUND_MASTER (0x09)	0..255	0..1

These values represent the current maximum ranges. The real number of available DSP units depends on the system and the DSP usage configuration. Please note that the high byte of the signal index is represented in the Subcomponent value if the index exceeds 255, e.g. Surround Master 256 has the HLSD 0x01090001 or Input 514 has the HLSD 0x01000202.

10.2 DSP Channel Cut

DSP channel cut is working similar to DSP main level by using EU_CUT_CUT for head.m with the following data struct in the message body. Cut can be switched ON or OFF.

```
struct
{
    CSA    address;
    KEYS   cut;
}
```

10.3 DSP Panpot Balance

DSP panpot balance is working similar to DSP main level by using EU_PANPOT_PANBAL for head.m with the following data struct in the message body. The range of PanpotBalance is -20 to 20.

```
struct
{
    CSA    address;
    RANGE  PanpotBalance;
}
```

10.4 DSP Panpot Frontback

DSP panpot frontback is working similar to DSP main level by using EU_PANPOT_FRONTBACK for head.m with the following data struct in the message body. The range of FrontBack is -20 to 20.

```
struct
{
    CSA    address;
    RANGE  FrontBack;
}
```

10.5 DSP Access Channel

The current mixing console access channel can be modified and queried using the EC_ACCESS1 command. The command syntax and signal addressing scheme are again similar to the DSP main level control feature. The data part of the command (when sending an event or receiving a query reply) just consists of a CSA value. The query command just consists of a header and has no data attached.

10.6 DSP Audio Follows Video

All parameters of Audio Follows Video are controlled similar to the DSP Main Level parameter. The list of parameters is written down in the following table:

head.m	Parameter Set	Comments
AFV_ENABLED	<pre>struct { CSA address; KEYS enable; }</pre>	This enables the Audio Follows Video functionality.

AFV_EVENTNUMBER	<pre>struct { CSA address; STATE8 eventno; }</pre>	One of up to 128 events can be assigned for AFV functions.
AFV_ONLEVEL	<pre>struct { CSA address; LEVEL level; }</pre>	<p>This is the level to which the fader will open when triggered by the AFV event. If AFV is in ON state, this parameter will also control the DSP Main Level.</p> <p>The level may be adjusted from -128dB to +15dB which yields LEVEL values from -4096 to +480.</p>
AFV_OFFLEVEL	<pre>struct { CSA address; LEVEL level; }</pre>	<p>This is the level to which the fader will be reset as soon as the event is switched off.</p> <p>If AFV is in OFF state, this parameter will also control the DSP Main Level.</p> <p>The level may be adjusted from -128dB to +15dB which yields LEVEL values from -4096 to +480.</p>
AFV_HOLDTIME	<pre>struct { CSA address; TIME time; }</pre>	<p>The HOLD TIME delays the opening of the fader after the event on trigger.</p> <p>The time may be adjusted in steps of 1/48 from 0 to 10 seconds which yields TIME values from 0 to 480000.</p>
AFV_RISETIME	<pre>struct { CSA address; TIME time; }</pre>	<p>The RISE TIME sets the time taken for the fader to move from off to on level after the hold time has expired.</p> <p>The time may be adjusted in steps of 1/48 from 0 to 10 seconds which yields TIME values from 0 to 480000.</p>
AFV_MAXEVENTTIME	<pre>struct { CSA address; TIME time; }</pre>	<p>The MAX EVENT TIME sets the time the fader will stay open even if the event is not switched off.</p> <p>The time may be adjusted in steps of 1/48 from 0 to 10 seconds which yields TIME values from 0 to 480000.</p>
AFV_ONTIME	<pre>struct { CSA address; TIME time; }</pre>	<p>The ON TIME sets the time the fader stays at the on level after the event has been switched off.</p> <p>The time may be adjusted in steps of 1/48 from 0 to 10 seconds which yields TIME values from 0 to 480000.</p>
AFV_FALLTIME	<pre>struct { CSA address; TIME time; }</pre>	<p>The FALL TIME sets the time for the fader to move from on to off level.</p> <p>The time may be adjusted in steps of 1/48 from 0 to 10 seconds which yields TIME values from 0 to 480000.</p>

Symbolic head.m Name	Hex Value	Type
AFV_ONLEVEL	0x19C1	LEVEL
AFV_OFFLEVEL	0x19C2	LEVEL
AFV_RISETIME	0x19C3	TIME
AFV_ONTIME	0x19C4	TIME
AFV_FALLTIME	0x19C5	TIME
AFV_EVENTNUMBER	0x19C6	STATE8
AFV_ENABLED	0x19C8	KEYS
AFV_HOLDTIME	0x19CA	TIME
AFV_MAXEVENTTIME	0x19CB	TIME

