SpeakerCraft MRA Remote Management Guide

Rev	Date	Modified By	Description of Changes
1.0	Jul 8, 2014	Joseph Benoit	Initial version based on *SpeakerCraft MRA UART Protocol v1.1*

Table of Contents

1 Introduction

This document outlines the SPEAKERCRAFT MRA Remote Management Mode:

- How to enable & disable remote management mode
- How to send remote management mode messages
- Detailed information regarding management message formats

1.1 Assumptions

Familiarity with the following is assumed:

- Operation of the SPEAKERCRAFT MRA
- Socket communication over TCP/IP

1.2 Conventions

Wherever possible the values are displayed in decimal, hexadecimal, and binary format.

2 IP Control

A single SPEAKERCRAFT MRA is manageable via sending messages to the device's IP address. Multiple devices may be controlled by sending management messages to the unit's respective IP addresses.

It is assumed that the SPEAKERCRAFT MRA device and the managing host are on the same LAN.

2.1 Enabling/Disabling SPEAKERCRAFT MRA Remote Management Mode

SPEAKERCRAFT MRA Remote Management may be enabled or disabled via sending a UDP message (To port 444) sent to the SPEAKERCRAFT MRA's IP address:

```
ENABLE  remote management data stream:  0x08 0x00 0x00 0x00 0xFF 0xEE 0x00 0xBB

DISABLE remote management data stream:  0x08 0x00 0x00 0x00 0xDD 0xCC 0x11 0xAA
```

The function `cfgRemoteMgmtMode()` is used by sample code remMgmt.c to send and receive management messages. Refer to the sample code, included in the last section of this document, for more information.

2.2 SPEAKERCRAFT MRA Management Messages

Prior to sending management messages, SPEAKERCRAFT MRA remote management must be ENABLED.

Management messages are sent via TCP over port 10200.

The function `sendMgmtReqRecvMgmtRsp()` is used by sample code remMgmt.c to send and receive management messages. Refer to the sample code, included in the last section of this document, for more information.

3 Management Message Format

3.1 Request Format

The request format is as follows:

Field	Size	Description
Sync 1	1	Sync byte 1, Value = 255(0xFF)
Sync 2	1	Sync byte 2, Value = 85(0x55)
Length Hi	1	Payload length. High byte of two byte count of command and data bytes
Length Lo	1	Payload length. Low byte of two byte count of command and data bytes
Command	1	Command ID
Data	n	Variable length data
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

3.2 Response Format (If Result Code = 0 or Result Code = 1)

The response format is as follows:

Field	Size	Description
Sync 1	1	Sync byte 1, Value = 255(0xFF)
Sync 2	1	Sync byte 2, Value = 85(0x55)
Length Hi	1	Payload length. High byte of two byte count of command, result, and data bytes
Length Lo	1	Payload length. Low byte of two byte count of command, result, and data bytes
Command	1	Command ID (Same ID as request message)
Result	1	0(0x00) = Success/OK 1(0x01) = Data Returned
Data	n	Variable length data (only if Result is 1(0x01))
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

3.3 Response Format (If Result Code >= 251 or Result Code <= 255)

The response format is as follows:

Field	Size	Description
Sync 1	1	Sync byte 1, Value = 255(0xFF)
Sync 2	1	Sync byte 2, Value = 85(0x55)
Length Hi	1	Payload length. High byte of two byte count of result byte
Length Lo	1	Payload length. Low byte of two byte count of result byte
Result	1	252(0xFC) = Invalid Command (command not defined/implemented) 254(0xFE) = Invalid checksum
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

3.4 Message Building

The function `buildMgmtMsg()` is used by sample code remMgmt.c to build a management message:

```
/*=============================================================================*/

unsigned char * buildMgmtMsg(int *                  iMsgLen,        // OUTPUT
                             unsigned char          ucCmd,          // INPUT
                             unsigned short         ushPayloadLen,  // INPUT
                             unsigned char *        pPayload)       // INPUT

/*=============================================================================*/
{
    unsigned char   *pMsg = 0;
    unsigned short  ushCheckSum = 0;
    int iCount = 0;

    if(iMsgLen == 0)
        {
        return 0;
        }

    *iMsgLen = ushPayloadLen + 6;

    pMsg = (unsigned char *)malloc(*iMsgLen);

    if((pPayload == 0) && (ushPayloadLen))
        {
        *iMsgLen = 0;
        return 0;
        }

    memset(pMsg, 0x00, *iMsgLen);

    pMsg[0] = 0xFF; // First  sync byte
    pMsg[1] = 0x55; // Second sync byte
    pMsg[2] = (ushPayloadLen + 1) >> 8;
    pMsg[3] = (ushPayloadLen + 1) & 0x00FF;
    pMsg[4] = ucCmd;

    ushCheckSum = (unsigned short)pMsg[2] + (unsigned short)pMsg[3] +(unsigned short)pMsg[4];

    for(iCount = 0; iCount < ushPayloadLen; iCount++)
    {
        pMsg[iCount + 5] = pPayload[iCount];
        ushCheckSum += (unsigned short)pMsg[iCount + 5];
    }

    pMsg[*iMsgLen - 1] = (unsigned char)(0x100 - (ushCheckSum & 0x00FF));

    return(pMsg);
}
```

The sample code remMgmt.c is included in the last section of this document.

3.5 Sequence of Operation

Operations are executed with this sequence:

- Request send to the device
- Device initiates the operation
- Response sent to the requesting host before operation is completed

3.6 Consecutive Operations

In most cases a managing host may initiate a request immediately upon receiving a response from the previous request.

Some operations require a number of milliseconds to complete. During this time the device will be unable to accept any additional requests. Operations which fall into this category will be identified in their respective section.

4 Sample Utility

The sample program remMgmt.c is provided allow a developer to experiment with the SPEAKERCRAFT MRA Remote Management API. The program is intended to be portable across native Linux versions. It will also work under Cygwin.

Output examples of the compiled sample program remMgmt.exe are used throughout this document. For these examples, the executable remMgmt.exe was generated using GCC 4.8.1 and run under Cygwin.

