MM Remote Interface Description

en | MM Remote Interface Description

BOSCH

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to describe the remote interface for microphone management between the CCU and third party software.

1.2 Scope

This document describes the remote interface for microphone management. It is meant for developers who want to use this remote interface to control the microphone management application, present in the CCU, remotely. The Interface can be used to build a Microphone Management User interface or a Synoptic Microphone User interface.
For a complete description of the System Set-up can be referred to [SRS_INF].

1.3 Definitions, Acronyms and Abbreviations

CCU	Central Control Unit. This can be either a single-CCU system or a Multi-CCU system.
ACN	Audio Communication Network
DCN	Digital Congress Network
MM	Microphone Management
SC	System Configuration
SI	System Installation
RTS list	Request To Speak list
SPK list	Speakers list
NBK list	Notebook list (list of chairmen and special assigned delegates)
CR list	Comment Request list. An extra type of request to speak list to offer delegates the possibility to request for a comment on the current speaker. On the units and on the Control PC a comment is indicated as 'Response'
CS list	Comment Speakers list. An extra type of speakers list in which delegates can be placed to make a comment on the current speaker.
UnitId	Unit identification, also called unit-number. A unique identification of a unit within the CCU system.
PC	Personal Computer
remote controller	Device (e.g. PC) connected to the CCU which remotely controls a part of the applications present in the CCU.

1.4 References

[SRS_INF]	General Remote Interface Description
[SRS_SCSIINF]	SC & SI Remote Interface Description
[USERDOC_MM]	User Manual LBB 3570
[USERDOC_SM]	User Manual LBB 3571

This document should be referenced as [SRS_MMINF].

1.5 Overview

Chapter 2 describes the Microphone Management Remote Interface in general.
Chapter 3 and chapter 4 describe respectively, the remote functions and the update notifications which can be used to control the microphones of the units connected to the CCU.
Appendix Appendix A gives an overview of the constants used in combination with the remote functions described in this document.
Appendix Appendix B gives an overview of the possible error's which could be returned upon a remote function.
Appendix Appendix C gives an example on using the remote interface for Microphone Management.

2 Microphone management for a Remote interface

2.1 Introduction

The Microphone Management Remote Interface is part of the DCN software which allows for another controlling entity outside the CCU, not being the DCN Control PC, to use the Microphone Management application.

2.2 Remote Microphone Management Control

Microphone Management is the application that allows for controlling the microphones in the conference hall. Typical control issues are e.g.: turning a Microphone On, adding a delegate to the RTS list, changing the Operation Mode etc. More details on the complete MM application can be found in the user manual [USERDOC_MM].
Controlling microphones with a remote interface is by means of calling a defined set of Remote Functions and acting upon a defined set of Update Notifications. The general concept of Remote Functions and Update Notifications is described in [SRS_INF]. [SRS_INF] also describes the protocol and hardware conditions concerning the remote interface.
Together with this remote interface, there are up to three locations in a full connected CCU where MM can be influenced. These locations are:

- The remote interface or remote controller using the RS-232 interface. The remote controller makes Remote Function calls for microphone management.
- A DCN Control PC connected using a PC-card. This DCN control PC also uses Remote Function calls for microphone management.
- The actual units that handle their microphone keys.

To get a full operational system both the DCN control PC and the remote controller must register themselves to the CCU, so they will receive update messages from the CCU.
Remote functions coming from either the DCN control PC or the remote controller initiates in the CCU an update of the internal lists. During the update, notifications are generated and sent to both the DCN control PC and the remote controller. In this way both remote controllers get the update information about the actions performed on either the DCN control PC or the remote controller.
During the processing of remote functions on the CCU, the update messages are created and transmitted. This implies that the response information of a remote function can be received after the reception of an update notification. The remote controller must wait for the response of the remote function. After reception of the response appropriate action should be taken upon the error code returned. The notifications received during the wait for the response may be processed directly.
Requests coming from a unit are processed and the lists updated. During the update, notifications are generated and sent to all registered PC's. In the system mentioned above, both the DCN control PC and the remote controller will receive the same update notifications.
This document gives the set of Remote Functions and the set of Update Notifications concerning Microphone Management. The relation between Remote Function, sent by the remote controller, and Update Notifications is given in the description of each separate Remote Function. The relation between unit events and Update Notifications is given in section 4.1.2. At last, there is a relation between remote functions sent by the DCN control PC and update notifications. Since both remote controller and DCN control PC receive all update notifications, the set of update notifications therefor also contains those that are the result of a remote function from the DCN control PC.

2.3 Microphone List and Mode Management

Handling the microphones in the system is basically a way of managing the various microphone lists identified inside the CCU and choosing the appropriate operation mode. The Microphone Management application has five microphone lists, which will be explained in the table below:

List	Explanation
Notebook	The notebook contains units having special privileges for turning on their microphone. This list always contains the Chairman units in the system. Other units can only be added to the notebook from within the MM application on a DCN Control PC. The notebook exists in all operation modes.
Speakers list	The speakers list contains the normal delegate units that are currently allowed to speak. Note that this does not mean that those units have

	their microphone switched on. Depending on the operation mode it is possible that a unit is in the speakers list with its microphone switched off. The speakers list exists in all operation modes except for the mode Delegate with Voice activation.
Request to Speak list	The request to speak list contains the unit/delegate combinations that requested to have their microphone switched on so they can speak. Depending on the operation mode an unit/delegate is automatically promoted to the speakers list or by means of an operator action. The request to speak list exists in the modes Operator with Request list, Operator with Request and Response list and Delegate with Request list.
Comment Request list	The comment request list, or response request list, contains the unit/delegate combinations that wants to make an immediate response on the current speaker. This comment request list is to prevent them from being added at the end of the normal request to speak list and thus loosing the urgency of the response. The comment request list is only available in the mode Operator with Request and Response list.
Comment Speakers list	The comment speakers list, or response speakers list, contains the units that are promoted from the comment request list to make their response (i.e. they are allowed to speak now). Promoting a unit from the comment request list to the comment speakers list can only be done by means of an operator action. The comment speakers list is only available in the mode Operator with Request and Response list.

A full description about the operation modes is given in the user manuals [USERDOC_MM] and [USERDOC_SM]. In the table below the operation modes are identified by the value used in the remaining part of this document. This table also describes the enabling/disabling of sets of remote functions and update notifications as result of choosing a specific operation mode.

Mode	Mode description & Group enable/disable
OPERATOR WITH REQUEST LIST equals *MM_C_OPERATOR_WITH_REQ_LIST*	Manual mode. The operator (using the remote controller) controls the RTS list. Delegates are always added to the RTS list and the operator determines which delegate may speak. Special features are to disable the cancel of an request and to turn off the microphone by the delegates (see section 3.2.7) • enables all RTS functions/notifications • enables all SPK functions/notifications • disables all CR functions/notifications • disables all CS functions/notifications
DELEGATE WITH REQUEST LIST equals *MM_C_DELEGATE_WITH_REQ_LIST*	Open delegate mode. All functions can be done by either the operator or the delegates. When a delegate turns his microphone off and there are still delegates present in the RTS list, then an automatic shift will take place. Special features are to disable the cancel of an request. • enables all RTS functions/notifications • enables all SPK functions/notifications • disables all CR functions/notifications • disables all CS functions/notifications
DELEGATE WITH OVERRIDE equals *MM_C_DELEGATE_WITH_OVERRIDE*	Override mode. In this mode there is no RTS-list. Whenever a delegate presses his micro-button, he is directly able to speak. When the SPK list was full, then the oldest speaker will be removed to make place for the new delegate. • disables all RTS functions/notifications • enables all SPK functions/notifications • disables all CR functions/notifications • disables all CS functions/notifications
DELEGATE WITH VOICE ACTIVATION equals *MM_C_DELEGATE_WITH_VOICE*	Voice mode. The CCU automatic focus on the delegate currently speaking. In this mode there is no RTS list and SPK list. Also none of the chairmen microphones will be notified. • disables all RTS functions/notifications • disables all SPK functions/notifications • disables all CR functions/notifications • disables all CS functions/notifications • disables all microphone on/off functions
OPERATOR WITH REQUEST AND RESPONSE LIST equals *MM_C_OPERATOR_WITH_COMMENT_LIST*	Comment mode. The operator (using the remote controller) controls the RTS and CR lists. Delegates are always added to the RTS list for normal requests and to the CR list for responses. The operator determines which delegate may speak and/or make a response. In this mode the maximum number of active microphones must be set to 1. Special features are to disable the cancel of an request and to turn off the microphone by the delegates (see section 3.2.7) • enables all RTS functions/notifications • enables all CR functions/notifications • enables all SPK functions/notifications • enables all CS functions/notifications

The SPK functions and notifications mentioned in the table are described in respectively. sections 3.3 and 4.3.

The CS functions and notifications mentioned in the table are described in respectively. sections 3.4 and 4.4.

The RTS functions and notifications mentioned in the table are described in respectively. sections 3.5.4 and 4.6.
The CR functions and notifications mentioned in the table are described in respectively. sections 3.7 and 4.7.

3 Remote Functions

3.1 Introduction

This chapter describes the various remote functions needed to perform microphone management on the system.

3.1.1 Preconditions

The remote functions for the MM application acting on any of the microphone lists always use the UnitId to perform the requested functionality. For the Request to speak list or Comment Request list functions also a DelegateId is required. This UnitId and DelegateId must be retrieved respectively set, using the appropriate functions of the SC/SI Remote Interface as described in [SRS_SCSIINF].