10.7 DSP PFL On/Off, PFL 1 Clear, Aux Send On/Off, PEQ, PF, AF

All parameters of Pre Fader Listening are controlled similar to the DSP Main Level parameter. The list of parameters is written down in the following table:

head.m	Parameter Set	Comments												
EU_PFL	<pre>struct { CSA address; KEYS on; }</pre>	<p>Routes a signal to the PFL 1 bus depending on the parameter “on” which can be ON or OFF.</p> <p>Please note the size of 2 bytes for “on”.</p>												
EU_CLEAR_PFL	No Data	This command clears all PFL 1 routings.												
EU_AUX_ON	<pre>struct { CSA address; KEYS on; }</pre>	<p>Switches the Aux Send on or off depending on the parameter “on” which can be ON or OFF.</p> <p>Please note the size of 2 bytes for “on”.</p> <p>The index of the Aux Send is determined by the head.n field.</p> <p>The valid range is 0 to 31 for the Aux Send 1 to 32.</p>												
EU_AUX_SOURCE_SIGNAL (0x1320)	<pre>struct { CSA address; byte source; }</pre>	<p>Controls the Aux Sends Source Signal in the channel area.</p> <p>Valid sources are</p> <table><tr><td>Pre Equalizer</td><td>10</td></tr><tr><td>Pre Fader</td><td>37</td></tr><tr><td>After Fader</td><td>46</td></tr></table> <p>Please note that different values might occur on monitor channels 17 to 32 if “Cue Aux Send/Return” is active. If this is the case, the client is allowed to switch between the following sources only:</p> <table><tr><td>Aux Cue</td><td>47</td></tr><tr><td>Tape Return</td><td>48</td></tr><tr><td>Tape Send</td><td>49</td></tr></table> <p>The index of the Aux Send is determined by the head.n field.</p> <p>The valid range to 0 to 31 for the Aux Sends 1 to 32.</p>	Pre Equalizer	10	Pre Fader	37	After Fader	46	Aux Cue	47	Tape Return	48	Tape Send	49
Pre Equalizer	10													
Pre Fader	37													
After Fader	46													
Aux Cue	47													
Tape Return	48													
Tape Send	49													

11 Snapshots

The snapshot feature offers the possibility to save and recall many system settings within a so-called “production”. While productions additionally contain many configuration parameters, snapshots are limited to operational parameters such as routings, mixing console channel settings, mixing console desk assignments, user labels etc.

Please note that loading or saving productions is not yet available in Remote MNOPL.

11.1 Loading and Saving Snapshots

Remote MNOPL clients can subscribe to the MX_LOAD_SNAPSHOT or the MX_SAVE_SNAPSHOT stream in order to receive system notifications when a snapshot has been recalled or saved. Currently, the system does not support querying the active snapshot, i.e. a resynchronization after connection loss has to be done by loading a defined snapshot, saving a snapshot or waiting for the next MX_LOAD_SNAPSHOT or MX_SAVE_SNAPSHOT change.

Note: Saving a snapshot usually replaces the currently active snapshot by the newly created (saved) snapshot.

The data structure used for the snapshot folder and name is the SnapshotPath structure which assembles like follows:

```
struct SnapshotPath
{
    string[32] folder;
    string[32] filename;
}
```

Please note that the **folder and filename strings have to be null-terminated**. Therefore the folder name and the filename are limited to 31 characters. Please also note that the system uses a latin1 encoding for file transfers. This is especially important when dealing with German umlauts or other “special” characters.

The client is also allowed to load or save snapshots by sending MX_LOAD_SNAPSHOT or MX_SAVE_SNAPSHOT events:

Field	Value	Description
head.m	MX_LOAD_SNAPSHOT	Load Snapshot.
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_EVENT	Request for data change.
head.l	8 + 32 + 32 = 72	8 octets for the message head, 32 octets for the snapshot folder and 32 octets for the snapshot filename
data	string[32]	null-terminated 31 characters snapshot folder name
	string[32]	null-terminated 31 characters snapshot name

Note: If the snapshot folder or file cannot be located, the system will not respond with a snapshot change or any error message. The request will simply be ignored. As already mentioned above, to receive notifications when loading or saving snapshots completed successfully the respective head.m stream has to be subscribed (OPEN_STREAM message).

Saving snapshots is performed using the same message structure already described above except for the head.m field which has to be replaced by MX_SAVE_SNAPSHOT.