Enable Remote Management Example

```
$ ./remMgmt.exe 192.168.0.198 on
Remote management mode ON
```

Disable Remote Management Example

```
$ ./remMgmt.exe 192.168.0.198 off
Remote management mode OFF
```

4.1 Trouble Shooting

4.1.1 Help text

```
$ ./remMgmt.exe
Usage:
./remMgmt <ipaddr> on  (  Enables remote management mode)
./remMgmt <ipaddr> off ( Disables remote management mode)
./remMgmt <ipaddr> <cmd> <data 1> ... <data n> (DECIMAL VALUES ONLY)

$ ./remMgmt.exe ?
Usage:
./remMgmt <ipaddr> on  (  Enables remote management mode)
./remMgmt <ipaddr> off ( Disables remote management mode)
./remMgmt <ipaddr> <cmd> <data 1> ... <data n> (DECIMAL VALUES ONLY)

$ ./remMgmt.exe <ipaddr> ?
Usage:
./remMgmt <ipaddr> on  (  Enables remote management mode)
./remMgmt <ipaddr> off ( Disables remote management mode)
./remMgmt <ipaddr> <cmd> <data 1> ... <data n> (DECIMAL VALUES ONLY)
```

4.1.2 Forget To Enable Remote Management Mode

You've forgotten to enable remote management mode if you send a message to a valid IP address and get the following error:

```
$ ./remMgmt.exe 192.168.0.198 0
ERROR, connecting to TCP socket
```

5 Factory Default Settings

Setting	Default	How to change default
Standby Mode	Enabled	5(0x05) Set Standby Mode
Volume	35	48(0x30) Set Default Volume
Maximum Volume	100	50(0x32) Set Maximum Volume
Treble/Bass	+0 dB	52(0x34) Set Default Tone Control
Loudness	Off	52(0x34) Set Default Tone Control
Default Tone Setting	Use Default	52(0x34) Set Default Tone Control
Do Not Disturb	Off	**Not possible**
Input Level	+0 dB	54(0x36) Set Input Level
Zone Preamp Output Mode	Variable	56(0x38) Set Zone Preamp Output Mode
Paging Zones	All Zones	64(0x40) Set Paging Zones
Paging Volume	35	66(0x42) Set Paging Volume
Whole House Music Zones	All Zones	74(0x4A) Set Whole House Music Zones
Default Routing Setup (Power On State)	Input 1 to Output 1 Input 2 to Output 2 … Input 6 to Output 6 (Test Mode Enabled)	58(0x3A) Set Startup Mode

Settings can be reset to factory defaults with the **7(0x07) Reset Default Settings** command.

If Default Tone Setting is changed to “Use Last Setting”, the system must remember the tone settings set with **34(0x22) Set Tone Control** and use those settings for zone and unit power on. Otherwise, if the Default Tone Setting is “Use Default”, the tone settings set with **52(0x34) Set Default Tone Control** are used.

6 Command Groups

This section outlines how commands to set/get configuration data, as well as commands to initiate operations.

6.1 Status Group

6.1.1 Get System Version - Command 0(0x00)

- This command requests the firmware version
- The version number is expected to be in the format of <major>.<minor>.<subversion>.<build>

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	1(0x01)
Command	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	6(0x06)
Command	1	0(0x00)
Result	1	1(0x01)
Data 1	**1**	**Major version number**
Data 2	**1**	**Minor version number**
Data 3	**1**	**Subversion number**
Data 4	**1**	**Build number**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 0

req msg: length 0006 bytes 255 085 000 001 000 255
             cmd 0000
```

```
rsp msg: length 0011 bytes 255 085 000 006 000 001 001 011 008 000 229
            cmd 0000 result 0001 payload 001 011 008 000
```

6.1.2 Get Audio Sense State - Command 3(0x03)

- This command queries the audio sense state on each of the external six audio inputs

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	1(0x01)
Command	1	3(0x03)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Command	1	3(0x03)
Result	1	1(0x01)
Data 1	**1**	**Bitmask indicating whether or not an input has audio:** **Bit 7: Audio input 1** **Bit 6: Audio input 2** **Bit 5: Audio input 3** **Bit 4: Audio input 4** **Bit 3: Audio input 5** **Bit 2: Audio input 6** **Bit 1: Paging input** **Bit 0: Not used**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 3

req msg: length 0006 bytes 255 085 000 001 003 252
            cmd 0003

rsp msg: length 0008 bytes 255 085 000 003 003 001 128 121
            cmd 0003 result 0001 payload 128
```

128(0x80) - 0b10000000 (Audio detected on input 1)

6.1.3 Get Protection State - Command 4(0x04)

- This command queries the protection state on each of the audio outputs. The protection state is split into thermal and overload protection.
- The output is in the overload protection state if too much current is being drawn and the amps shut down. The audio output is in the thermal protection state is if the unit gets too hot and the amps shut down.

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	1(0x01)
Command	1	4(0x04)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	4(0x04)
Command	1	4(0x04)
Result	1	1(0x01)
Data 1	**1**	**Bitmask indicating whether or not an output is in the thermal protection state:** **Bit 7: Audio output 1** **Bit 6: Audio output 2** **Bit 5: Audio output 3** **Bit 4: Audio output 4** **Bit 3: Audio output 5** **Bit 2: Audio output 6** **Bit 1: Not used** **Bit 0: Not used**
Data 2	**1**	**Bitmask indicating whether or not an output is in the overload protection state:** **Bit 7: Audio output 1** **Bit 6: Audio output 2** **Bit 5: Audio output 3** **Bit 4: Audio output 4** **Bit 3: Audio output 5** **Bit 2: Audio output 6** **Bit 1: Not used** **Bit 0: Not used**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 4

req msg: length 0006 bytes 255 085 000 001 004 251
            cmd 0004
```

```
rsp msg: length 0009 bytes 255 085 000 004 004 001 016 032 247
            cmd 0004 result 0001 payload 016 032

016(0x10) - 0b00010000 (Audio output 4 in thermal state)
032(0x20) - 0b00100000 (Audio output 3 in overload protection state)
```

6.1.4 Set Standby Mode - Command 5(0x05)

- This command will enable or disable standby mode
- When ENABLED, the system will power off both the amplifier and CPU after 15 minutes of no UART activity and no audio activity
- When DISABLED, the unit will only power off the amplifier after 15 minutes of no UART activity and no audio activity

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	5(0x05)
Data	**1**	**1(0x01) = Enable standby mode** **0(0x00) = Disable standby mode**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	5(0x05)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 5 1

req msg: length 0007 bytes 255 085 000 002 005 001 248
            cmd 0005 payload 001

001(0x01) - Enable standby mode

rsp msg: length 0007 bytes 255 085 000 002 005 000 249
            cmd 0005 result 0000
```