3.1.2 Remote function item explanation

Each description consists of the following items:

- **Purpose**
  A global description of the purpose of the function.
- **Parameter structure for the function**
  The input parameters needed to fulfil the function. When the function requires no parameters, no structure is described here. The type definitions of the basic types used to build up the input parameter structure are given in [SRS_INF].
- **Response structure from the function**
  The output information coming from the function called. This information is only valid when the 'wError' field of the received response information equals MM_E_NOERROR.
- **Error codes returned**
  The possible error values returned in the 'wError' field of the response information for this remote function. All different error codes are described in appendix Appendix B.
- **Update notifications**
  The update notifications which are generated during the execution of the remote function. When there are no notifications generated, then this part will be omitted.
- **Related functions**
  The related function in conjunction with the function described. It refers to other remote functions and to related update notifications.

3.2 MM General functions

3.2.1 MM_C_START_MM

Purpose

Indicates the CCU that the remote controller wants updates notifications from the MM application inside the CCU. After receiving this function the CCU increments the update use count. As long as the update use count is greater than zero, the CCU will sent update notifications to the remote controller. Update notifications are sent upon state changes due to actions from the control PC(s) and all microphone actions on the units. During the time the Microphone Management application is controlled remotely (i.e. the update use count is greater than zero) the LED's on the control panel of the CCU are turned off and buttons on the control panel are disabled.
When you omit the execution of this remote function, you can still execute remote functions, but no update notifications will be sent to the remote controller.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function returns the following structure:

```
WORD    wNrOfInstances
```

where:

wNrOfInstances	The value of the update use count for the MM application at the end of the function handling. It contains the number of times a remote PC has connected over the same communication medium. E.g. the first time the MM_C_START_MM function is

called, it contains the value 1.

Error codes returned

MM_E_NOERROR
MM_E_OPEN_CLOSE_FAILED

Related functions

MM_C_STOP_MM

3.2.2 MM_C_STOP_MM

Purpose

Indicates the CCU that the remote controller no longer requires updates from the MM application inside the CCU. After receiving this function the CCU decrements the update use count. As long as the update use count is still greater than zero, the CCU remains sending the update notifications to the remote controller.
A call to this function when the update use count is already zero will keep the use count to zero and nothing shall happen.
When the use count reaches zero then the microphone management application inside the CCU returns to its stand-alone operation. This return involves a change in the following settings of the MM-application:

Setting Parameter	section	Destination of change
wOperationMode	3.2.5	When the operation mode is MM_C_OPERATOR_WITH_REQ_LIST or MM_C_OPERATOR_WITH_COMMENT_LIST the mode will be changed to MM_C_DELEGATE_WITH_REQ_LIST. All other modes will remain active.
wActiveMics	3.2.6	When the number of active microphones is 3, this will be extended to 4 (as visible on the front panel). This implies also changes of the SPK and RTS lists.
bAllowMicroOff	3.2.7	This value is set to TRUE. Note that this value is only used in the modes MM_C_OPERATOR_WITH_REQ_LIST and MM_C_OPERATOR_WITH_COMMENT_LIST.

All other MM-settings remain active while functioning in stand-alone mode.

Note that: Upon communication lost this function will be activated, if MM_C_START_MM was activated. The activation of this function is repeated till the update use count becomes zero.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function has the same response structure as the remote function MM_C_START_MM (section 3.2.1).

Error codes returned

MM_E_NOERROR
MM_E_OPEN_CLOSE_FAILED

Related functions

MM_C_START_MM

3.2.3 MM_C_START_MON_MM

Purpose

Function to start the monitoring behaviour of the Microphone Management application. It is not allowed/possible to control settings of Microphone Management.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function returns the following structure:

```
WORD    wNrOfInstances
```

where:

wNrOfInstances The value of the update use count for the MM application at the end of the function handling. It contains the number of times a remote PC has connected over the same communication medium. E.g. the first time the MM_C_START_MON_MM

function is called, it contains the value 1.

Error codes returned

MM_E_NOERROR

Related functions

MM_C_STOP_MM
MM_C_STOP_MON_MM

3.2.4 MM_C_STOP_MON_MM

Purpose

Function to stop monitoring the behaviour of the Microphone Management application.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function returns the same response structure as the remote function MM_C_START_MON_MM (section 3.2.3)

Error codes returned

MM_E_NOERROR

Related functions

MM_C_START_MM
MM_C_START_MON_MM

3.2.5 MM_C_SET_MIC_OPER_MODE

Purpose

This function allows the remote controller to change the microphone operation-mode.

Parameter structure for the function

The function requires the following structure as parameter:

```
WORD    wOperationMode;
```

where:

wOperationMode The operation mode of the MM application which is one of the following:

- MM_C_OPERATOR_WITH_REQ_LIST
- MM_C_DELEGATE_WITH_REQ_LIST
- MM_C_DELEGATE_WITH_OVERRIDE
- MM_C_DELEGATE_WITH_VOICE
- MM_C_OPERATOR_WITH_COMMENT_LIST

If the operation mode is set to MM_C_OPERATOR_WITH_COMMENT_LIST, the maximum number of active microphones will be set to 1 if not done by the operator.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MIC_OPER_MODE (unknown mode selected)

Update notifications

MM_C_SET_MIC_OPER_MODE_ON_PC
and various SPK, CS, RTS and/or CR updates depending on the difference between the old and new mode set.

3.2.6 MM_C_SET_ACTIVE_MICS

Purpose

This function allows the remote controller to change the maximum number of active microphones (SPK list length).
When the number of active microphones is increased, the created (empty) places will be filled with entries coming from the RTS list if the selected mode equals MM_C_DELEGATE_WITH_REQ_LIST.
When the number of active microphones is reduced, the following rules are applied if the number of speakers in the SPK list is greater than the final size.

- If there are speakers in the list with their microphone off, then first of these will be removed.
- When there are only speakers in the list with their microphone on, the first unit in the list will be turned off and removed from the list

When the microphone operation-mode equals MM_C_OPERATOR_WITH_COMMENT_LIST and the maximum number of active microphones is increased to more than 1 an error is returned.

Parameter structure for the function

The function requires the following structure as parameter:

```
WORD    wActiveMics;
```

where:

wActiveMics	The number of active microphones which can be on at the same time. Valid values are in the range 1..4.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MAX_ACT_MICS

Update notifications

MM_C_SET_ACTIVE_MICS_ON_PC
and various SPK, CS, RTS and/or CR updates depending on the change in size of the speakers list length.

3.2.7 MM_C_GET_SETTINGS

Purpose

Retrieve the general settings from the MM-application. This function can be used to get the initial state of the operation mode and the number of active microphones as set using the button on the front panel of the CCU.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function returns the following structure:

```
typedef struct
{
    WORD    wOperationMode;
    WORD    wActiveMics;
    WORD    wMaxRTSListLen;
    BOOLEAN bAllowCancelRequests;
    BOOLEAN bAllowMicroOff;
    BOOLEAN bAttentionTone;
    BOOLEAN bAmbientMicCtrl;
} MM_T_CCU_GLOBAL_SETTINGS;
```

where:

wOperationMode	The operation mode of the MM application which is one of the following: • MM_C_OPERATOR_WITH_REQ_LIST • MM_C_DELEGATE_WITH_REQ_LIST • MM_C_DELEGATE_WITH_OVERRIDE • MM_C_DELEGATE_WITH_VOICE • MM_C_OPERATOR_WITH_COMMENT_LIST For more information about the different modes see section 3.2.5.
wActiveMics	The number of active delegate microphones which can be on at the same time (chairman micro's are not counted). Range 1..4
wMaxRTSListLen	The maximum Request To Speak list length. Range: 0..100.
bAllowCancelRequest	TRUE: A Delegate is able to cancel a request to speak using the Micro-key on the unit. FALSE: A Delegate is not able to cancel a request to speak. (This parameter is only valid within the operation modes MM_C_OPERATOR_WITH_REQ_LIST, MM_C_DELEGATE_WITH_REQ_LIST and MM_C_OPERATOR_WITH_COMMENT_LIST). ***Note**: A Delegate is always able to cancel a comment request*
bAllowMicroOff	TRUE: A Delegate is able to turn off the microphone on the unit. FALSE: A Delegate is not able to turn off the microphone. This

	implies that the micro can only be turned off by the remote controller. (only valid for the operation modes MM_C_OPERATOR_WITH_REQ_LIST and MM_C_OPERATOR_WITH_COMMENT_LIST).
bAttentionTone	TRUE: An attention tone is generated when the priority key is pressed on a chairman-unit. FALSE: No attention is generated when the priority key is pressed.
bAmbientMicCtrl	TRUE: The ambient microphone control is enabled. Ambient mic. control means that the ambient mic. is turned on when the last microphone of all units in the conference hall is switched off and it is turned off when the first microphone is switched on. FALSE: The ambient microphone control is disabled, i.e. the ambient mic. will always be switched off.

Error codes returned

MM_E_NOERROR

Related functions

MM_C_SET_SETTINGS

3.2.8 MM_C_SET_SETTINGS

Purpose

Set the general operating settings of the MM-application.
If the operation mode is set to MM_C_OPERATOR_WITH_COMMENT_LIST, the value for maximum number of active microphones will be omitted and the maximum number of active microphones will be set to 1.

Parameter structure for the function

The structure to be passed along with this function is the same structure as the structure received during the remote function MM_C_GET_SETTINGS (see 3.2.7).

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MIC_OPER_MODE
MM_E_ILLEGAL_MAX_ACT_MICS
MM_E_ILLEGAL_MAX_RTS_LIST_LEN
MM_E_RTS_LIST_CHANGED
MM_E_DELETE_RTS_LIST_FAILED

Update notifications

MM_C_SET_SETTINGS_ON_PC
and various SPK, CS, RTS and/or CR updates depending on the settings made.