Here is a message sample which tries to load the snapshot “snap0001” from the folder “TestFolder”:

Index	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	...
Hex	FE	00	00	00	20	02	00	48	54	65	73	74	46	6F	6C	64	65	72	00	x	x	...
Chr									T	e	s	t	F	o	l	d	e	r	\0			

Index	...	40	41	42	43	44	45	46	47	48	49	50	...	70	71
Hex	...	73	6E	61	70	30	30	30	31	00	x	x	...	x	x
Chr		s	n	a	p	0	0	0	1	\0					

11.2 Using Snapshot Filters

When working with complete snapshots of all available operational parameters, users sometimes need the possibility to exclude some of the system’s aspects from the modification by loading the snapshot. Regarding the mixing console appliance, the Lawo system offers two major possibilities to prevent settings from snapshot modifications:

1. Isolation of DSP channels on a per-channel basis.
2. Global snapshot filters.

This chapter deals with the second possibility: the global snapshot filters.

The available filters are:

- The CONNECTS snapshot filter (MX_SNAPSHOT_FILTER_CONNECTS) which prevents matrix routings from being modified by the snapshot.
- The IO snapshot filter (MX_SNAPSHOT_FILTER_IO) which prevents IO settings (such as SRC, Microphone Preamp parameters etc.) from being modified by the snapshot.
- The LABELS snapshot filter (MX_SNAPSHOT_FILTER_LABELS) which prevents all labels from being modified by the snapshot.
- The MX_DSP snapshot filter (MX_SNAPSHOT_FILTER_MX_DSP) which prevents Matrix DSP card settings from being modified by the snapshot.
- The DESK snapshot filter (MX_SNAPSHOT_FILTER_DESK) which prevents the console from being modified by the snapshot.
- The DSP snapshot filter (MX_SNAPSHOT_FILTER_DSP) which prevents mixing console channels from being modified by the snapshot.

The filters can be queried and modified using the corresponding PK_EVENT or PK_QUERY messages.

Field	Value	Description
head.m	MX_SNAPSHOT_FILTER_CONNECTS	Specifies the snapshot filter that shall be modified, e.g. the CONNECTS snapshot filter. Other head.m values can be found in the filter description above.
head.n	0	Don’t care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_EVENT	Request for data change.
head.l	9	8 octets for the message head and 1 octet for the state value.
data	byte: 0 or 1	Switches the filter OFF or ON.

Please remember that the client has to subscribe to the respective snapshot filter stream in order to receive change notifications.

To query the state of a snapshot filter, please use the PK_QUERY message without any data – the PK_QUERY_REPLY response from the system will contain one data byte describing the filter state (0 = off, 1 = on).

Field	Value	Description
head.m	MX_SNAPSHOT_FILTER_CONNECTS	Specifies the snapshot filter that shall be queried, e.g. the CONNECTS snapshot filter. Other head.m values can be found in the filter description above.
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_QUERY	Query for data.
head.l	8	8 octets for the message head.

11.3 Working with Snapshot Isolate

DSP Channel settings in mixing console appliances can be saved from being overwritten by a snapshot using the snapshot isolate feature. The control messages for this feature are implemented with a different addressing scheme which is intended to be used for further DSP related controls in future. The scheme provides functional unit indices which have to be used in the MNOPL header "N" field. The addresses of the functional units can be taken from the following table.

Functional Unit	Address [From ... To]	Address Hex [From ... To]
DSP Input	0 ... 767	0x0 ... 0x2FF
DSP Monitor	768 ... 1535	0x300 ... 0x5FF
DSP Group	1536 ... 1727	0x600 ... 0x6BF
DSP Sum	1728 ... 1919	0x6C0 ... 0x77F
DSP Aux	1920 ... 1951	0x780 ... 0x79F
DSP Master	1952 ... 2079	0x7A0 ... 0x81F
DSP General Purpose Channel	2080 ... 2335	0x820 ... 0x91F
DSP Surround Master	2416 ... 2927	0x970 ... 0xB6F

Currently, the number of available DSP units has to be taken from the available CSAs of their Direct Out signals (please refer to chapter 10.1). A change of the mixing console's DSP preset will not be reflected to the Remote MNOPL interface, i.e. the controller has to be reset to re-query all available CSAs. This behaviour is subject to change in future versions.

The control command for setting the Snapshot Isolate flag is defined as follows:

Field	Value	Description
head.m	SNAP_ISO_CHANNEL	Isolate command.
head.n	Functional Unit Address (see above)	Address of the functional DSP unit.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_EVENT	Request for data change.
head.l	9	8 octets for the message head and 1 octet of data.
data	Values 0 or 1 as byte	Status of Snapshot Isolate flag (0 = inactive, 1 = active)

The corresponding system response when a flag change is requested uses the same message structure with the difference that the head.p field is changed to PK_CHANGE.

The status of the Snapshot Isolate flag can be queried using the following command:

Field	Value	Description
head.m	SNAP_ISO_CHANNEL	Isolate query.
head.n	Functional Unit Address (see above)	Address of the functional DSP unit.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_QUERY	Query for data.
head.l	8	8 octets for the message head.