6.1.5 Get Standby Mode - Command 6(0x06)

- This command queries the standby mode state

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	1(0x01)
Command	1	6(0x06)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Command	1	6(0x06)
Result	1	1(0x01)
Data 1	**1**	**1(0x01) = Standby mode enabled** **0(0x00) = Standby mode disabled**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 6

req msg: length 0006 bytes 255 085 000 001 006 249
            cmd 0006

rsp msg: length 0008 bytes 255 085 000 003 006 001 001 245
            cmd 0006 result 0001 payload 001
```

001(0x01) - Standby mode ENABLED

6.1.6 Reset Default Settings - Command 7(0x07)

- This command will reset values to factory defaults
- **NOTE: Once you have reset factory defaults you will have to enable remote management mode to continue sending management messages to the device**

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	1(0x01)
Command	1	7(0x07)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	7(0x07)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

6.2 Audio Control Group

6.2.1 Set Current Volume - Command 32(0x20)

- This command is used to set the volume for a given zone
- A volume of 100 indicates maximum volume (+26 dB) and each step down is 0.5 dB (99 = +25.5 dB, 98 = +25 dB, etc.). The minimum volume is -24 dB.
- If the volume value is set to 0, the output is muted

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Command	1	32(0x20)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Volume:** **0(0x00) to 100(0x64)** **0(0x00) = mute**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	32(0x20)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 32 3 45

req msg: length 0008 bytes 255 085 000 003 032 003 045 173
            cmd 0032 payload 003 045

003(0x03) - Zone Output 3
045(0x2D) - Volume 45

rsp msg: length 0007 bytes 255 085 000 002 032 000 222
            cmd 0032 result 0000
```

6.2.2 Get Current Volume - Command 33(0x21)

- This command queries a zone's current volume
- A volume of 100 indicates maximum volume (+26 dB) and each step down is 0.5 dB (99 = +25.5 dB, 98 = +25 dB, etc.). The minimum volume is -24 dB.
- If the volume value is set to 0, the output is muted

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	33(0x21)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	4(0x04)
Command	1	33(0x21)
Result	1	1(0x01)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Volume value:** **0(0x00) to 100(0x64)** **0(0x00) = mute**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 33 1

req msg: length 0007 bytes 255 085 000 002 033 001 220
            cmd 0033 payload 001

001(0x01) - Zone Output 1

rsp msg: length 0009 bytes 255 085 000 004 033 001 001 035 182
            cmd 0033 result 0001 payload 001 035

001(0x01) - Zone Output 1
035(0x23) - Volume 35
```

6.2.3 Set Tone Control - Command 34(0x22)

- This command is used to set the current tone control settings
- Treble and bass values are represented as a **signed byte**
- The valid range for treble and bass is -12 to +12 dB in 1 dB steps
- Treble adjusts audio at 5 kHz and bass adjusts audio at 100 Hz
- Loudness is an enabled/disabled setting

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	5(0x05)
Command	1	34(0x22)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Treble (dB):** **-12(0xF4) to +12(0x0C)**
Data 3	**1**	**Bass (dB):** **-12(0xF4) to +12(0x0C)**
Data 4	**1**	**Loudness:** **1(0x01) = enable** **0(0x00) = disable**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	34(0x22)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 34 2 -5 3 1

req msg: length 0010 bytes 255 085 000 005 034 002 251 003 001 216
            cmd 0034 payload 002 251 003 001

002(0x02) - Zone Output 2
251(0xFB) - Treble -5 (As a signed value 0xFB = -5)
003(0x03) - Bass 3
001(0x01) - Enable loudness

rsp msg: length 0007 bytes 255 085 000 002 034 000 220
            cmd 0034 result 0000
```

6.2.4 Get Tone Control - Command 35(0x23)

- This command is used to query the current tone control settings
- Treble and bass values are represented as a **signed byte**
- The valid range for treble and bass is -12 to +12 dB in 1 dB steps
- Treble adjusts audio at 5 kHz and bass adjusts audio at 100 Hz
- Loudness is an enabled/disabled setting

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	35(0x23)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	6(0x06)
Command	1	35(0x23)
Result	1	1(0x01)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Treble adjustment value in dB as a signed byte:** **-12(0xF4) to +12(0x0C)**
Data 3	**1**	**Bass adjustment value in dB as a signed byte:** **-12(0xF4) to +12(0x0C)**
Data 4	**1**	**Loudness setting:** **1(0x01) = enabled** **0(0x00) = disabled**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 35 3

req msg: length 0007 bytes 255 085 000 002 035 003 216
            cmd 0035 payload 003

003(0x03) - Zone Output 3

rsp msg: length 0011 bytes 255 085 000 006 035 001 003 012 012 000 187
            cmd 0035 result 0001 payload 003 012 012 000

003(0x03) - Zone Output 3
012(0x0C) - Treble 12
012(0x0C) - Bass 12
000(0x00) - Loudness disabled
```

6.2.5 Set Do Not Disturb - Command 36(0x24)

- This command is used to enable/disable Do Not Disturb mode for a particular output
- If Do Not Disturb is enabled on a zone, that zone will ignore paging and whole house music settings

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Command	1	36(0x24)
Data 1	**1**	**Zone Output :** **1(0x01) to 6(0x06)**
Data 2	**1**	**Do Not Disturb setting:** **1(0x01) = enable** **0(0x00) = disable**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	36(0x24)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 36 5 1

req msg: length 0008 bytes 255 085 000 003 036 005 001 211
            cmd 0036 payload 005 001

005(0x05) - Zone Output 5
001(0x01) - Enable Do Not Disturb

rsp msg: length 0007 bytes 255 085 000 002 036 000 218
            cmd 0036 result 0000
```

6.2.6 Get Do Not Disturb - Command 37(0x25)

- This command queries the Do Not Disturb mode setting for a particular zone
- If Do Not Disturb is enabled on a zone, that zone will ignore paging and whole house music settings

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	37(0x25)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	4(0x04)
Command	1	37(0x25)
Result	1	1(0x01)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Do Not Disturb setting:** **1(0x01) = enabled** **0(0x00) = disabled**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 37 6

req msg: length 0007 bytes 255 085 000 002 037 006 211
            cmd 0037 payload 006

006(0x06) - Zone Output 6

rsp msg: length 0009 bytes 255 085 000 004 037 001 006 000 208
            cmd 0037 result 0001 payload 006 000