Related functions

MM_C_GET_SETTINGS

3.3 MM Speaker list functions

This section describes the functions to manipulate the speakers list.

3.3.1 MM_C_SET_MICRO_ON_OFF

Purpose

Control the microphone of a unit. This function gives the ability to turn the microphone of a unit on or off. To describe the functionality included with this function several cases of this function are described in the table below:

Case	Action performed
Delegate unit micro on	The unit is appended to the SPK list if possible.
Delegate unit micro off	The units microphone is turned off, but the unit still remains in the SPK list. To remove the speaker also from the SPK list, use the remote call MM_C_SPK_REMOVE (see section 3.3.3).
Delegate unit micro on	The units microphone is turned on. The unit remains in

(already in the SPK list)	the SPK list.
Chairman unit micro on	The units microphone is turned on.
Chairman unit micro off	The units microphone is turned off.

Parameter structure for the function

The function requires the following structure as parameter:

```
typedef struct
{
    WORD    wUnitId;
    BOOLEAN bMicroOn;
} MM_T_MICRO_ONOFF;
```

where:

wUnitId	Unit Identifier. Unit identifiers can be retrieved from the system using the remote functions for System Config [SRS_SCSIINF].
bMicroOn	TRUE to turn the microphone on, FALSE to turn the microphone off

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_SPEAKERS_LIST_FULL
MM_E_INSERT_SPEAKERS_LIST_FAILED
MM_E_NOT_IN_SPL_OR_NOB
MM_E_UNIT_NOT_CONNECTED

Update notifications

MM_C_SPK_APPEND_ON_PC	(delegate micro on and added to SPK)
MM_C_MICRO_ON_OFF	(micro on/off and already in SPK)

Related functions

MM_C_SPK_APPEND
MM_C_SPK_REMOVE

3.3.2 MM_C_SPK_APPEND

Purpose

Add a unit to the end of the speakers list on the CCU. The addition of a unit to the SPK list automatically implies that the microphone will be turned on.
Note that this function always adds the unit to the speakers list. Even if this unit is a chairman. A good practice is to use the remote function MM_C_SET_MICRO_ON_OFF for managing the microphones state.
When the unit is already present in the SPK list, an error is reported and the current microphone status of the unit is unchanged.
The CS list, if present, will be cleared.

Parameter structure for the function

The function requires the following structure as parameter:

```
typedef struct
{
    WORD    wUnitId;
} MM_T_SPK;
```

where:

wUnitId	Unit Identifier

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MIC_OPER_MODE
MM_E_ILLEGAL_MICRO_TYPE
MM_E_UNIT_ALREADY_PRESENT
MM_E_SPEAKERS_LIST_FULL
MM_E_INSERT_SPEAKERS_LIST_FAILED
MM_E_UNIT_NOT_CONNECTED

Update notifications

MM_C_SPK_APPEND_ON_PC

Related functions

MM_C_SPK_REMOVE

3.3.3 MM_C_SPK_REMOVE

Purpose

Removes a speaker from the SPK list on the CCU. A removal of a unit from the SPK list automatically implies that the units microphone will be turned off.

Parameter structure for the function

This function requires the structure MM_T_SPK as parameter. This structure is defined in section 3.3.2.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MIC_OPER_MODE
MM_E_UNIT_NOT_PRESENT
MM_E_DELETE_SPEAKERS_LIST_FAILED

Update notifications

MM_C_SPK_REMOVE_ON_PC

Related functions

MM_C_SPK_APPEND

3.3.4 MM_C_SPK_CLEAR

Purpose

Clear all entries in the SPK list on the CCU. All delegate microphones are turned off. The chairmen microphones remains in the same state.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR

Update notifications

MM_C_SPK_CLEAR_ON_PC

Related functions

MM_C_SPK_APPEND

3.3.5 MM_C_SPK_GET

Purpose

Retrieve the complete contents of the Speakers list as present in the CCU.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function returns the following structure:

```
typedef struct
{
    WORD    wNrOfSpk;
    MM_T_SPK_MICRO   tSpkList[DBSC_MAX_SPEAKERLIST];
} MM_T_CCU_SPKLIST;
```

Where the MM_T_SPK_MICRO is defined as:

```
typedef struct
{
    WORD    wUnitId;
    BOOLEAN bMicroOn;
} MM_T_SPK_MICRO;
```

where:

wNrOfSpk	The number of SPK list entries actual present in the tSpkList array. Only this amount of array elements are transmitted. This value never exceeds the constant

DBSC_MAX_SPEAKERLIST.

tSpkList [] Array holding the SPK list information. Each array element is defined as a MM_T_SPK_MICRO structure which is defined below.

wUnitId Unit identifier

bMicroOn TRUE if the microphone is currently on
FALSE if the microphone is currently off

Error codes returned

MM_E_NOERROR

Related functions

MM_C_SPK_APPEND

3.4 MM Comment Speaker list functions

This section describes the functions to manipulate the comment speakers list. Note that a Comment Speaker can only be generated by shifting a Comment Request using the MM_C_SHIFT_CR function (see also section 3.7.3).

3.4.1 MM_C_CS_REMOVE

Purpose

Removes a speaker from the CS list on the CCU. A removal of a unit from the CS list automatically implies that the units microphone will be turned off.

Parameter structure for the function

This function requires the structure MM_T_SPK as parameter. This structure is defined in section 3.3.2.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MIC_OPER_MODE
MM_E_UNIT_NOT_PRESENT
MM_E_UNKNOWN_UNIT

Update notifications

MM_C_CS_REMOVE_ON_PC

3.4.2 MM_C_CS_GET

Purpose

Retrieve the complete contents of the Comment Speakers list as present in the CCU.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function returns the following structure:

```
typedef struct
{
    WORD             wNrOfCS;
    MM_T_SPK_MICRO   tCSList[DBSC_MAX_DELCS];
} MM_T_CCU_CSLIST;
```

where:

wNrOfCS The number of CS list entries actual present in the tCSList array. This value never exceeds the constant DBSC_MAX_DELCS.

tCSpkList [] Array holding the CS list information. Each array element is defined as a MM_T_SPK_MICRO structure which is defined in 3.3.5.

Error codes returned

MM_E_NOERROR

3.5 MM Notebook list functions

This section describes the functions to manipulate the Notebook list.

3.5.1 MM_C_NBK_REMOVE

Purpose

Remove one entry from the Notebook as present in the CCU.

Parameter Structure for the function

The function requires the MM_T_NBK structure for input. This structure is defined in section 4.5.1.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_DELETE_NOTEBOOK_FAILED

Update notifications

MM_C_NBK_REMOVE_ON_PC

Related Functions

MM_C_NBK_SET
MM_C_NBK_GET

3.5.2 MM_C_NBK_CLEAR

Purpose

Clear the complete contents of the Notebook list

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR

Update notifications

MM_C_NBK_SET_ON_PC

Related Functions

MM_C_NBK_SET

3.5.3 MM_C_NBK_GET

Purpose

Retrieve the complete contents of the Notebook list as present in the CCU.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function returns the following structure:

```
typedef struct
{
    WORD    wNrOfNbk;
    MM_T_NBK_MICRO   tNbkList[DBSC_MAX_NOTEBOOKLIST];
} MM_T_CCU_NBKMICROLIST;
```

Where the MM_T_NBK_MICRO is defined as:

```
typedef struct
{
    WORD    wUnitId;
    WORD    wMicroType;
    BOOLEAN bMicroOn;
} MM_T_NBK_MICRO;
```

where:

wNrOfNbk	The number of NBK list entries actual present in the tNbkList array. Only this amount of array elements are transmitted. This value never exceeds the constant DBSC_MAX_NOTEBOOKLIST.
tNbkList [ ]	Array holding the NBK list information. Each array element is defined as a MM_T_NBK_MICRO structure which is defined

below.

wUnitId Unit Identifier

wMicroType The type of microphone handling for the notebook entry. The following microphone types are valid for the notebook entries:

- MM_C_VIP_CHAIRMAN
- MM_C_VIP_KEY
- MM_C_VIP_OPERATOR
- MM_C_VIP_VOICE
- MM_C_VIP_VCHAIR
- MM_C_CHAIRMAN_NO_AC
- MM_C_KEY_NO_AC
- MM_C_OPERATOR_NO_AC
- MM_C_VOICE_NO_AC
- MM_C_VCHAIR_NO_AC

bMicroOn TRUE if the microphone is currently on
FALSE if the microphone is currently off

In a typical, stand alone, configuration the notebook contains only the chairman units, which appear as MM_C_VIP_CHAIRMAN entries in the notebook list. Other type of notebook entries can only be added using a DCN Control PC.