The response is similar to the PK_CHANGE message. Again, the head.p field is changed, for replies to PK_QUERY_REPLY. If the queried address is invalid or if the status cannot be determined the server will send a PK_QUERY_NOREPLY message:

Field	Value	Description
head.m	SNAP_ISO_CHANNEL	Isolate query.
head.n	Functional Unit Address (see above)	Address of the functional DSP unit.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_QUERY_NOREPLY	Query reply.
head.l	8 or 9	8 octets for the message head, 1 byte data (on some server versions).

Please note that some server versions send a data byte (value 0) in the QUERY_NOREPLY message which has to be ignored by the client.

12 Access Channel Presets (valid for mc² series only)

RemoteMNOPL clients may modify the current mixing console access channel by sending a binary Channel Preset. Such Presets can be queried from the system as binary dumps containing the whole mixing console channel related settings like dynamics, eq, upmix, bus assignment etc. The contents of the preset are identical compared to the contents of Channel Presets written by the Lawo GUI application.

If the client wishes to save a preset of another channel than the current access channel, it is suggested to query the current access channel (cf- chapter 10.5), change it temporarily to the channel to be saved, save the preset and reset the access channel to the previous value. This also yields for recalling presets on channels different from the current access channel.

12.1 Querying a Channel Preset

The current access channel settings can be queried using the following telegram message:

Field	Value	Description
head.m	GET_ACCESS_PRESET	Telegram identifier for the access channel preset query.
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_QUERY	Query for data.
head.l	8	8 octets for the message head.

Note: Nova73 setups do not support Channel Presets. Hence queries for the preset will not be successful.

The query will be replied with a GET_ACCESS_PRESET PK_QUERY_REPLY with the following structure:

Field	Value	Description
head.m	GET_ACCESS_PRESET	Telegram identifier for the access channel preset query.
head.n	0	Don't care.
head.o	Server OriginID	The server's origin ID. Usually 0.
head.p	PK_QUERY_REPLY	Query Reply.
head.l	8 + x	8 octets for the message head and an arbitrary amount of octets for binary data.
data	<binary>	Contains a binary dump of the current access channel settings.

The binary data can be stored and recalled at any time using the SET_ACCESS_PRESET telegram.

12.2 Recalling a Channel Preset

Saved access channel preset can be recalled using the SET_ACCESS_PRESET telegram. Please be sure to send data which was created using the GET_ACCESS_PRESET feature only due to the fact that the binary data validity checks will not always be successful and the system can be harmed by sending wrong contents accidentally.

Field	Value	Description
head.m	SET_ACCESS_PRESET	Telegram identifier for the access channel preset recall.
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_EVENT	Request for data change.
head.l	8 + x	8 octets for the message head and an arbitrary amount of octets for binary data.
data	<binary>	Contains a binary dump saved with the GET_ACCESS_PRESET feature.

Please be aware that the system currently does not support sending changes (i.e. confirmations) when recalling presets. Therefore, there is also no stream subscription for the SET_ACCESS_PRESET feature.

13 Working with Routings

To work with routings, the client has to subscribe to the MX_CONNECT data stream at the server. If this stream is enabled the client will get any changes concerning routings (PK_CHANGE messages) the server is sending to any other client. This way it can automatically stay in sync with the routing entity in the server. If the server creates a routing for a client, the appropriate information is mirrored to all clients automatically.

Note: If the data field contains definitions of unknown signals the client does not get any answer. This is usually a configuration problem. To prevent from such problems, the dynamic client configuration should be used.

13.1 Basic Routing Messages

13.1.1 Routing Request Message

Routings are done by sending the following event:

Field	Value	Description
head.m	MX_CONNECT	Matrix routing.
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_EVENT	Request for data change.
head.l	E.g. 16	8 octets for the message head and 8 octets of data for 2 CSAs (source and target)
data	E.g. 0x23000100	CSA of an audio source
	E.g. 0x9B000200	CSA of an audio target

The contained source CSA will be connected to the subsequent target CSA.

Multiple source target pairs can be sent within a single routing request. The length of the message (head.l) must be set to the appropriate value. To disconnect a target it must be routed to the special source CSA "NO_SIGNAL" (0xFFFFFFFF).

13.1.2 Routing Reply Message

The server would reply to the routing request message in the example above with:

Field	Value	Description
head.m	MX_CONNECT	Matrix routing.
head.n	0	Don't care.
head.o	OriginID	OriginID of the server.
head.p	PK_CHANGE	Report of data change.
head.l	16	8 octets for the message head and 8 octets of data for 2 CSAs (source and target).
data	0x23000100	CSA of an audio source.
	0x9B000200	CSA of an audio target.