006(0x06) - Zone Output 6
000(0x00) - Do Not Disturb is disabled
```

6.2.7 Set Routing Map - Command 38(0x26)

- This command configures audio input to output routing
- In order to send one input to multiple outputs, multiple instances of this command will need to be issued. For example, if we wanted to route audio input 2 to zone outputs 3 and 5, the two data bytes for the two commands should be 0x02 0x03 followed by 0x02 0x05 (data fields for the two commands)
- If the audio input is set to 0, then the zone should be powered off (no input)
- **ALERT: This command requires 200 milliseconds to complete. Another request should not be initiated until 200 milliseconds have elapsed.**

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Command	1	38(0x26)
Data 1	**1**	**Audio Input:** **1(0x01) - 6(0x06)** **0(0x00) = Off**
Data 2	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	38(0x26)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 38 1 5

req msg: length 0008 bytes 255 085 000 003 038 001 005 210
            cmd 0038 payload 001 005

001(0x01) - Audio Input 1
005(0x05) - Zone Output 5

rsp msg: length 0007 bytes 255 085 000 002 038 000 216
            cmd 0038 result 0000
```

6.2.8 Get Routing Map - Command 39(0x27)

- This command queries which input a particular audio output is currently receiving audio from
- If an audio output is receiving audio from input 4, the return value should be 4(0x04)
- A return value of 0 indicates that the zone is powered off (no input)

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	39(0x27)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	4(0x04)
Command	1	39(0x27)
Result	1	1(0x01)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Audio Input:** **1(0x01) to 6(0x06)** **0(0x00) = Off**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 39 3

req msg: length 0007 bytes 255 085 000 002 039 003 212
            cmd 0039 payload 003

003(0x03) - Zone Output 3

rsp msg: length 0009 bytes 255 085 000 004 039 001 003 003 206
            cmd 0039 result 0001 payload 003 003

003(0x03) - Zone Output 3
003(0x03) - Audio Input 3
```

6.3 Audio Setup Group

6.3.1 Set Default Volume - Command 48(0x30)

- This command is used to set the default volume for a zone
- A default volume of 100 indicates maximum volume (+26 dB) and each step down is 0.5 dB (99 = +25.5 dB, 98 = +25 dB, etc.). The minimum default volume is -24 dB.
- If the default volume value is set to 0, the output is muted by default
- The default volume is the volume the zone output is set to when it is powered on

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Command	1	48(0x30)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Volume:** **0(0x00) to 100(0x64)** **0(0x00) = mute**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	48(0x30)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 48 5 45

req msg: length 0008 bytes 255 085 000 003 048 005 045 155
            cmd 0048 payload 005 045

005(0x05) - Zone Output 5
045(0x2D) - Volume 45

rsp msg: length 0007 bytes 255 085 000 002 048 000 206
            cmd 0048 result 0000
```

6.3.2 Get Default Volume - Command 49(0x31)

- This command queries the default volume
- A default volume of 100 indicates maximum volume (+26 dB) and each step down is 0.5 dB (99 = +25.5 dB, 98 = +25 dB, etc.). The minimum default volume is -24 dB.
- If the default volume value is set to 0, the output is muted by default
- The default volume is the volume the zone output is set to when it is powered on

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	49(0x31)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	4(0x04)
Command	1	49(0x31)
Result	1	1(0x01)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Default volume:** **0(0x00) to 100(0x64)** **0(0x00) = mute**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 49 1

req msg: length 0007 bytes 255 085 000 002 049 001 204
            cmd 0049 payload 001

001(0x01) = Zone Output 1

rsp msg: length 0009 bytes 255 085 000 004 049 001 001 035 166
            cmd 0049 result 0001 payload 001 035

001(0x01) - Zone Output 1
035(0x23) - Default Volume 35
```

6.3.3 Set Maximum Volume - Command 50(0x32)

- This command is used to set the maximum volume
- A maximum volume of 100 indicates maximum volume (+26 dB) and each step down is 0.5 dB (99 = +25.5 dB, 98 = +25 dB, etc.). The minimum maximum volume is -24 dB.
- If the maximum volume value is set to 0, the output is always muted
- The current volume can never exceed the maximum volume (if the volume is set to a value above the maximum volume, it changes to the maximum volume)

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Volume:** **0(0x00) to 100(0x64)** **0(0x00) = mute**
Command	1	50(0x32)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	50(0x32)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 50 3 32

req msg: length 0008 bytes 255 085 000 003 050 003 032 168
            cmd 0050 payload 003 032

003(0x03) - Zone Output 3
032(0x20) - Volume 32

rsp msg: length 0007 bytes 255 085 000 002 050 000 204
            cmd 0050 result 0000
```

6.3.4 Get Maximum Volume - Command 51(0x33)

- This command queries the maximum volume
- A maximum volume of 100 indicates maximum volume (+26 dB) and each step down is 0.5 dB (99 = +25.5 dB, 98 = +25 dB, etc.). The minimum maximum volume is -24 dB.
- If the maximum volume value is set to 0, the output is always muted
- The current volume can never exceed the maximum volume (if the volume is set to a value above the maximum volume, it changes to the maximum volume)

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	51(0x33)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	4(0x04)
Command	1	51(0x33)
Result	1	1(0x01)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Maximum volume:** **0(0x00) to 100(0x64)** **0(0x00) = mute**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 51 4

req msg: length 0007 bytes 255 085 000 002 051 004 199
            cmd 0051 payload 004

004(0x04) - Zone Output 4

rsp msg: length 0009 bytes 255 085 000 004 051 001 004 100 096
            cmd 0051 result 0001 payload 004 100

004(0x04) - Zone Output 4
100(0x64) - Maximum Volume 100
```

6.3.5 Set Default Tone Control - Command 52(0x34)

- This command is used to set the default tone control settings
- Treble and bass values are set as a signed byte
- The valid range for treble and bass is -12 to +12 dB in 1 dB steps
- Treble adjusts audio at 5 kHz and bass adjusts audio at 100 Hz
- Loudness is an on/off setting
- For default tone setting, if it is set to 0 (use default), the zone will always power on with these tone settings. If it is set to 1 (use last setting), the zone will power on with the previously used tone settings (last set with command 0x22 Set Tone Control)

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	6(0x06)
Command	1	52(0x34)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Treble (dB):** **-12(0xF4) to +12(0x0C)**
Data 3	**1**	**Bass (dB):** **-12(0xF4) to +12(0x0C)**
Data 4	**1**	**Loudness:** **1(0x01) = enable** **0(0x00) = disable**
Data 5	**1**	**Default Tone Setting:** **0(0x00) = use default** **1(0x01) = use last setting**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	52(0x34)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 52 5 -12 4 1 0

req msg: length 0011 bytes 255 085 000 006 052 005 244 004 001 000 200
            cmd 0052 payload 005 244 004 001 000

005(0x05) - Zone Output 5
244(0xF4) – Treble -12 (As a signed value 0xF4 = -5)
004(0x04) – Bass 4
001(0x01) – Enable loudness
000(0x00) – Use default tone setting

rsp msg: length 0007 bytes 255 085 000 002 052 000 202
            cmd 0052 result 0000
```