Error codes returned

MM_E_NOERROR

3.5.4 MM_C_NBK_SET

Purpose

Set the complete contents of the Notebook list

Parameter structure for the function

The function requires the following structure as parameter:

```
typedef struct
{
    WORD         wNrOfNbk;
    MM_T_NBK     tNbkList[DBSC_MAX_NOTEBOOKLIST];
} MM_T_CCU_NBKLIST;
```

Where the MM_T_NBK is defined as:

```
typedef struct
{
    WORD     wUnitId;
    WORD     wMicroType;
} MM_T_NBK;
```

where:

wNrOfNbk The number of NBK list entries actual present in the tNbkList array. Only this amount of array elements are transmitted. This value never exceeds the constant DBSC_MAX_NOTEBOOKLIST.

tNbkList [ ] Array holding the NBK list information. Each array element is defined as a MM_T_NBK_MICRO structure which is defined below.

wUnitId Unit Identifier

wMicroType The type of microphone handling for the notebook entry. The following microphone types are valid for the notebook entries:

- MM_C_VIP_CHAIRMAN
- MM_C_VIP_KEY
- MM_C_VIP_OPERATOR
- MM_C_VIP_VOICE
- MM_C_VIP_VCHAIR

- MM_C_CHAIRMAN_NO_AC
- MM_C_KEY_NO_AC
- MM_C_OPERATOR_NO_AC
- MM_C_VOICE_NO_AC
- MM_C_VCHAIR_NO_AC

In a typical, stand alone, configuration the notebook contains only the chairman units, which appear as MM_C_VIP_CHAIRMAN entries in the notebook list. Other type of notebook entries can only be added using a DCN Control PC.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_DELETE_NOTEBOOK_FAILED
MM_E_INSERT_NOTEBOOK_FAILED
MM_E_UPDATE_NOTEBOOK_FAILED

Update notifications

MM_C_NBK_SET_ON_PC

Related Functions

MM_C_NBK_GET

3.6 MM Request to Speak list functions

This section describes the functions to manipulate the RTS list. The RTS list is a list of delegates with their unit identifications which are waiting to get speech-time.
Both the UnitId and the DelegateId are present in the RTS list, because using access-control with cards and free seating, allows a delegate to leave its unit (taking out his card) and go to another unit (inserting his card again). During these actions a pending request of that delegate must remain in the RTS list and while the card is not in the system the unit of the delegate is unknown.
For manipulation of the RTS list a special structure is used to identify a RTS list entry. The structure is defined as follows:

```
typedef struct
{
    WORD    wUnitId;
    WORD    wDelegateId;
} MM_T_RTS;
```

where:

wUnitId	Unit Identifier. Must be unique in the RTS list
wDelegateId	Delegate Identifier. May also have the value DBSC_EMPTY_DELEGATE, when the delegate is unknown. Delegate identifiers can be set in the system using the remote functions for System Config [SRS_SCSIINF].

When a RTS list entry is passed with one of the RTS functions the CCU tries to complete the RTS information passed. This means that when only the ‘wUnitId’ is provided, the CCU will search the correct delegate and when only the ‘wDelegateId’ is provided, the CCU will search for the correct unit. Assumed is that not provided elements are filled with the according DBSC_EMPTY_UNIT or DBSC_EMPTY_DELEGATE value. When both elements of the structure have empty values or the unit and the delegate contradict each other, all functions (except MM_C_SHIFT, see section 3.6.6) generate an error (MM_E_UNKNOWN_UNITID_AND_DELID or MM_E_UNITID_DELID_MISMATCH).

3.6.1 MM_C_RTS_APPEND

Purpose

Add a delegate/unit combination to the RTS list on the CCU.

Parameter structure for the function

This function requires the structure MM_T_RTS as parameter. This structure is defined in section 3.6.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MIC_OPER_MODE
MM_E_UNKNOWN_UNITID_AND_DELID
MM_E_UNIT_ALREADY_PRESENT
MM_E_UNIT_NOT_CONNECTED
MM_E_UNITID_DELID_MISMATCH
MM_E_RTS_LIST_FULL

Update notifications

MM_C_RTS_INSERT_ON_PC
MM_C_RTS_FIRST_ON_PC (if appended delegate becomes the first in the list)

Related functions

MM_C_RTS_REMOVE
MM_C_RTS_CLEAR

3.6.2 MM_C_RTS_REMOVE

Purpose

Remove one delegate/unit combination from the RTS list on the CCU.

Parameter structure for the function

This functions requires the structure MM_T_RTS as parameter. This structure is defined in section 3.6.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MIC_OPER_MODE
MM_E_RTS_LIST_EMPTY
MM_E_UNKNOWN_UNITID_AND_DELID
MM_E_UNIT_NOT_PRESENT
MM_E_UNITID_DELID_MISMATCH

Update notifications

MM_C_RTS_REMOVE_ON_PC
MM_C_RTS_FIRST_ON_PC (if removed delegate was the first in the list)

Related functions

MM_C_RTS_APPEND
MM_C_RTS_CLEAR

3.6.3 MM_C_RTS_CLEAR

Purpose

Clear all pending requests in the system. This includes clearing all entries in the RTS list, and clearing all entries in the CR list, if present.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR

Update notifications

MM_C_RTS_CLEAR_ON_PC
MM_C_RTS_CLEAR_COMMENT_ON_PC

Related functions

MM_C_RTS_APPEND
MM_C_RTS_REMOVE

3.6.4 MM_C_RTS_GET

Purpose

Retrieve the complete contents of the Request To Speak list as present in the CCU.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function returns the following structure:

```
typedef struct
{
    WORD    wNrOfRts;
    MM_T_RTS tRtsList[DBSC_MAX_DELRTS];
} MM_T_CCU_RTSLIST;
```

where:

wNrOfRts	The number of RTS list entries actual present in the tRtsList array. Only this amount of array elements are transmitted. This value never exceeds the constant DBSC_MAX_DELRTS.
tRtsList [ ]	Array holding the RTS list information. Each array element is defined as a MM_T_RTS structure which is defined in section 3.6.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MIC_OPER_MODE

Related functions

MM_C_RTS_SET

3.6.5 MM_C_RTS_SET

Purpose

Set a new RTS list on the CCU. The current RTS list will be cleared and the provided RTS list will be made current.

Parameter structure for the function

The function needs as parameter a list of RTS entries as defined as response structure by the function MM_C_RTS_GET (section 3.6.4). The same structure received by the function MM_C_RTS_GET must be transmitted by this function.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MIC_OPER_MODE
MM_E_RTS_LIST_TOO_BIG
MM_E_UNKNOWN_UNITID_AND_DELID
MM_E_INSERT_RTS_LIST_FAILED
MM_E_UNITID_DELID_MISMATCH

Update notifications

MM_C_RTS_SET_ON_PC

Related functions

MM_C_RTS_GET

3.6.6 MM_C_SHIFT

Purpose

Perform a shift function, i.e. promote a delegate from the RTS list to the Speakers list. The shift differs from other RTS list or Speakers list functions in such a way that the promoted delegate is always added to the speakers list, whether this list is full or not. Besides, the CS list and CR list if present are also cleared. This includes the following steps:

1. Clear the CR list and the CS list if the mode is MM_C_OPERATOR_WITH_COMMENT_LIST
2. Remove the indicated RTS entry from the RTS list. When the indicated entry does not exist in the RTS list, then the removal is skipped and the entry provided will be used. Note that the latter also holds when the operation mode is MM_C_DELEGATE_WITH_OVERRIDE (see also 3.2.5).
3. Look if there is a entry free in the SPK list. If not, then a free entry will be created using on of the following rules:
    - If there are SPK entries with their microphone off, then first of these will be removed.
    - When there are only SPK entries with their microphone on, the first unit in the list will be turned off and removed from the list
4. Create from the RTS entry a SPK entry and add this to the SPK list.

Parameter structure for the function

The function requires the structure MM_T_RTS as parameter. This structure is defined in section 3.6. Normally the provided RTS list entry defines which delegate/unit combination is candidate to shift to the speakers list.
When the provided RTS is filled with empty values (wUnitId = DBSC_EMPTY_UNIT and wDelegateId = DBSC_EMPTY_DELEGATE), the first RTS entry present in the RTS list is used. If there are no RTS entries present or when the operation mode is MM_C_DELEGATE_WITH_OVERRIDE, nothing happens.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_UNIT_NOT_CONNECTED
MM_E_ILLEGAL_MIC_OPER_MODE
MM_E_RTS_LIST_EMPTY
MM_E_UNITID_DELID_MISMATCH
MM_E_UNKNOWN_UNITID_AND_DELID

Update notifications

MM_C_CR_CLEAR_ON_PC
MM_C_CS_CLEAR_ON_PC
MM_C_SPK_REMOVE_ON_PC
MM_C_RTS_REMOVE_ON_PC
MM_C_SPK_APPEND_ON_PC
MM_C_RTS_FIRST_ON_PC

3.7 MM Comment Request list functions

This section describes the functions to manipulate the CR list. The Comment Request list is a list of delegates with their unit identifications which are waiting to get speech-time to respond to the current speaker. This comment request list is to prevent the delegate from being added at the end of the normal RTS list.
Comment Requests are identified by the same MM_T_RTS structure as normal RTS entries.
Comment Requests show the same behaviour in combination with access-control and cards as normal RTS entries.

3.7.1 MM_C_CR_REMOVE

Purpose

Remove one delegate/unit combination from the CR list on the CCU.

Parameter structure for the function

This functions requires the structure MM_T_RTS as parameter. This structure is defined in section 3.6.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MIC_OPER_MODE
MM_E_RTS_LIST_EMPTY
MM_E_UNKNOWN_UNITID_AND_DELID
MM_E_UNIT_NOT_PRESENT
MM_E_UNITID_DELID_MISMATCH

Update notifications

MM_C_CR_REMOVE_ON_PC

Related functions

MM_C_CR_GET

3.7.2 MM_C_CR_GET

Purpose

Retrieve the complete contents of the CR list as present in the CCU.

Parameter structure for the function

The function has no additional parameters.

Response structure from the function

The function returns the following structure:

```
typedef struct
{
```

```
    WORD        wNrOfCR;
    MM_T_RTS    tCRList[DBSC_MAX_DELCR];
} MM_T_CCU_CRLIST;
```

where:

wNrOfCR	The number of CR list entries actual present in the tCRList array. This value never exceeds the constant DBSC_MAX_DELCR.
tCRList [ ]	Array holding the CR list information. Each array element is defined as a MM_T_RTS structure which is defined in section 3.6.