Multiple source target pairs can be sent within a routing reply. The length of the message (head.l) is set to the appropriate value. If the special CSA "NO_SIGNAL" (0xFFFFFFFF) is reported as source the target is not connected.

If the special CSA "UNKNOWN_SIGNAL" (0xFFFEFFFF) is reported in either the source or the target field, this signal is not mentioned in the current configuration. Unknown signals are locally defined signals which are not available for the control system that is receiving this message.

The system is sending a reply even if the requested routing already existed.

13.1.3 Querying a Routing

The source connected to a target can be queried by an MX_CONNECT message with packet type PK_QUERY.

Field	Value	Description
head.m	MX_CONNECT	Matrix routing.
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_QUERY	Query for data.
head.l	12	8 octets for the message head and 4 octets of data for a CSA (target).
data	E.g. 0x9B000200	CSA of the audio target.

This request will be answered with one of the following replies:

13.1.4 Query Reply for a Routing

Field	Value	Description
head.m	MX_CONNECT	Matrix routing.
head.n	0	Don't care.
head.o	OriginID	OriginID of the server.
head.p	PK_QUERY_REPLY	Query reply.
head.l	16	8 octets for the message head and 8 octets of data for two CSAs (source and target).
data	E.g. 0x23000100	CSA of the audio source.
	E.g. 0x9B000200	CSA of the audio target.

13.1.5 Reply for Unsuccessful Routing Query

Field	Value	Description
head.m	MX_CONNECT	Matrix routing.
head.n	0	Don't care.
head.o	OriginID	OriginID of the server.
head.p	PK_QUERY_NOREPLY	Query unsuccessful.
head.l	12	8 octets for the message head and 4 octets of data for a CSA.
data	E.g. 0x9B000200	CSA of the audio target.

13.2 Static Routing

As of version 4.12 of the Remote MNOPL protocol static routings are supported. The static routing flag is a target flag and if set, it prevents from overwriting a routing accidentally. As long as a target is "protected", all routing requests concerning this target are rejected (without notification).

13.2.1 Static Routing Request

The routing can be declared static using the following message:

Field	Value	Description
head.m	MX_TARGET_STATIC	Identifier for static routing.
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_EVENT	Request for data change.

Field	Value	Description
head.l	13	8 octets for the message header, 4 octets of data for the target CSA and 1 octet of data for ON or OFF.
data	E.g. 0x9B000200	CSA of an audio target
	0 or 1	OFF / ON

13.2.2 Static Routing Change

On success, the server will reply to the request above by sending a PK_CHANGE message with the server OriginID containing the requested data.

If a target's static flag is changed by any client, the server will create notifications for all clients which have subscribed to the MX_TARGET_STATIC stream.

13.2.3 Static Routing Query

The client may query a target's static routing flag by sending the following message:

Field	Value	Description
head.m	MX_TARGET_STATIC	Identifier for static routing.
head.n	0	Don't care.
head.o	OriginID	Must be set to the OriginID the server has sent to the client in the login reply.
head.p	PK_QUERY	Request for data.
head.l	12	8 octets for the message header and 4 octets of data for the target CSA.
data	E.g. 0x9B000200	CSA of an audio target

13.2.4 Static Routing Reply

If the specified target is found, the server will reply to the above sending a static routing reply message.

Field	Value	Description
head.m	MX_TARGET_STATIC	Identifier for static routing.
head.n	0	Don't care.
head.o	OriginID	Server's Origin ID.
head.p	PK_QUERY_REPLY	Information about queried data.
head.l	13	8 octets for the message header, 4 octets of data for the target CSA and 1 octet of data for ON or OFF.
data	E.g. 0x9B000200	CSA of an audio target
	0 or 1	OFF / ON

13.2.5 Unsuccessful Static Routing Query

If the specified target cannot be found or if the data is not available, the server will respond with the following message:

Field	Value	Description
head.m	MX_TARGET_STATIC	Identifier for static routing.
head.n	0	Don't care.
head.o	OriginID	Server's Origin ID.
head.p	PK_QUERY_NOREPLY	Unsuccessful query.
head.l	12	8 octets for the message header and 4 octets of data for the target CSA.
data	E.g. 0x9B000200	CSA of an audio target

14 Using signal generators (noise & tone)

The Nova73 series router offers two different possibilities to implement a tone or noise generator. If the system is equipped with a 980/31 or 980/32 audio router, the generators have to be installed as DALLIS I/O units. This implies, that the HLSD or CSA can be set freely including the signal mapping feature as describe in chapter 2.4.

If the system is equipped with a 980/33 audio router, the signal generators are located on the router card and the HLSDs are generated automatically. Therefore, the Remote MNOPL control is slightly different for the two cases and each of them will be depicted in a separate chapter.