6.3.6 Get Default Tone Control - Command 53(0x35)

- This command is used to retrieve the default tone control settings
- Treble and bass values are represented as signed byte values
- The valid range for treble and bass is -12 to +12 dB in 1 dB steps
- Treble adjusts audio at 5 kHz and bass adjusts audio at 100 Hz
- Loudness is an on/off setting
- For default tone setting, if it is set to 0 (use default), the zone will always power on with these tone settings. If it is set to 1 (use last setting), the zone will power on with the previously used tone settings (last set with command 0x22 Set Tone Control)

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	53(0x35)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	7(0x07)
Command	1	53(0x35)
Result	1	1(0x01)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Treble (dB):** **-12(0xF4) to +12(0x0C)**
Data 3	**1**	**Bass (dB):** **-12(0xF4) to +12(0x0C)**
Data 4	**1**	**Loudness setting:** **1(0x01) = enabled** **0(0x00) = disabled**
Data 5	**1**	**Default tone setting:** **0(0x00) = use default** **1(0x01) = use last setting**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 53 6

req msg: length 0007 bytes 255 085 000 002 053 006 195
            cmd 0053 payload 006

006(0x06) - Zone Output 6

rsp msg: length 0012 bytes 255 085 000 007 053 001 006 012 012 000 000 165
            cmd 0053 result 0001 payload 006 012 012 000 000

006(0x06) - Zone Output 6
012(0x0C) – Treble 12
012(0x0C) – Bass 12
000(0x00) – Loudness disabled
000(0x00) – Use default tone setting
```

6.3.7 Set Input Level - Command 54(0x36)

- This command is used to set the input level of each audio input

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Command	1	54(0x36)
Data 1	**1**	**Audio Input:** **1(0x01) to 6(0x06)** **9(0x09) = Page Input**
Data 2	**1**	**Input Level Gain (0 = +6 dB, 1 = +3 dB, 2 = +0 dB, 3 = -3 dB, 4 = -6 dB)**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	54(0x36)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 54 1 2

req msg: length 0008 bytes 255 085 000 003 054 001 002 196
            cmd 0054 payload 001 002

001(0x01) – Audio Input 1
002(0x02) – Input Level Gain 0 dB

rsp msg: length 0007 bytes 255 085 000 002 054 000 200
            cmd 0054 result 0000
```

6.3.8 Get Input Level - Command 55(0x37)

- This command queries the input level of each audio input.

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	55(0x37)
Data 1	**1**	**Audio Input:** **1(0x01) to 6(0x06)** **9(0x09) = Page Input**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	4(0x04)
Command	1	55(0x37)
Result	1	1(0x01)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Input Level Gain:** **0 = +6 dB, 1 = +3 dB, 2 = +0 dB, 3 = -3 dB, 4 = -6 dB**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 55 6

req msg: length 0007 bytes 255 085 000 002 055 006 193
            cmd 0055 payload 006

006(0x06) – Audio Input 6

rsp msg: length 0009 bytes 255 085 000 004 055 001 006 002 188
            cmd 0055 result 0001 payload 006 002

006(0x06) – Zone Output 6
002(0x02) – Input Level Gain 0 dB
```

6.3.9 Set Zone Preamp Output Mode - Command 56(0x38)

- This command is used to set the zone preamp mode for each audio output
- In variable mode, the current volume and tone settings are applied to the preamp output
- In fixed mode, the volume is fixed to line level audio
- **NOTE: The preamp and speaker outputs are active simultaneously**

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Command	1	56(0x38)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Output Mode:** **0(0x00) = Variable** **1(0x01) = Fixed**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	56(0x38)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 56 4 0

req msg: length 0008 bytes 255 085 000 003 056 004 000 193
            cmd 0056 payload 004 000

004(0x04) – Zone Output 4
000(0x00) – Variable output mode

rsp msg: length 0007 bytes 255 085 000 002 056 000 198
            cmd 0056 result 0000
```

6.3.10 Get Zone Preamp Output Mode - Command 57(0x39)

- This command is used to retrieve the zone preamp mode for each audio output
- For variable output, the current volume and tone settings are applied to the preamp output
- For fixed output, the volume is fixed to line level audio

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	57(0x39)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	4(0x04)
Command	1	57(0x39)
Result	1	1(0x01)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**0(0x00) = Variable** **1(0x01) = Fixed**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 57 1

req msg: length 0007 bytes 255 085 000 002 057 001 196
            cmd 0057 payload 001

001(0x01) – Zone Output 1

rsp msg: length 0009 bytes 255 085 000 004 057 001 001 000 193
            cmd 0057 result 0001 payload 001 000

001(0x01) – Zone Output 1
000(0x00) – Variable output mode
```

6.3.11 Set Startup Mode - Command 58(0x3A)

- This command sets the state of test mode
- When the unit starts up with test mode **enabled**, the default routing map is as follows: Input 1 -> Zone 1, Input 2 -> Zone 2, …, Input 6 -> Zone 6
- When the unit starts up with test mode **disabled**, all zones **will** be powered off (no audio input routing set)

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	58(0x3A)
Data 1	**1**	**1(0x01) = Enable test mode** **0(0x00) = Disable test mode**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	58(0x3A)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 58 1

req msg: length 0007 bytes 255 085 000 002 058 001 195
            cmd 0058 payload 001

001(0x01) – Enable test mode

rsp msg: length 0007 bytes 255 085 000 002 058 000 196
            cmd 0058 result 0000
```

6.3.12 Get Startup Mode - Command 59(0x3B)

- This command queries the state of test mode on power on (i.e. from rear panel power switch or after mains power is restored)
- If test mode is **enabled**, the default routing map is as follows: Input 1 -> Zone 1, Input 2 -> Zone 2, …, Input 6 -> Zone 6
- When the unit starts up with test mode **disabled**, all zones **must** be powered off (no audio input routing set)