Error codes returned

MM_E_NOERROR
MM_E_ILLEGAL_MIC_OPER_MODE

Related functions

MM_C_RTS_CLEAR_COMMENT

3.7.3 MM_C_SHIFT_CR

Purpose

Perform a shift function on the CR list, i.e. promote a delegate from the CR list to the CS list. The shift differs from other Comment Request list or Speakers list functions in such a way that the promoted delegate is always added to the comment speakers list, whether this list is full or not. Besides, of all units present in the SPK list the microphones will be turned off. This includes the following steps:

1. Remove the indicated Comment Request entry from the CR list. When the indicated entry does not exist in the CR list an error is returned.
2. Turn off the microphones off all entries in the SPK list.
3. Look if there is a entry free in the CS list. If not, then a free entry will be created by removing the first unit in the CS list.
4. Create from the Comment Request entry a SPK entry and add this to the CS list.

If however, the delegate was already present in the normal speakers list, then the Comment Request entry is removed from the CR list and the microphone of the entry in the SPK list is switched on again.
Note: Currently the operation mode MM_C_OPERATOR_WITH_COMMENT_LIST is only allowed with a maximum number of active speakers of 1. Also the CS list has currently a maximum length of 1. This means that when a comment request is shifted, the microphone of the current speaker in the SPK list is switched off and the current speaker in the CS list, if present, is removed to make place for the shifted CR entry.

Parameter structure for the function

The function requires the structure MM_T_RTS as parameter. This structure is defined in section 3.6.

Response structure from the function

The function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_NOT_PRESENT
MM_E_UNIT_NOT_CONNECTED
MM_E_ILLEGAL_MIC_OPER_MODE
MM_E_UNITID_DELID_MISMATCH
MM_E_UNKNOWN_UNITID_AND_DELID

Update notifications

MM_C_CR_REMOVE_ON_PC
MM_C_CS_REMOVE_ON_PC
MM_C_CS_APPEND_ON_PC

3.8 MM Speechtime functions

This section describes the functions to manipulate the speechtime.
There is no synchronisation between different controllers, e.g. Remote Control and Control-PC. The last controller which is used, is the active one.
It is the responsibility of the controller to invoke the different functions when necessary. The CCU won't do this for you. The controller should check the speechtime for each individual speaker and invoke the relevant speechtime function.

3.8.1 MM_C_SET_SPEECHTIME_SETTINGS

Purpose

This function stores the speechtime settings in the CCU.

Parameter structure for the function

This function requires the following structure as parameter:

```
typedef struct
{
    WORD    wSpeechTimeLimit;
    BOOLEAN bTimerOn;
    BOOLEAN bHoldOnChairPriority;
    BOOLEAN bShowRemainingTime;
} MM_T_SET_SPEECHTIME_SETTINGS;
```

where:

wSpeechTimeLinit	Speech time limit in minutes
bTimerOn	TRUE: use the speech timer FALSE: don't use the speech timer
bHoldOnChairPriority	TRUE: hold timer if one or more Chairman press their Prio button. FALSE: don't hold timer.
bShowRemainingTime	TRUE: downcounting timer. FALSE: upcounting timer.

Response structure from the function

This function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_FAILED

Update notifications

MM_C_TIMER_ON_OFF

Related functions

MM_C_LAST_MINUTE_WARNING
MM_C_TIME_FINISHED_WARNING

3.8.2 MM_C_LAST_MINUTE_WARNING

Purpose

This function is used to inform a particular unit that it is in his last minute of speaking.

Parameter structure for the function

This function has one parameter:

```
WORD    wUnitId;
```

where:

wUnitId	The unit on which to place the message.

Response structure from the function

This function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_UNKNOWN_UNIT
MM_E_SEND_ACTION_FAILED

Related functions

MM_C_SET_SPEECHTIME_SETTINGS
MM_C_TIME_FINISHED_WARNING

3.8.3 MM_C_TIME_FINISHED_WARNING

Purpose

This function is used to inform a particular unit that its time to speak is run out.

Parameter structure for the function

This function has one parameter:

```
WORD    wUnitId;
```

where:

wUnitId The unit on which to place the message.

Response structure from the function

This function has no response parameters.

Error codes returned

MM_E_NOERROR
MM_E_UNKNOWN_UNIT

Related functions

MM_C_SET_SPEECHTIME_SETTINGS
MM_C_LAST_MINUTE_WARNING

4 Update Notifications

4.1 Introduction

This chapter describes the various update notifications sent by the CCU. All the update notifications of the MM application are listed in this chapter.

4.1.1 Update notification item explanation

Each update notification description consists of the following items:

- **Purpose**
  A global description of the purpose of the notification.
- **Notify structure with this update**
  The information passed with the update notification.

4.1.2 Unit/user event relations

As we have mentioned in section 2.2, update notifications are not only the results of remote functions generated by the remote controller, but can also be the results of unit/user events. To understand these relationships, a unit-event matrix is given in this section. It is assumed that the remote controller is used with a stand-alone configuration (i.e. no DCN Control PC connected), so only a distinction between chairman and delegate[1] is made.
In the unit-event matrix for each event the corresponding update notifications are given, depending on the operational mode and the type of unit/user. For the Voice Activated mode there are no update notifications generated at all, so this mode isn't mentioned in the table either. The update notifications themselves are described in the remaining sections of this chapter.
Note that the input events for Microphone and/or Request to Speak are initiated by pressing the Micro button on a Delegate and/or Chairman unit and the input event for Priority is initiated by pressing the Priority button on a chairman unit. The input events for Comment Requests can only occur in the operation mode MM_C_OPERATOR_WITH_COMMENT_LIST. In that mode the main menu[2] and the speakers menu of the delegate units have assigned softkey 3 to the response (i.e. comment) option. This implies that this response option is only available when the unit has the main menu or the MM menus as current menu. Thus, if a voting round is running, or a message is being read, the comment option is not available.

[1] When speaking of chairman or delegate we really mention the user in the conference hall acting on a chairman unit (e.g. LBB3554) and on a delegate unit (e.g. LBB3550 or LBB3551) respectively
[2] On units having softkeys but no display the working is equal as if it were units with display and always showing the main menu.

UNIT-EVENT MATRIX

Input event	**Operational Mode**			
C: Chairman *D: Delegate*	*Delegate with Req.List*	*Operator with Req.List*	*Operator with Request an Response List*	*Delegate with Override*
C: Microphone On	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON
C: Microphone Off	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON
C: Priority On	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON
C: Priority Off	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON	MM_C_MICRO_ON_OFF MM_C_NR_CHAIR_MICS_ON
D: Request to Speak	If the speakers list is not full: MM_C_SPK_APPEND_ON_PC else, if the RTS list is not full: MM_C_RTS_INSERT_ON_PC and if it is also the first in the RTS list: MM_C_RTS_FIRST_ON_PC	if the RTS list is not full: MM_C_RTS_INSERT_ON_PC and if it is also the first in the RTS list: MM_C_RTS_FIRST_ON_PC	if the RTS list is not full: MM_C_RTS_INSERT_ON_PC and if it is also the first in the RTS list: MM_C_RTS_FIRST_ON_PC	If the speakers list is not full: MM_C_SPK_APPEND_ON_PC else: MM_C_SPK_REMOVE_ON_PC MM_C_SPK_APPEND_ON_PC
D: Cancel Req. to Speak	MM_C_RTS_REMOVE_ON_PC and if it was the first in the RTS list: MM_C_RTS_FIRST_ON_PC	MM_C_RTS_REMOVE_ON_PC and if it was the first in the RTS list: MM_C_RTS_FIRST_ON_PC	MM_C_RTS_REMOVE_ON_PC and if it was the first in the RTS list: MM_C_RTS_FIRST_ON_PC	N/A.
D: Microphone Off	MM_C_SPK_REMOVE_ON_PC	MM_C_MICRO_ON_OFF	MM_C_MICRO_ON_OFF	MM_C_SPK_REMOVE_ON_PC
D: Comment Request	N/A.	N/A.	if the CR list is not full: MM_C_CR_ADD_ON_PC	N/A.
D: Cancel Comment Request	N/A.	N/A.	MM_C_CR_REMOVE_ON_PC	N/A.
C: Cancel all speakers	MM_C_RTS_CLEAR_ON_PC MM_C_SPK_CLEAR_ON_PC	MM_C_RTS_CLEAR_ON_PC MM_C_SPK_CLEAR_ON_PC	MM_C_RTS_CLEAR_ON_PC MM_C_CR_CLEAR_ON_PC MM_C_SPK_CLEAR_ON_PC MM_C_CS_CLEAR_ON_PC	MM_C_SPK_CLEAR_ON_PC
C: Cancel all requests	MM_C_RTS_CLEAR_ON_PC	MM_C_RTS_CLEAR_ON_PC	MM_C_RTS_CLEAR_ON_PC MM_C_CR_CLEAR_ON_PC	<None>

Note that a delegate does not really turns on its microphone, but he makes a Request to speak. Depending on the operation mode and the current lists, he is added to the SPK list or the RTS list. On this Request-to-Speak-event also a remark has to be made if the unit/delegate is in the Speakerslist but with the microphone off (which is possible with the function MM_C_SET_MICRO_ON_OFF, see section 3.3.1). In that case for all operation modes a MM_C_SPK_REMOVE_ON_PC update notification is first given for the current unit after which the update notifications according to the event matrix are generated.

4.2 MM General notifications

4.2.1 MM_C_SET_MIC_OPER_MODE_ON_PC

Purpose

Notifies the remote controller that the microphone operation-mode has changed on the CCU.