14.1 Signal generators on DALLIS I/O units

The device type query (cf. chapter 5.3) on a HLSD or CSA which represents a signal generator on a DALLIS I/O unit will return DEVICETYPE_SIGGEN. In this case, the following control commands can be used to modify or query the settings:

head.m	Parameter Set	Comments																																
IO_DALLIS_SIGGEN_SIGTYPE	<pre>struct { CSA address; byte type; }</pre>	<p>The type of the generated signal. The following values are allowed:</p> <table><tr><th>data</th><th>type</th></tr><tr><td>0</td><td>off</td></tr><tr><td>1</td><td>sine generator</td></tr><tr><td>2</td><td>white noise generator</td></tr><tr><td>3</td><td>pink noise generator</td></tr></table>	data	type	0	off	1	sine generator	2	white noise generator	3	pink noise generator																						
data	type																																	
0	off																																	
1	sine generator																																	
2	white noise generator																																	
3	pink noise generator																																	
IO_DALLIS_SIGGEN_FREQUENCY	<pre>struct { CSA address; FREQ freq; }</pre>	<p>The frequency of the generated signal if the sigtype value above specifies a tone generator. This setting has no effect on a noise generator.</p> <p>The following table declares the valid settings for the <i>freq</i> parameter:</p> <table><tr><th>data</th><th>freq</th></tr><tr><td>2131</td><td>20 Hz</td></tr><tr><td>2782</td><td>50 Hz</td></tr><tr><td>3276</td><td>100 Hz</td></tr><tr><td>3769</td><td>200 Hz</td></tr><tr><td>4262</td><td>400 Hz</td></tr><tr><td>4330</td><td>440 Hz</td></tr><tr><td>4914</td><td>1000 Hz</td></tr><tr><td>5407</td><td>2000 Hz</td></tr><tr><td>5695</td><td>3000 Hz</td></tr><tr><td>5900</td><td>4000 Hz</td></tr><tr><td>6057</td><td>5000 Hz</td></tr><tr><td>6298</td><td>7000 Hz</td></tr><tr><td>6552</td><td>10000 Hz</td></tr><tr><td>6840</td><td>15000 Hz</td></tr><tr><td>7045</td><td>20000 Hz</td></tr></table>	data	freq	2131	20 Hz	2782	50 Hz	3276	100 Hz	3769	200 Hz	4262	400 Hz	4330	440 Hz	4914	1000 Hz	5407	2000 Hz	5695	3000 Hz	5900	4000 Hz	6057	5000 Hz	6298	7000 Hz	6552	10000 Hz	6840	15000 Hz	7045	20000 Hz
data	freq																																	
2131	20 Hz																																	
2782	50 Hz																																	
3276	100 Hz																																	
3769	200 Hz																																	
4262	400 Hz																																	
4330	440 Hz																																	
4914	1000 Hz																																	
5407	2000 Hz																																	
5695	3000 Hz																																	
5900	4000 Hz																																	
6057	5000 Hz																																	
6298	7000 Hz																																	
6552	10000 Hz																																	
6840	15000 Hz																																	
7045	20000 Hz																																	
IO_DALLIS_SIGGEN_LEVEL	<pre>struct { CSA address; LEVEL level; }</pre>	<p>The output level of the generated signal with a resolution of 1/32 dB.</p> <p>The level may be adjusted from -128dB to 0 dB which yields LEVEL values from -4096 to 0.</p>																																

14.2 Signal generators on Router Card 980/33

14.2.1 Automatic generation of HLSDs

The HLSDs for signal generators on 980/33 router cards will be created automatically when the card is configured. The created HLSDs can neither be modified nor mapped.

Please note: Due to the automatic generation of the generator's HLSDs, the Signal Mapping feature as described in chapter 2.4 is not available. This means that if a client wants to control generator settings, it currently has to use the HLSD addressing scheme (mapping mode 0).

The structure of the generated HLSDs is as follows:

HLSD Class	HLSD Subclass	HLSD Component	HLSD Subcomponent	Function
GEN (=0x5C)	System HLSD Prefix as set in AdminHD, usually 0	0	0	Sine generator
GEN (=0x5C)	System HLSD Prefix as set in AdminHD, usually 0	1	0	2 nd sine generator
GEN (=0x5C)	System HLSD Prefix as set in AdminHD, usually 0	2	0	White noise generator
GEN (=0x5C)	System HLSD Prefix as set in AdminHD, usually 0	3	0	Pink noise generator

14.2.2 Controlling the core sine generators

The device type query (cf. chapter 5.3) on a HLSD which represents a core sine signal generator on a 980/33 unit will return DEVICETYPE_CORE_SIGGEN_SINE. In this case, the following control commands can be used to modify or query the settings:

