Kali Audio
Santa Monica Project
Network Interface Specification

Revision 1.0

Revision History

Date	Version	Author	Revision
5/2/2020	0.8	Jeff Claar	Initial draft for review.
5/8/2020	0.9	Jeff Claar	Updated message header to KSM. Added support for the loudspeaker sending messages to the controller. Updated speaker controller workflow to indicate shutting down of UDP server. Defined additional controller messages.
5/12/2020	0.91	Jeff Claar	Added the rest of the currently defined messages. (Some are listed as TBD.) Cleaned up some formatting.
5/14/2020	0.93	Jeff Claar	Changed SPKAVAIL and CONTAVAIL to use a comma between IP address and port. Message delimiter is now a semicolon in order to allow spaces in the data. Data length field has been removed. Added list of valid characters for the speaker name.
5/27/2020	0.94	Jeff Claar	Changed SPK_LOCATE to SPKLOCATE for consistency. Added message for speaker EQ settings.
5/27/2020	0.95	Jeff Claar	Added box to open TCP on port 38001 in speaker connect diagram.
6/2/2020	0.96	Jeff Claar	Added definitions for Speaker Preset Select. Added Speaker EQ Enable property.
6/7/2020	0.97	Jeff Claar	Added information about firmware downloads. (Section 6.5) Added messages to initiate speaker messages for testing. (Section 6.3.17.)
6/8/2020	0.97.1	Jeff Claar	Fixed preset selection (Section 6.3.10) so that presets are 1-based, not 0-based.
6/9/2020	0.97.2	Jeff Claar	Added CRC error to error table.
6/15/2020	0.97.3	Jeff Claar	New trim level is returned in the ACK message for Speaker Trim Level Increment/Decrement. (See section 6.3.14.)
7/8/2020	1.0	Jeff Claar	Updated Q range in speaker EQ parameters. (See section 6.3.5.)

Table of Contents

1 Overview and Purpose

The Kali Santa Monica powered studio monitors will be premium studio monitors that have the ability to set room tuning and/or setup configurations via communication with Graphical User Interface software running on a network connected PC/Mac. The GUI will also operate as a monitor controller, allowing the monitoring and control of operational parameters.

This document details the network interface specification for the Santa Monica project. It describes the protocol with which a controller PC will communicate with the loudspeaker, as well as discovery of loudspeakers on the network and error detection and recovery.

One of the main goals of this specification is to be platform-independent, in order to ease implementation on various controller platforms. To that end, values are defined in such a way that should be independent of word size or endian-ness.

2 Terminology

Controller: The controlling system on the network, typically a Windows 10 PC.

Loudspeaker: A Santa Monica loudspeaker.

Network: A wired network, containing both Loudspeakers and one Controller on the same subnet.

3 Message Format

3.1 Transport

Messages will be transported using through TCP/IP, using sockets or another mechanism to implement TCP connections. (For example, the Windows .NET framework uses TcpClient and TcpListener for these connections.) The port for loudspeaker discovery will be 38000. Loudspeakers will receive messages on port 38001, and controllers will receive messages on port 38002. (These ports can change at run-time, and are specified in the messages used to establish connections.)

3.2 Serialization

It is assumed that only a single message will be sent at a time. That is, after sending a message, the controller will wait for a response from the loudspeaker before sending another. Depending on the implementation, the controller may send messages to multiple loudspeakers simultaneously, but only a single message to any particular loudspeaker at a time.

3.3 Response Time

Loudspeakers should respond to a message within one second. If no response is received, it will be assumed that the loudspeaker has become non-responsive, and the controller will display an appropriate error to the user.

3.4 Format

Messages will be sent in plain-text format. While this leads to messages that are more verbose than binary, it has the following advantages:

1. Messages are human-readable. Particularly on embedded systems, which may have limited debugging capabilities, it is much simpler to be able to display a text string to a terminal or through a debugger.
2. Endian problems are avoided for any numeric values. (e.g. The controller is little-endian and the loudspeaker platform is big-endian, or vice-versa.) Parsing a string to a number avoids having to be concerned about this issue.
3. String parsing is simply handled through functions available in standard C, such as strtok, atoi, etc.

As mentioned above, string-based messages will be longer than binary-encoded messages, but a typical message will be on the order of tens of bytes. On a wired network, any additional bandwidth required by such a message will be negligible.

Messages have the format:

Header	Message	Data	Checksum

All message sections will be in upper-case, apart from the Data section, which may contain lower-case characters depending on the message. (e.g. the loudspeaker name may contain lower-case characters). Each field is separated by a delimiter character, a semicolon.

Header: Indicates the start of a message. This is always the same. For this project, the header will always be the string “KSM”.

Message: A string containing the type of message.

Data: A string containing message-specific data, if any.

Checksum: A simple checksum of the message, in decimal. This is simply the sum of all bytes in the message (not including the checksum itself, or the delimiter character before the checksum), modulo 256.[1]

3.5 Numeric Data Format

For messages that require numeric values, the data will be represented as an ASCII integer or floating-point number (defined on a per-message basis).

Integer: 123

Floating point: 1.23

Floating point numbers will always have a leading zero. (For example, 0.12, and not .12.)