Notify structure with this update

The update comes with a structure as defined in section 3.2.5.

4.2.2 MM_C_SET_ACTIVE_MICS_ON_PC

Purpose

Notifies the remote controller that the number of active microphones has changed on the CCU.

Notify structure with this update

The update comes with a structure as defined in section 3.2.6.

4.2.3 MM_C_SET_SETTINGS_ON_PC

Purpose

Notifies the remote controller that there is a change in the global settings on the CCU.

Notify structure with this update

The update comes with a structure as defined in section 3.2.7

4.3 MM Speaker list notifications

The Microphone Management speaker list notifications reports the changes in the speakers list.

4.3.1 MM_C_MICRO_ON_OFF

Purpose

Notifies the remote controller that a microphone of a unit is turned on or off. This notification will be sent when a delegate turns its microphone on or off.

Notify structure with this update

The update comes with the following structure:

```
typedef struct
{
    WORD    wUnitId;
    WORD    wMicroId;
    WORD    wPrioId;
} MM_T_MICRO_ONOFF_ON_PC;
```

where:

wUnitId	Unit Identifier
wMicroId	Passes the status of the microphone. This parameter can be one of the following values: • MM_C_PC_MIC_ON • MM_C_PC_MIC_OFF • MM_C_PC_MIC_NONE
wPrioId	Passes the prio-status of the chairman unit. This priority information indicates to the remote controller that the delegate units can be muted due to a priority key pressed on this chairman-unit. Although the microphone is turned on, the delegate can not yet speak. This parameter can be one of the following values: • MM_C_PC_PRIO_ON

- MM_C_PC_PRIO_OFF
- MM_C_PC_PRIO_NONE

The 'NONE' values of the parameters 'wMicroId' and 'wPrioId' indicate that the specific parameter is not used.

Examples

To illustrate the values of the parameters 'wMicroId' and 'wPrioId' the following value for these parameters are returned with the events:

	wMicroId	wPrioId
Delegate micro ON	MM_C_PC_MIC_ON	MM_C_PC_PRIO_NONE
Delegate micro OFF	MM_C_PC_MIC_OFF	MM_C_PC_PRIO_NONE
Chairman micro ON (no Prio)	MM_C_PC_MIC_ON	MM_C_PC_PRIO_NONE
Chairman micro OFF (no Prio)	MM_C_PC_MIC_OFF	MM_C_PC_PRIO_NONE
Chairman prio ON (no micro)	MM_C_PC_MIC_ON	MM_C_PC_PRIO_ON
Chairman prio OFF (no micro)	MM_C_PC_MIC_OFF	MM_C_PC_PRIO_OFF
Chairman prio ON (with micro on)	MM_C_PC_MIC_NONE	MM_C_PC_PRIO_ON
Chairman prio OFF (with micro on)	MM_C_PC_MIC_NONE	MM_C_PC_PRIO_OFF
Chairman prio ON (with other prio on)	MM_C_PC_MIC_ON	MM_C_PC_PRIO_ON
Chairman prio OFF (with other prio on)	MM_C_PC_MIC_OFF	MM_C_PC_PRIO_OFF

4.3.2 MM_C_NR_CHAIR_MICS_ON

Purpose

Notifies the remote controller that there are still chairmen which have pressed their micro or priority key on the unit.

Note: This notification is used to handle speech-time correctly (controlled by the DCN-control PC). E.g. The delegates speech-time must be held when at least one chairman is speaking.

Notify structure with this update

The update comes with the following structure:

```
WORD    wNrOfChairMicsOn;
```

where:

wNrOfChairMicsOn The number of chairmen, which are speaking.

4.3.3 MM_C_SPK_SET_ON_PC

Purpose

Notifies the remote controller that the CCU has a complete new list of SPK entries.

Notify structure with this update

The update comes with the structure defined in 3.3.5.

4.3.4 MM_C_SPK_CLEAR_ON_PC

Purpose

Notifies the remote controller that the SPK list is cleared.

Notify structure with this update

The update does not have any additional parameters.

4.3.5 MM_C_SPK_APPEND_ON_PC

Purpose

Notifies the remote controller that a unit is added to the SPK list.

Notify structure with this update

The update comes with the following structure:

```
MM_T_SPK    tSpkAdd;
```

where:

tSpkAdd The speaker who is added to the speakers list. The structure MM_T_SPK is defined in section 3.3.2.

4.3.6 MM_C_SPK_REMOVE_ON_PC

Purpose

Notifies the remote controller that a unit is removed from the SPK list (including turning off the microphone).

Notify structure with this update

The update comes with the following structure:

```
MM_T_SPK     tSpkRemove;
```

where:

tSpkRemove	The speaker who is removed from the speakers list. The structure MM_T_SPK is defined in section 3.3.2.

4.3.7 MM_C_SPK_INSERT_ON_PC

Purpose

Notifies the remote controller that a speaker is inserted before another speaker.

Notify structure with this update

The update comes with the following structure:

```
typedef struct
{
    MM_T_SPK     tSearchSpk;
    MM_T_SPK     tNewSpk;
} MM_T_SPK_INSERT;
```

where:

tSearchSpk	The speaker entry to search for. The new Speaker entry ('tNewSpk') shall be inserted before this Speaker.
tNewSpk	The Speaker entry to be added to the list.

4.3.8 MM_C_SPK_REPLACE_ON_PC

Purpose

Notifies the remote controller that a speaker is replaced by another speaker.

Notify structure with this update

The update comes along with the following structure:

```
typedef struct
{
    MM_T_SPK     tCurrSpk;
    MM_T_SPK     tNewSpk;
} MM_T_SPK_REPLACE;
```

where:

tCurrSpk	The SPK entry to search for. This SPK entry is replaced by the new value given in the parameter 'tNewSpk'.
tNewSpk	The SPK entry holding the new contents.

4.4 MM Comment Speaker list notifications

The Microphone Management comment speaker list notifications report the changes in the comment speakers list.

4.4.1 MM_C_CS_CLEAR_ON_PC

Purpose

Notifies the remote controller that the CS list is cleared.

Notify structure with this update

The update does not have any additional parameters.

4.4.2 MM_C_CS_ADD_ON_PC

Purpose

Notifies the remote controller that a unit is added to the CS list.

Notify structure with this update

The update comes with the following structure:

```
MM_T_SPK     tCSpkAdd;
```

where:

tCSpkAdd	The speaker who is added to the comment speakers list. The structure MM_T_SPK is defined in section 3.3.2.

4.4.3 MM_C_CS_REMOVE_ON_PC

Purpose

Notifies the remote controller that a unit is removed from the SPK list (including turning off the microphone).

Notify structure with this update

The update comes with the following structure:

```
MM_T_SPK    tCSpkRemove;
```

where:

tCSpkRemove	The speaker who is removed from the comment speakers list. The structure MM_T_SPK is defined in section 3.3.2.

4.5 MM Notebook list notifications

The Microphone Management notebook notifications report the remote controller the changes in the NBK-list.

4.5.1 MM_C_NBK_REMOVE_ON_PC

purpose

Notifies the remote controller that a notebook unit is removed from the NBK list.

Notify structure with this update

The update comes with the following structure:

```
typedef struct
{
    WORD    wUnitId;
    WORD    wMicroType;
} MM_T_NBK;
```

where:

wUnitId	Unit identifier
wMicroType	The type of microphone handling for the notebook entry as defined in 3.5.3

4.5.2 MM_C_NBK_SET_ON_PC

purpose

Notifies the remote controller that the CCU has a complete new notebook list. Note that all chairmen units will be included inside the notebook list.

Notify structure with this update

The update comes with the structure defined as response structure in section 3.5.3.

4.6 MM Request to Speak list notifications

The Microphone Management request to speak notifications report the remote controller the changes in the RTS-list.

4.6.1 MM_C_RTS_SET_ON_PC

Purpose

Notifies the remote controller that the CCU has a complete new list of request to speak delegates/units. Note that this notification implies a change of the first RTS entry in the list.

Notify structure with this update

The update comes with the structure defined in 3.6.4.

4.6.2 MM_C_RTS_CLEAR_ON_PC

Purpose

Notifies the remote controller that the RTS list is cleared.

Notify structure with this update

The update does not have any additional parameters.

4.6.3 MM_C_RTS_REMOVE_ON_PC

Purpose

Notifies the remote controller that a delegate/unit combination is removed from the RTS list.

Notify structure with this update

The update comes along with a MM_T_RTS structure which indicates the delegate/unit combination to be removed. The structure MM_T_RTS is defined in section 3.6.

4.6.4 MM_C_RTS_FIRST_ON_PC

Purpose

Notifies the remote controller which delegate/unit combination is the first in the list. When the UnitId and DelegateId fields of the structure are filled with DBSC_EMPTY_UNIT and DBSC_EMPTY_DELEGATE respectively, the first RTS entry becomes invalid. The last results into a empty RTS list.
Note that this notification invalidates the previous notification about the first RTS list entry.

Notify structure with this update

The update comes with the following structure:

```
MM_T_RTS     tRtsFirst;
```

where:

tRtsFirst	The RTS list entry which is now at the top of the RTS list.

4.6.5 MM_C_RTS_INSERT_ON_PC

Purpose

Notifies the remote controller that a delegate/unit combination is inserted in the RTS list before another RTS entry. This notification is sent for both an insertion between two RTS entries as a append of a RTS entry to the end of the RTS.

Notify structure with this update

The update comes along with the following structure:

```
typedef struct
{
    MM_T_RTS     tSearchRts;
    MM_T_RTS     tNewRts;
} MM_T_RTS_INSERT;
```

where:

tSearchRts	The RTS entry to search for. The new RTS entry ('tNewRts') shall be inserted before this RTS entry. When the elements of the entry are filled with empty values, then the entry 'tNewRts' will be added to the end of the list.
tNewRts	The RTS entry to be added to the list.