<i>head.m</i>	<i>Parameter Set</i>	<i>Comments</i>								
IO_BOX_SIGGEN_SINE_FREQUENCY	<pre>struct { HLSD address; FREQ freq; }</pre>	<p>The frequency of the generated signal. <i>Freq</i> can be computed using the following equation:</p> $\text{data} = 1638 * \log (\text{frequency/Hz})$ <p>The valid range is 2131 to 7045 (20 Hz to 20.000 Hz).</p> <p>Examples:</p> <table><tr><th>data</th><th>frequency [Hz]</th></tr><tr><td>2131</td><td>20</td></tr><tr><td>4914</td><td>1.000</td></tr><tr><td>7045</td><td>20.000</td></tr></table>	data	frequency [Hz]	2131	20	4914	1.000	7045	20.000
data	frequency [Hz]									
2131	20									
4914	1.000									
7045	20.000									
IO_BOX_SIGGEN_SINE_LEVEL	<pre>struct { HLSD address; LEVEL level; }</pre>	<p>The output level of the generated signal with a resolution of 1/32 dB.</p> <p>The level may be adjusted from -96 dB to 0 dB which yields LEVEL values from -3072 to 0.</p>								

14.2.3 Controlling the core noise generators

The device type query (cf. chapter 5.3) on a HLSD which represents a core noise signal generator on a 980/33 unit will return DEVICETYPE_CORE_SIGGEN_NOISE. In this case, the following control command can be used to modify or query the settings:

<i>head.m</i>	<i>Parameter Set</i>	<i>Comments</i>
IO_BOX_SIGGEN_NOISE_LEVEL	<pre>struct { HLSD address; LEVEL level; }</pre>	<p>The output level of the generated signal with a resolution of 1/32 dB.</p> <p>The level may be adjusted from -96 dB to 0 dB which yields LEVEL values from -3072 to 0.</p>

15 Receiving Error Messages and Warnings

In order to receive error and warning messages from the server, the client has to subscribe to the SA_ROUTERMESSAGE stream. The messages are unsolicited messages so it is not necessary to send queries in order to receive error or warning information. The server is sending error and warning messages immediately after having detected an abnormal state.

Field	Value	Description
head.m	SA_ROUTERMESSAGE	Router message.
head.n	0	Don't care.
head.o	OriginID	OriginID of the server.
head.p	PK_EVENT	Message is of type event.
head.l	20 + x	8 octets for the message head, 14 octets for the messageinfo struct and the number (x) of octets containing the description of the event occurred in the system.
data	Data structure defined in 15.1	Message info struct and a string of octets containing the event description.

15.1 Errorinfo Struct

The first six bytes contain a time stamp which is easily readable e.g. when sniffing the network protocol for debugging purposes.

The routernr field is the number of the router in the system, the slotnr is the number of the slot where I/O boards are integrated, the portnr is the port on the slot module where e.g. I/O systems can be connected and the ioslotnr is the number of the board in the I/O system if such a system is used. If one of these fields is of no interest it will be set to NONE (0xFF).

```
struct
{
    byte year;
    byte month;
    byte day;
    byte hour;
    byte minute;
    byte second;
    unsigned short errorkey;
    byte errorstate;
    byte severity;
    byte routernr;
    byte slotnr;
    byte portnr;
    byte ioslotnr;
};
```

The errorkey is a unique key value identifying the error. For details about the errorkey please refer to chapter 18.

The errorstate field may be set to ERR_OCCURRED or ERR_REMEDIED.

The severity field may contain one of the following values: MSG_CRITICAL, MSG_ERROR, MSG_WARNING, MSG_INFO (in descending order).

16 Manufacturer and Device ID Table

Company	Manufacturer ID	Device	Device ID
Lawo	0	Management Service	0
		Matrix Server	1
		NovaControl	2
Thomson Multimedia	1	VM3000	1
		CM4000	2
Ingenieurbüro Veith	2	ROSY	0
BFE	3	KSC 9000	0
L-S-B Broadcast Technologies	4	Virtual Studio Manager	0
Otaritec	5	CB-179	0
BBC Technology	6	Colledia Control	0
Fraunhofer Gesellschaft	7	Spamix	0
Pharos Communications	8	Pilot	0
Mel Broadcast Solutions	9	generic	0
Netia	10	generic	0
DataPath	11	MaxView	0
Reserved	12	Reserved	0
Reserved: TV4	13	R2 protocol converter	0
Google, Inc.	14	Google Radio Automation	0
SWR Outdoor Broadcasting	15	Cassandra	0
Evertz	16	SC-1000 System Controller	0
Utah Scientific	17	SC-4 Control System	0
Sony	18	ELC-MVS01	0
IBT Interfaces	19	switch...it	0
Miranda	20	NV9000 router control system	0