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	1(0x01)
Command	1	59(0x3B)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	59(0x3B)
Result	1	1(0x01)
Data 1	**1**	**1(0x01) = Test mode enabled** **0(0x00) = Test mode disabled**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 59

req msg: length 0006 bytes 255 085 000 001 059 196
            cmd 0059

rsp msg: length 0008 bytes 255 085 000 003 059 001 001 192
            cmd 0059 result 0001 payload 001
```

001(0x01) – Test mode enabled

6.4 Paging Group

6.4.1 Set Paging Zones - Command 64(0x40)

- This command configures paging input to output routing

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	64(0x40)
Data	**1**	**Zone Output (1-6) Bitmap** **Bit 7: Zone output 1 setting** **Bit 6: Zone output 2 setting** **Bit 5: Zone output 3 setting** **Bit 4: Zone output 4 setting** **Bit 3: Zone output 5 setting** **Bit 2: Zone output 6 setting** **Bit 1: Not used** **Bit 0: Not used**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	64(0x40)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 64 192

req msg: length 0007 bytes 255 085 000 002 064 192 254
            cmd 0064 payload 192

192(0xC0) – 0b11000000 Route paging input to zone outputs 1 & 2

rsp msg: length 0007 bytes 255 085 000 002 064 000 190
            cmd 0064 result 0000
```

6.4.2 Get Paging Zones - Command 65(0x41)

- This command queries which outputs the paging input is routing audio
- **NOTE: Paging occurs when audio input is detected on paging input**

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	1(0x01)
Command	1	65(0x41)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Command	1	65(0x41)
Result	1	1(0x01)
Data 1	**1**	**Zone Output (1-6) Bitmap** **Bit 7: Zone output 1 setting** **Bit 6: Zone output 2 setting** **Bit 5: Zone output 3 setting** **Bit 4: Zone output 4 setting** **Bit 3: Zone output 5 setting** **Bit 2: Zone output 6 setting** **Bit 1: Not used** **Bit 0: Not used**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 65

req msg: length 0006 bytes 255 085 000 001 065 190
            cmd 0065

rsp msg: length 0008 bytes 255 085 000 003 065 001 192 251
            cmd 0065 result 0001 payload 192
```

192(0xC0) – 0b11000000 Paging input routing to zone outputs 1 & 2

6.4.3 Set Paging Volume - Command 66(0x42)

- This command is used to set the paging volume for each zone
- A paging volume of 100 indicates maximum volume (+26 dB) and each step down is 0.5 dB (99 = +25.5 dB, 98 = +25 dB, etc.). The minimum paging volume is -24 dB.
- If the paging volume value is set to 0, paging to this zone is disabled
- Paging volume overrides the default and current volumes during a page
- **NOTE: Paging occurs when audio input is detected on paging input**

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Command	1	66(0x42)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Volume:** **0(0x00) to 100(0x64)** **0(0x00) = disable**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	66(0x42)
Result	1	1(0x01)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 66 1 23

req msg: length 0008 bytes 255 085 000 003 066 001 023 163
            cmd 0066 payload 001 023

001(0x01) – Zone Output 1
023(0x17) – Volume

rsp msg: length 0007 bytes 255 085 000 002 066 000 188
            cmd 0066 result 0000
```

6.4.4 Get Paging Volume - Command 67(0x43)

- This command queries the paging volume
- A paging volume of 100 indicates maximum volume (+26 dB) and each step down is 0.5 dB (99 = +25.5 dB, 98 = +25 dB, etc.). The minimum paging volume is -24 dB.
- If the paging volume value is set to 0, paging to this zone is disabled
- Paging volume overrides the default and current volumes during a page
- **NOTE: Paging occurs when audio input is detected on paging input**

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Command	1	67(0x43)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	4(0x04)
Command	1	67(0x43)
Result	1	1(0x01)
Data 1	**1**	**Zone Output:** **1(0x01) to 6(0x06)**
Data 2	**1**	**Paging volume value:** **0(0x00) to 100(0x64)** **0(0x00) = disabled**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 67 1

req msg: length 0007 bytes 255 085 000 002 067 001 186
            cmd 0067 payload 001

001(0x01) – Zone Output 1

rsp msg: length 0009 bytes 255 085 000 004 067 001 001 023 160
            cmd 0067 result 0001 payload 001 023

001(0x01) – Zone Output 1
023(0x17) – Volume
```

6.5 Whole House Music Group

6.5.1 Set Whole House Music Zones - Command 74(0x4A)

- This command configures which outputs to use for whole house music mode

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	74(0x4A)
Data	**1**	**Zone Output (1-6) Bitmap** **Bit 7: Zone output 1 setting** **Bit 6: Zone output 2 setting** **Bit 5: Zone output 3 setting** **Bit 4: Zone output 4 setting** **Bit 3: Zone output 5 setting** **Bit 2: Zone output 6 setting** **Bit 1: Not used** **Bit 0: Not used**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	74(0x4A)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 74 168

req msg: length 0007 bytes 255 085 000 002 074 168 012
            cmd 0074 payload 168

168(0xA8) – 0b10101000 Whole house mode will use zone outputs 1, 3, and 5

rsp msg: length 0007 bytes 255 085 000 002 074 000 180
            cmd 0074 result 0000
```

6.5.2 Get Whole House Music Zones - Command 75(0x4B)

- This command is used to retrieve which outputs are used for whole house music mode

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	1(0x01)
Command	1	75(0x4B)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	75(0x4B)
Result	1	1(0x01)
Data	**1**	**Zone Output (1-6) Bitmap** **Bit 7: Zone output 1 setting** **Bit 6: Zone output 2 setting** **Bit 5: Zone output 3 setting** **Bit 4: Zone output 4 setting** **Bit 3: Zone output 5 setting** **Bit 2: Zone output 6 setting** **Bit 1: Not used** **Bit 0: Not used**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 75

req msg: length 0006 bytes 255 085 000 001 075 180
            cmd 0075

rsp msg: length 0008 bytes 255 085 000 003 075 001 168 009
            cmd 0075 result 0001 payload 168
```

168(0xA8) – 0b10101000 Whole house mode will use zone outputs 1, 3, and 5

6.5.3 Start Whole House Music - Command 76(0x4C)

- This command is used start whole house music mode
- The unit must direct audio from the specified audio input to the audio outputs set with 0x4A Set Whole House Music Zones
- This command may also be used to switch audio inputs during Whole House Music mode. Zones in whole house music mode will be locked from input channel changes (except through this command).
- If this command is used to set the audio input to 0, the system still has whole house mode enabled but no active audio input
- **ALERT: The operation will require a maximum of 1200 milliseconds to complete. Each zone specified via the 0x4A Set Whole House Music Zones command will require 200 milliseconds to configure**
- **ALERT: Do not send another request message until your application has waited the appropriate number of milliseconds for the operation to complete**

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	76(0x4C)
Data	**1**	**Audio Input:** **1(0x01) to 6(0x06)** **0(0x00) = Off**
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	76(0x4C)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 76 1

req msg: length 0007 bytes 255 085 000 002 076 001 177
            cmd 0076 payload 001

001(0x01) – Audio Input 1

rsp msg: length 0007 bytes 255 085 000 002 076 000 178
            cmd 0076 result 0000
```