Note that an append of the new RTS entry will be done when the elements of this parameter are filled with empty values like:

```
tSearchRts.wUnitId = DBSC_EMPTY_UNIT;
tSearchRts.wDelegateId = DBSC_EMPTY_DELEGATE;
```

4.6.6 MM_C_RTS_REPLACE_ON_PC

Purpose

Notifies the remote controller that a delegate/unit combination is replaced by a new RTS entry.

Notify structure with this update

The update comes along with the following structure:

```
typedef struct
{
    MM_T_RTS     tCurrRts;
    MM_T_RTS     tNewRts;
} MM_T_RTS_REPLACE;
```

where:

tCurrRts	The RTS entry to search for. This RTS entry is replaced by the new value given in the parameter 'tNewRts'.
tNewRts	The RTS entry holding the new contents.

4.7 MM Comment Request list notifications

The Microphone Management Comment Request notifications report the remote controller the changes in the CR list.

4.7.1 MM_C_CR_CLEAR_ON_PC

Purpose

Notifies the remote controller that the CR list is cleared.

Notify structure with this update

The update does not have any additional parameters.

4.7.2 MM_C_CR_ADD_ON_PC

Purpose

Notifies the remote controller that a delegate/unit combination is added to the CR list.

Notify structure with this update

The update comes along with a MM_T_RTS structure which indicates the delegate/unit combination to be removed. The structure MM_T_RTS is defined in section 3.6.

4.7.3 MM_C_CR_REMOVE_ON_PC

Purpose

Notifies the remote controller that a delegate/unit combination is removed from the CR list.

Notify structure with this update

The update comes along with a MM_T_RTS structure which indicates the delegate/unit combination to be removed. The structure MM_T_RTS is defined in section 3.6.

4.7.4 MM_C_CR_REPLACE_ON_PC

Purpose

Notifies the remote controller that a delegate/unit combination is replaced by a new CR entry.

Notify structure with this update

The update comes along with a MM_T_RTS_REPLACE structure which indicates the delegate/unit combination to be removed and the delegate/unit combination to be added. The structure MM_T_RTS_REPLACE is defined in section 4.6.6.

4.8 MM Speechtime notifications

The Microphone Management speechtime notifications report the remote controller the changes in the Speechtime setting.

4.8.1 MM_C_TIMER_ON_OFF

Purpose

Notifies the controller that there is a change in using/not using of the speech timer.

Notify structure with this update

The update does not have any additional parameters.

APPENDIX A. VALUES OF THE DEFINES

In this document a lot of definitions are given, which have values connected to them. In this appendix all defines will be connected to their values;
The values are presented in 'C'-syntax

```
#define MKWORD(lb,hb)           ((WORD)(((WORD)(hb))<<8) | (WORD)(lb)))
#define DCNC_APP_MM             0

#define MM_C_MICRO_ON_OFF                ( MKWORD (1 , DCNC_APP_MM) )
#define MM_C_CHAIR_MICS_ON               ( MKWORD (2 , DCNC_APP_MM) )
#define MM_C_TIMER_ON_OFF                ( MKWORD (3 , DCNC_APP_MM) )
#define MM_C_RTS_SET_ON_PC               ( MKWORD (4 , DCNC_APP_MM) )
#define MM_C_RTS_CLEAR_ON_PC             ( MKWORD (5 , DCNC_APP_MM) )
#define MM_C_RTS_REMOVE_ON_PC            ( MKWORD (6 , DCNC_APP_MM) )
#define MM_C_RTS_INSERT_ON_PC            ( MKWORD (7 , DCNC_APP_MM) )
#define MM_C_RTS_REPLACE_ON_PC           ( MKWORD (8 , DCNC_APP_MM) )
#define MM_C_NBK_SET_ON_PC               ( MKWORD (9,  DCNC_APP_MM) )
#define MM_C_NBK_CLEAR_ON_PC             ( MKWORD (10, DCNC_APP_MM) )
#define MM_C_NBK_REMOVE_ON_PC            ( MKWORD (11, DCNC_APP_MM) )
#define MM_C_SPK_SET_ON_PC               ( MKWORD (12, DCNC_APP_MM) )
#define MM_C_SPK_CLEAR_ON_PC             ( MKWORD (13, DCNC_APP_MM) )
#define MM_C_SPK_APPEND_ON_PC            ( MKWORD (14, DCNC_APP_MM) )
#define MM_C_SPK_REMOVE_ON_PC            ( MKWORD (15, DCNC_APP_MM) )
#define MM_C_SPK_INSERT_ON_PC            ( MKWORD (16, DCNC_APP_MM) )
#define MM_C_SPK_REPLACE_ON_PC           ( MKWORD (17, DCNC_APP_MM) )
#define MM_C_SET_MIC_OPER_MODE_ON_PC     ( MKWORD (18, DCNC_APP_MM) )
#define MM_C_SET_ACTIVE_MICS_ON_PC       ( MKWORD (19, DCNC_APP_MM) )
#define MM_C_RTS_FIRST_ON_PC             ( MKWORD (20, DCNC_APP_MM) )
#define MM_C_SET_SETTINGS_ON_PC          ( MKWORD (21, DCNC_APP_MM) )
#define MM_C_CR_CLEAR_ON_PC              ( MKWORD (22, DCNC_APP_MM) )
#define MM_C_CR_ADD_ON_PC                ( MKWORD (23, DCNC_APP_MM) )
#define MM_C_CR_REMOVE_ON_PC             ( MKWORD (24, DCNC_APP_MM) )
#define MM_C_CR_REPLACE_ON_PC            ( MKWORD (25, DCNC_APP_MM) )
#define MM_C_CS_CLEAR_ON_PC              ( MKWORD (26, DCNC_APP_MM) )
#define MM_C_CS_ADD_ON_PC                ( MKWORD (27, DCNC_APP_MM) )
#define MM_C_CS_REMOVE_ON_PC             ( MKWORD (28, DCNC_APP_MM) )

#define MM_C_START_MM                    ( MKWORD (30, DCNC_APP_MM) )
#define MM_C_STOP_MM                     ( MKWORD (31, DCNC_APP_MM) )
#define MM_C_GET_SETTINGS                ( MKWORD (32, DCNC_APP_MM) )
#define MM_C_SET_SETTINGS                ( MKWORD (33, DCNC_APP_MM) )
#define MM_C_SET_MICRO_ON_OFF            ( MKWORD (34, DCNC_APP_MM) )
#define MM_C_SHIFT                       ( MKWORD (35, DCNC_APP_MM) )
#define MM_C_RTS_GET                     ( MKWORD (37, DCNC_APP_MM) )
#define MM_C_RTS_CLEAR                   ( MKWORD (38, DCNC_APP_MM) )
#define MM_C_RTS_REMOVE                  ( MKWORD (39, DCNC_APP_MM) )
#define MM_C_NBK_SET                     ( MKWORD (42, DCNC_APP_MM) )
#define MM_C_NBK_GET                     ( MKWORD (43, DCNC_APP_MM) )
#define MM_C_NBK_CLEAR                   ( MKWORD (44, DCNC_APP_MM) )
#define MM_C_NBK_REMOVE                  ( MKWORD (45, DCNC_APP_MM) )
#define MM_C_SPK_GET                     ( MKWORD (46, DCNC_APP_MM) )
#define MM_C_SPK_CLEAR                   ( MKWORD (47, DCNC_APP_MM) )
#define MM_C_SPK_APPEND                  ( MKWORD (48, DCNC_APP_MM) )
#define MM_C_SPK_REMOVE                  ( MKWORD (49, DCNC_APP_MM) )
#define MM_C_SET_MIC_OPER_MODE           ( MKWORD (52, DCNC_APP_MM) )
#define MM_C_SET_ACTIVE_MICS             ( MKWORD (53, DCNC_APP_MM) )
#define MM_C_SET_SPEECHTIME_SETTINGS     ( MKWORD (59, DCNC_APP_MM) )
#define MM_C_LAST_MINUTE_WARNING         ( MKWORD (60, DCNC_APP_MM) )
#define MM_C_TIME_FINISHED_WARNING       ( MKWORD (61, DCNC_APP_MM) )
#define MM_C_RTS_APPEND                  ( MKWORD (62, DCNC_APP_MM) )
#define MM_C_CR_REMOVE                   ( MKWORD (64, DCNC_APP_MM) )
#define MM_C_SHIFT_CR                    ( MKWORD (65, DCNC_APP_MM) )
#define MM_C_CR_GET                      ( MKWORD (66, DCNC_APP_MM) )
#define MM_C_CS_REMOVE                   ( MKWORD (67, DCNC_APP_MM) )
#define MM_C_CS_GET                      ( MKWORD (68, DCNC_APP_MM) )
#define MM_C_START_MON_MM                ( MKWORD (69, DCNC_APP_MM) )
#define MM_C_STOP_MON_MM                 ( MKWORD (70, DCNC_APP_MM) )

#define MM_C_PC_MIC_ON                   1
#define MM_C_PC_MIC_OFF                  2
```

```
#define MM_C_PC_MIC_NONE                3
#define MM_C_PC_PRIO_ON                 1
#define MM_C_PC_PRIO_OFF                2
#define MM_C_PC_PRIO_NONE               3

#define MM_C_VIP_CHAIRMAN        1  (Chairman)
#define MM_C_VIP_KEY             2  (Delegate set as Key activated
                                    notebooker)
#define MM_C_VIP_OPERATOR        3  (Delegate set as Operator activated
                                    notebooker)
#define MM_C_VIP_VOICE           4  (Delegate set as Voice activated
                                    notebooker)
#define MM_C_VIP_VCHAIR          5  (Chairman set as Voice activated)
#define MM_C_CHAIRMAN_NO_AC      6  (Chairman exclude from Access Control)
#define MM_C_KEY_NO_AC           7  (Key Activated Delegate excluded from
                                    Access Control)
#define MM_C_OPERATOR_NO_AC      8  (Operator Activated Delegate excluded
                                    from Access Control)
#define MM_C_VOICE_NO_AC         9  (Voice Activated Delegate excluded
                                    from Access Control)
#define MM_C_VCHAIR_NO_AC       10  (Voice Activated Chairman excluded
                                    from Access Control)


#define MM_C_OPERATOR_WITH_REQ_LIST          0
#define MM_C_DELEGATE_WITH_REQ_LIST          1
#define MM_C_DELEGATE_WITH_OVERRIDE          2
#define MM_C_DELEGATE_WITH_VOICE             3
#define MM_C_OPERATOR_WITH_COMMENT_LIST      4


#define DBSC_MAX_SPEAKERLIST                 4
#define DBSC_MAX_NOTEBOOKLIST               15
#define DBSC_MAX_DELRTS                    100
#define DBSC_MAX_DELCR                       5
#define DBSC_MAX_DELCS                       1


#define DBSC_EMPTY_UNIT                 (0xFFFF)
#define DBSC_EMPTY_DELEGATE             (0xFFFF)
```