17 Information Element Table

The Information Element Table is provided in a C++-Style source code format, included in the respective ZIP-file of the documentation:

`remote_mnopl_declarations.h` for the basic MNOPL specifications
`devicetype.h` for the Extended Device Type Information

Labels	Hex Value
LABEL_SIGNALNAME	0
LABEL_STARTUP	1
LABEL_USER	2
LABEL_INHERITED	3
LABEL_GROUP	4
Signals	Hex Value
SIGNAL	0
SILENCE	1
Fade	Hex Value
FADE_IN	1
FADE_OUT	0
Error	Hex Value
ERR_OCCURRED	1
ERR_REMEDIED	0
Protect Check	Hex Value
PROTECT_NO_SUCCESS	0xFF
SIGNAL_SET_STEREO	Hex Value
SIGNAL_SET_STEREO	0xF29F

18 Error classes and error types

The following table serves as a brief description of error classes and error types within the RemoteMNOPL Error Messages. The composition of error class and error type forms the so-called error key. The composition instruction is as follows:

error key = (error class << 8) | error type;

Error class (hex)	Error Type (hex)	Meaning
00 - 0F	-	System Errors
01	-	System Internal Errors
02	-	System Redundancy Errors
02	00 - 1F	Redundancy Takeover Errors
02	01	Redundancy Takeover
03	-	System Doublestar Errors
03	00 - 1F	Doublestar Takeover Errors
03	01	Doublestar Takeover
60 - 6F	-	Device Errors
61	-	"Box" Device Errors
61	00 - 0F	Box PSU Errors
61	01	48V Failure / Complete Power Failure
61	02	PSU1 Failure
61	03	PSU2 Failure
61	10 - 2F	Box Temperature Errors
61	11	Box Fan Alarm
61	30 - 3F	Box Network Errors
61	31	Network Connection Error
61	40 - 4F	Box Sync Device Errors
61	41	Sync Source 1 Failure
61	42	Sync Source 2 Failure
61	43	Sync Source Port (Multichannel Sync) Failure
61	F0 - FD	Box User Alarms
61	F1	User Alarm 1
61	F2	User Alarm 2
61	F3	User Alarm 3
61	F3	User Alarm 4
62	-	Slot Device Errors
62	00 - 0F	Slot Availability Errors
62	01	Slot Module Availability
63	-	Port Device Errors
63	00 - 1F	Common Port Errors
63	01	Link Error Port A (main)
63	02	Link Error Port B (redundant)
63	20 - 6F	Madi Port Errors

63	20 - 2F	Madi Port Common Errors
63	21	Madi Layer Error Port A (main)
63	22	Madi Layer Error Port B (redundant)
63	30 - 3F	Madi Port Rawport Errors
63	40 - 4F	Madi Port Dallisport Errors
63	70 - BF	ATM Port Errors
63	70 - 7F	ATM Port Common Errors
63	71	ATM Layer Error Port A (main)
63	72	ATM Layer Error Port B (redundant)
63	80 - 8F	ATM Port Rawport Errors
63	90 - 9F	ATM Port Dallisport Errors
64	-	Dallisslot Device Errors
64	00 - 0F	Dallisslot Availability Errors
64	01	Dallisslot Module Availability
65	-	Resource Errors
6B	-	Dallis Device Errors
6B	00 - 0F	Dallis PSU Errors
6B	01	Dallis PSU1 Failure
6B	02	Dallis PSU2 Failure
6B	10 - 2F	Dallis Temperature Errors

19 Appendix A – Mic/Line card dynamic range design limits

19.1 Basic Condition

- The Control System sets the adjustable range to -20 dB to +70 dB.
- The value of the term (Headroom + Reference Level) is in the range of +12 dB to +24 dB in 1 dB steps.

19.2 Adjustment Range

The basic condition needs to apply:

941/51 Mic Line 8 Mono electric/symmetric	
Mode	Design Limit [dB]
Line	Min: - 24 (+Headroom + Reference level)
	Max: + 35 (+Headroom + Reference level)
Mic	Min: - 4 (+Headroom + Reference level)
	Max: + 55 (+Headroom + Reference level)
941/52 Mic Line 8 Mono trafo symmetric	
941/62 Mic Line 8 Mono trafo symmetric	
Mode	Design Limit [dB]
Line	Min: - 30 (+Headroom + Reference level)
	Max: + 18 (+Headroom + Reference level)
Mic, without Pad	Min: - 6.5 (+Headroom + Reference level)
	Max: + 55 (+Headroom + Reference level)
Mic, with Pad	Min: - 30 (+Headroom + Reference level)
	Max: + 55 (+Headroom + Reference level)

Example:

941/51 in Line-Mode with 9 dB Headroom and 6 dB Reference Level: Min: -9, Max +70 [dB].