6.5.4 Stop Whole House Music - Command 77(0x4D)

- This command is used stop whole house music mode
- All whole house music zones will continue to play audio from the same input but will accept commands to change inputs

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	1(0x01)
Command	1	77(0x4D)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 0(0x00))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	2(0x02)
Command	1	77(0x4D)
Result	1	0(0x00)
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

6.5.5 Get Whole House Music State - Command 78(0x4E)

- This command is used to retrieve whether or not the system is in Whole House Music mode

Request

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	1(0x01)
Command	1	78(0x4E)
Checksum	1	The checksum is calculated by taking the sum of length, command, and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Response (If Result == 1(0x01))

Field	Size	Description
Sync 1	1	255(0xFF)
Sync 2	1	85(0x55)
Length Hi	1	0(0x00)
Length Lo	1	3(0x03)
Command	1	78(0x4E)
Result	1	1(0x01)
Data	**1**	**0(0x00) = Whole House Music mode is stopped** **1(0x01) = Whole House Music mode is started**
Checksum	1	The checksum is calculated by taking the sum of the length and data bytes and then taking the least significant byte and subtracting that byte from 256(0x100)

Example

```
$ ./remMgmt 192.168.0.198 78

req msg: length 0006 bytes 255 085 000 001 078 177
            cmd 0078

rsp msg: length 0008 bytes 255 085 000 003 078 001 001 173
            cmd 0078 result 0001 payload 001
```