APPENDIX B. ERROR CODES

Responses returned upon a remote function request contain a error field ('wError'). In this appendix an overview is given of the possible errors and their values.

Microphone Management Error code Explanation	Value
MM_E_NOERROR The execution of the remote function was successful.	0
MM_E_OPEN_CLOSE_FAILED The internal database on the CCU was not able to update the total use count for the MM application.	5
MM_E_UNIT_ALREADY_PRESENT The unit to be added to the list (RTS or SPK) is already present in that list.	6
MM_E_NOT_PRESENT The record to search for in the list (Comment Request) is not present in the list.	8
MM_E_UNIT_NOT_PRESENT The unit to search for in the list (RTS or SPK) is not present in the list.	9
MM_E_NOT_IN_SPL_OR_NOB You tried to turn off a microphone of a unit which was not present in either the speakers list or the notebook list.	15
MM_E_ILLEGAL_MAX_ACT_MICS The number provided for the maximum number of active microphones is illegal with respect to the current Operation Mode. Valid value for the mode MM_C_OPERATOR_WITH_COMMENT_LIST is 1. Valid values for the mode MM_C_DELEGATE_WITH_VOICE are within the range 2..4 and for all other modes in the range 1..4.	17
MM_E_ILLEGAL_MIC_OPER_MODE The function requested is illegal for the current operation mode. The function is not executed.	18
MM_E_UNKNOWN_UNITID_AND_DELID You have provided a RTS list entry with both elements (UnitId and DelegateId) set to empty values (DBSC_EMPTY_UNIT, DBSC_EMPTY_DELEGATE). At least one of the elements must be defined to fulfil the function.	19
MM_E_DELETE_RTS_LIST_FAILED A delete of a RTS list entry in the internal database failed. Probably illegal values for either the elements UnitId or DelegateId are passed.	21
MM_E_INSERT_RTS_LIST_FAILED The CCU was not able to insert the RTS list entry into the internal database. Probably illegal values for either the elements UnitId or DelegateId are passed.	22
MM_E_RTS_LIST_FULL The RTS list is full. No more RTS entries can be added using the function MM_C_RTS_APPEND.	24
MM_E_RTS_LIST_CHANGED During a reduction of the maximum length of the RTS list, the database was unable to retrieve the last RTS list entry. The actual length is not changed. To recover this error; clear the RTS list, set the new RTS list length and set the new contents in the RTS list.	25
MM_E_RTS_LIST_EMPTY The RTS list is empty, therefore the function cannot be fulfilled. e.g.	26

Microphone Management Error code Explanation	Value
remove on a RTS list entry on an empty RTS list.	
MM_E_ILLEGAL_MAX_RTS_LIST_LEN The maximum length provided for the RTS list is out of range. Valid values for the RTS list length are within the range 0..100.	27
MM_E_RTS_LIST_TOO_BIG The RTS list provided is too big to store it. None of the RTS entries provided is put into the RTS list and the old RTS list remains active.	28
MM_E_DELETE_SPEAKERS_LIST_FAILED A delete of a SPK list entry in the internal database failed. Probably an illegal value for the element UnitId is passed.	31
MM_E_INSERT_SPEAKERS_LIST_FAILED The CCU was not able to insert the SPK list entry into the internal database. Probably an illegal value for the element UnitId is passed.	32
MM_E_SPEAKERS_LIST_FULL The SPK list is full. No more SPK entries can be added using the function MM_C_SPK_APPEND.	34
MM_E_ILLEGAL_MICRO_TYPE This unit is also present in the Notebook and has a microtype that is not allowed in the speakers list.	47
MM_E_UNIT_NOT_CONNECTED The unit is not connected to the system (any more).	48
MM_E_UNITID_DELID_MISMATCH The unit and delegate do not match with each other according to the database on the CCU.	49

APPENDIX C. EXAMPLES

In the example below the remote functions and update notifications, that are defined in this document as constant values for the wFnId parameter of the message (see [SRS_INF]), are presented as functions described in a 'C' syntax. The parameter structures of these functions are according the input, output or notify structures described in the appropriate section.
For every function is assumed that the function will create his structure, transport the parameters to the CCU and waits for the result information coming from the CCU.
For both the remote functions as the update notifications the same names are used as their identifier, but without the constant mark "C" and using mixed case names. So, e.g. remote function MM_C_SET_SETTINGS shall be referenced as function as:

MM_Set_Settings (MM_T_CCU_GLOBAL_SETTINGS tMMSettings);

C.1. Microphone Management Control

This example shows the minimum steps to be taken for controlling the MM application.
First we have to start the MM application inside the CCU.

```
WORD    wNrOfInstances;

error = MM_Start_MM(&wNrOfInstances);
if (error != MM_E_NOERROR)
{
    /* do error handling */
}
else
{
    switch (wNrOfInstances)
    {
        case 0 :/* something went wrong with registering for remote interface
                so, do error handling */
            break;
        case 1 :/* OK */
            break;
        default : /* 2 or more. This means there are more remote controllers
            identified by the CCU. Stop as many times as needed */
            WORDwNewNumber;
            do
            {
                MM_Stop_MM(&wNewNumber);
            } while (wNewNumber > 1);
            break;
    }
}
```

If there are no errors on starting the MM application the next thing we are interested in are the settings. Assume that we want the system to operate in a Operator with RTS list mode, 4 active mics and a maximum RTS list length of 50. The first thing to do is retrieve the current settings, then check them against the wanted settings and, if they are not the same, set the new settings.
The results in the following control flow:

```
/* declare variables */
MM_T_CCU_GLOBAL_SETTINGS tMMSettings;
BOOLEAN          bMustSend = FALSE;

/* retrieve the current settings */
MM_Get_Settings(&tMMSettings);

/* and check if they are what we want *?
if (tMMSettings.wOperationMode != MM_C_OPERATOR_WITH_REQ_LIST)
{
    tMMSettings.wOperationMode = MM_C_OPERATOR_WITH_REQ_LIST;
    bMustSend = TRUE;
}
if (tMMSettings.wActiveMics != 4)
{
    tMMSettings.wActiveMics = 4);
    bMustSend = TRUE;
}
if (tMMSettings.wMaxRTSListLen != 50)
{
```

```
        tMMSettings.wMaxRTSListLen = 50;
        bMustSend = TRUE;
    }

    /* Set new settings if we have to */
    if (bMustSend)
    {
        error = MM_Set_Settings(&tMMSettings);
        if (error != MM_E_NOERROR)
        {
            /* do error handling */
        }
    }
```

Setting new settings also results in an update notification, so the last thing to do is to check if our settings are accepted by the CCU.
Therefor, we need the following function:

```
    void MM_Set_Settings_On_Pc(MM_T_CCU_GLOBAL_SETTINGS tNotifiedSettings)
    {
        BOOLEAN  bIdentical = FALSE;

    /* assume we have a user defined function to compare both settings structures */
        bIdentical = MyCompareSettings(tNotifiedSettings, tMMSettings);

        if (bIdentical == FALSE)
        {
            /*
               If they are not the same:
               Either update your local settings with the CCU settings
               or try to set them again
            */
        }
    }
```

Once the settings are known, we could retrieve the current notebook-, speakers- and RTS list and wait for the updates to monitor the microphone status in the conference hall, or send remote functions to influence that status.
When the congress is finished we must tell the CCU that we stopped monitoring the MM application, using the following function:

```
    WORD    wNrOfInstances;

    error = MM_Stop_MM(&wNrOfInstances);
    if (error != MM_E_NOERROR)
    {
        /* do error handling */
    }
    else
    {
        switch (wNrOfInstances)
        {
            case 0 :/* OK */
                break;
            default : /* 1 or more. This means there are still remote controllers
                identified by the CCU. Stop as many times as needed */
                WORDwNewNumber;
                do
                {
                    MM_Stop_MM(&wNewNumber);
                } while (wNewNumber != 0);
                break;
        }
    }
```

This ends controlling the MM application. The remote controller and CCU can now safely be switched off.

For more information please visit www.boschsecuritysystems.com

December 2003 | MM Remote Interface Description

BOSCH