001(0x01) – Whole house mode is started

7 Example Test Code - remMgmt.c

```
/*
-------------------------------------------------------------------------------

    remMgmt.c

    Copyright 2014 CoreBrands LLC

-------------------------------------------------------------------------------
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/unistd.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <netdb.h>

#ifndef TRUE
#define TRUE    1
#define FALSE   0
#endif

#define REQMSG_SIZE             64
#define RSPMSG_SIZE             1024
#define RETRIES                 10

#define SET_MODE_REQ            8
#define SET_MODE_RSP            9

#define REMOTE_MGMT_MODE_OFF    0xAA11CCDD
#define REMOTE_MGMT_MODE_ON     0xBB00EEFF

#define REMOTE_MGMT_TCP_PORT    10200
#define REMOTE_MGMT_UDP_PORT    444

/*=============================================================================*/

unsigned char * buildMgmtMsg(int *                  iMsgLen,        // OUTPUT
                             unsigned char          ucCmd,          // INPUT
                             unsigned short         ushPayloadLen,  // INPUT
                             unsigned char *        pPayload)       // INPUT

/*=============================================================================*/
{
    unsigned char   *pMsg = 0;
    unsigned short  ushCheckSum = 0;
    int iCount = 0;

    if(iMsgLen == 0)
        {
        return 0;
        }

    *iMsgLen = ushPayloadLen + 6;

    pMsg = (unsigned char *)malloc(*iMsgLen);
```

```
    if((pPayload == 0) && (ushPayloadLen))
        {
        *iMsgLen = 0;
        return 0;
        }

    memset(pMsg, 0x00, *iMsgLen);

    pMsg[0] = 0xFF; // First  sync byte
    pMsg[1] = 0x55; // Second sync byte
    pMsg[2] = (ushPayloadLen + 1) >> 8;
    pMsg[3] = (ushPayloadLen + 1) & 0x00FF;
    pMsg[4] = ucCmd;

    ushCheckSum = (unsigned short)pMsg[2] + (unsigned short)pMsg[3] +(unsigned short)pMsg[4];

    for(iCount = 0; iCount < ushPayloadLen; iCount++)
    {
        pMsg[iCount + 5] = pPayload[iCount];
        ushCheckSum += (unsigned short)pMsg[iCount + 5];
    }

    pMsg[*iMsgLen - 1] = (unsigned char)(0x100 - (ushCheckSum & 0x00FF));

    return(pMsg);
}

/*===========================================================================*/

int sendMgmtReqRecvMgmtRsp( char * pServerName,         // INPUT
                            unsigned char * pMyBuffer,  // INPUT
                            int iLen)                   // INPUT

/*===========================================================================*/
{
    int iCount = 0;
    int iSocketFD = 0;
    int iByteCount = 0;
    int iDisplayLen = 0;
    struct hostent *pServer;
    struct sockaddr_in sockAddr;
    unsigned char ucResponseMessage[RSPMSG_SIZE];

    if((pMyBuffer == 0) || ((pMyBuffer != 0) && (iLen < 1)))
        {
        printf("ERROR, null message pointer or illegal message length\n");
        return(1);
        }

    bzero(ucResponseMessage, RSPMSG_SIZE);

    iSocketFD = socket(AF_INET, SOCK_STREAM, 0);
    if(iSocketFD < 0)
        {
        printf("ERROR, opening TCP socket\n");
        return(1);
        }

    pServer = gethostbyname(pServerName);
    if(pServer == 0)
        {
```

```
    printf("ERROR, no such host %s\n", pServerName);
    return(1);
    }

bzero((char *) &sockAddr, sizeof(sockAddr));
sockAddr.sin_family = AF_INET;
bcopy((char *)pServer->h_addr, (char *)&sockAddr.sin_addr.s_addr, pServer->h_length);
sockAddr.sin_port = htons(REMOTE_MGMT_TCP_PORT);

if(connect(iSocketFD, (struct sockaddr *)&sockAddr, sizeof(sockAddr)) < 0)
    {
    printf("ERROR, connecting to TCP socket\n");
    return(1);
    }

iByteCount = send(iSocketFD, (const void *)pMyBuffer, (size_t)iLen, 0);

if(iByteCount < 0)
    {
    printf("ERROR, sending request to TCP socket\n");
    return(1);
    }

printf("\n");
printf("req msg: length %04d bytes ", iByteCount);
for(iCount = 0; iCount < iByteCount; iCount++)
    printf("%03d ", pMyBuffer[iCount]);
printf("\n");

iDisplayLen = (pMyBuffer[2] << 8);
iDisplayLen += pMyBuffer[3] - 1;

if(iDisplayLen == 0)
    {
    printf("            cmd %04d\n", pMyBuffer[4]);
    }
else
    {
    printf("            cmd %04d payload ", pMyBuffer[4]);
    for(iCount = 0; iCount < iDisplayLen; iCount++)
        printf("%03d ", pMyBuffer[iCount + 5]);
    printf("\n");
    }
printf("\n");

free(pMyBuffer);

iByteCount = recv(iSocketFD, ucResponseMessage, RSPMSG_SIZE, 0);
if(iByteCount < 0)
    {
    printf("ERROR, receiving response from TCP socket\n");
    return(1);
    }

printf("rsp msg: length %04d bytes ", iByteCount);
for(iCount = 0; iCount < iByteCount; iCount++)
    printf("%03d ", ucResponseMessage[iCount]);
printf("\n");

iDisplayLen = (ucResponseMessage[2] << 8);
iDisplayLen = iDisplayLen + ucResponseMessage[3] - 2;
```

```
    if((ucResponseMessage[4] >= 251) && (ucResponseMessage[4] <= 255))
        {
        printf("            result %03d\n", ucResponseMessage[4]);
        }
    else
        {
        if(iDisplayLen == 0)
            {
            printf("            cmd %04d result %04d\n",
                                ucResponseMessage[4], ucResponseMessage[5]);
            }
        else
            {
            printf("            cmd %04d result %04d payload ",
                                ucResponseMessage[4], ucResponseMessage[5]);
            for(iCount = 0; iCount < iDisplayLen; iCount++)
                printf("%03d ", ucResponseMessage[iCount + 6]);
            printf("\n");
            }
        }
    printf("\n");

    close(iSocketFD);

    return(0);
}

/*=========================================================================*/

int cfgRemoteMgmtMode(  char *  pServerName,        // INPUT
                        int     iRemoteMgmtModeCfg) // INPUT

/*=========================================================================*/
{
        int iSocketFD = 0;
        int iByteCount = 0;
        int iRetryCount = RETRIES;
        int iRequestCode = SET_MODE_REQ;
        int iResponseCode = 0;
        int iRemoteMgmtMode = 0;
        struct hostent *pServer;
        struct sockaddr_in sockAddr;
        unsigned char ucRequestMessage[REQMSG_SIZE];
        unsigned char ucResponseMessage[RSPMSG_SIZE];

        memset(ucRequestMessage, 0, REQMSG_SIZE);
        memcpy(ucRequestMessage + 0, &iRequestCode, 4);     // Big Endian
        memcpy(ucRequestMessage + 4, &iRemoteMgmtModeCfg, 4); // Big Endian

        iSocketFD = socket(AF_INET, SOCK_DGRAM, 0);

        pServer = gethostbyname(pServerName);
        if(pServer == 0)
            {
            printf("ERROR, no such host %s\n", pServerName);
            return(1);
            }

        sockAddr.sin_family = AF_INET;
        sockAddr.sin_port = htons(REMOTE_MGMT_UDP_PORT);
        bcopy((char *)pServer->h_addr, (char *)&sockAddr.sin_addr.s_addr, pServer->h_length);
```

```
        while(iRetryCount > 0)
            {
            if(connect(iSocketFD, (struct sockaddr *)&sockAddr, sizeof(sockAddr)) < 0)
                {
                printf("ERROR, connecting to socket to UDP socket\n");
                return(1);
                }

            iByteCount = (int)sendto(iSocketFD,
                                     (const void *)ucRequestMessage,
                                     (size_t)REQMSG_SIZE,
                                     0,
                                     (const struct sockaddr *)&sockAddr,
                                     sizeof(sockAddr));

            if(iByteCount != REQMSG_SIZE)
                {
                printf("ERROR, sending request to UDP socket\n");
                return(1);
                }

            iByteCount = recv(iSocketFD, (void *)ucResponseMessage, RSPMSG_SIZE, 0);
            if(iByteCount >= 8)
                {
                memcpy(&iResponseCode, ucResponseMessage + 0, 4);   // Big Endian
                memcpy(&iRemoteMgmtMode, ucResponseMessage + 4, 4); // Big Endian

                if((iResponseCode == SET_MODE_RSP) && (iRemoteMgmtModeCfg == iRemoteMgmtMode))
                    {
                    close(iSocketFD);
                    return(0);
                    }
                }

            iRetryCount--;
            }

        close(iSocketFD);

        printf("ERROR, sending request to UDP socket after %d retries\n", RETRIES);
        return(1);
}

/*=========================================================================*/

int main(   int argc,       // INPUT
            char *argv[])   // INPUT

/*=========================================================================*/
{
    int iCount = 0;
    int iParamCount = 0;
    int iBufferLength = 0;
    unsigned char * pMyBuffer = 0;
    unsigned char ucCommand = 0;
    unsigned char ucParams[64];

    if( (argc < 3) ||
        ((argc == 2) && (strcmp("?", argv[1]) == 0)) ||
        ((argc == 3) && (strcmp("?", argv[2]) == 0)))
        {
```

```
        printf("Usage:\n"
            "%s <ipaddr> on  (  Enables remote management mode)\n"
            "%s <ipaddr> off ( Disables remote management mode)\n"
            "%s <ipaddr> <cmd> <data 1> ... <data n> (DECIMAL VALUES ONLY)\n",
            argv[0], argv[0], argv[0]);
        exit(1);
        }

    if((strcmp(argv[2],"on") == 0) && (argc == 3))
        {
        if(cfgRemoteMgmtMode(argv[1], REMOTE_MGMT_MODE_ON)) exit(1);
        printf("Remote management mode ON\n");
        }
    else if((strcmp(argv[2],"off") == 0) && (argc == 3))
        {
        if(cfgRemoteMgmtMode(argv[1], REMOTE_MGMT_MODE_OFF)) exit(1);
        printf("Remote management mode OFF\n");
        }
    else
        {
        iParamCount = argc - 3;

        ucCommand = atoi(argv[2]);

        if((ucCommand ==  1) || // Do not use these command codes
           (ucCommand ==  2) ||
           (ucCommand == 16) ||
           (ucCommand == 17) ||
           (ucCommand == 18) ||
           (ucCommand == 19) ||
           (ucCommand == 20)
           )
            {
            printf("Command %d not supported\n", ucCommand);
            exit(1);
            }

        for(iCount = 0; iCount < iParamCount; iCount++)
            {
            ucParams[iCount] = atoi(argv[3 + iCount]);;
            }

        pMyBuffer = buildMgmtMsg(&iBufferLength, ucCommand, iParamCount, ucParams);

        if(sendMgmtReqRecvMgmtRsp(argv[1], pMyBuffer, iBufferLength)) exit(1);
        }

    exit(0);
}
```