This provides simple conversions to numeric values using the C functions atoi and atof.

3.6 Acknowledgement

Upon receipt and successful processing of a message, the loudspeaker will reply with an acknowledgment in the standard message format, containing the message “ACK” and data containing any additional information, depending on the message type. If the message is not handled properly (due to a checksum failure, invalid parameters, etc.) the loudspeaker will respond with the message “NACK” and a numeric error code. Error values are defined in the “Errors” section, below.

The controller will not send another message until it has received a response from the loudspeaker. If the message has not been responded to within one second, it will be assumed that the loudspeaker has been removed from the network or become non-responsive.

3.7 Loudspeaker-Initiated Messages

While most messages from the loudspeaker are prompted by a message from the controller, some can be initiated by the loudspeaker. Messages such as an amp fault, limiter engagement, or speaker status change are initiated from the loudspeaker when those particular conditions occur. The possible messages are detailed in section 6.4.

[1] More robust error checking can also be used (e.g. CRC, all the way up to SHA-based hashes). This seems to be overkill for this case, since the system is on a wired network and should be highly reliable.

These messages are formatted like any other. The main caveat is that the controller needs to be able to handle the possible race condition, where it sends a message to a loudspeaker, but receives an amp fault (for example) message before receiving the ACK.

4 Discovery

Discovery (that is, the finding of new loudspeakers on the network) is handled through standard UDP broadcast messages. The workflow for a loudspeaker upon powerup is shown on Figure 1:

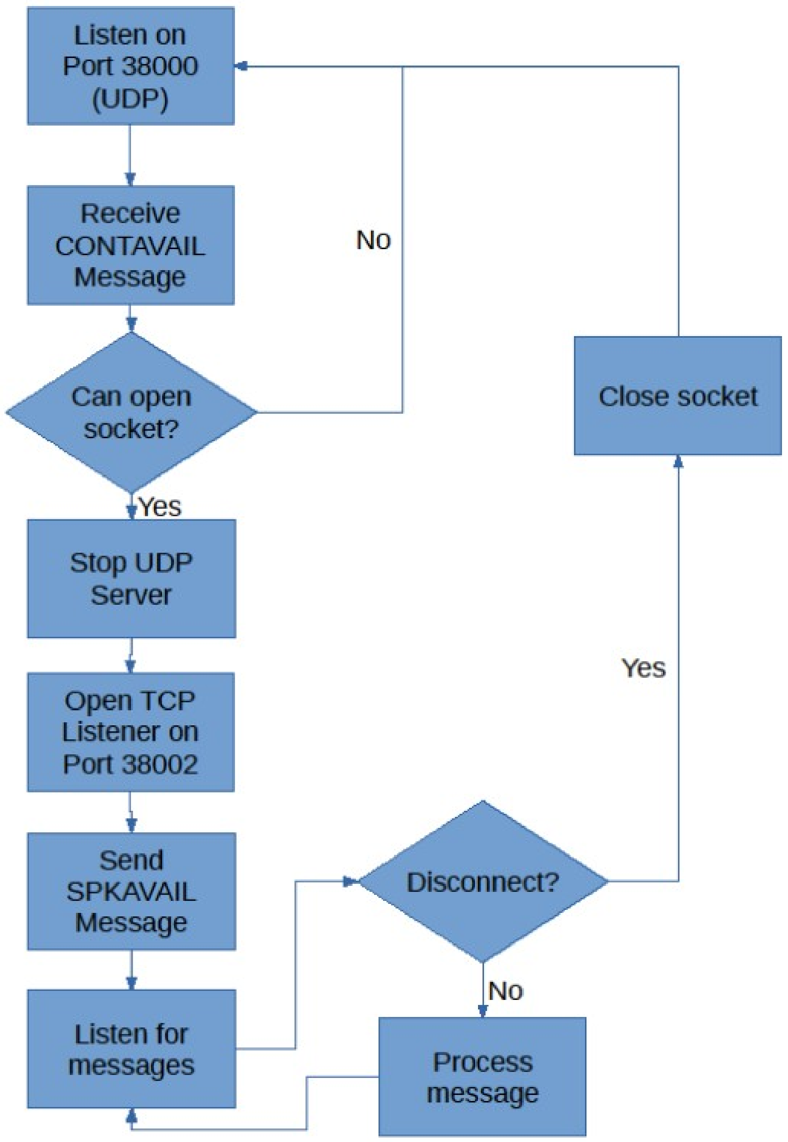

Figure 1 – Loudspeaker Firmware Discovery/Connection Flow

On boot, the loudspeaker creates a UDP server that listens on port 38000. When a controller is launched, it will broadcast a CONTAVAIL message. When a loudspeaker receives this message, it will attempt to establish a connection with the controller, using the address information sent in the CONTAVAIL data.

If the connection is established, the loudspeaker will send a SPKAVAIL message containing its address information and enter its message processing loop.

If the connection is not established, the loudspeaker will return to its listening mode and wait for another CONTAVAIL message. The connection may fail if another loudspeaker has already connected using that port. It is the controller's responsibility to continue to send CONTAVAIL messages on a regular basis to discover new loudspeakers. The general flow for the controller is:

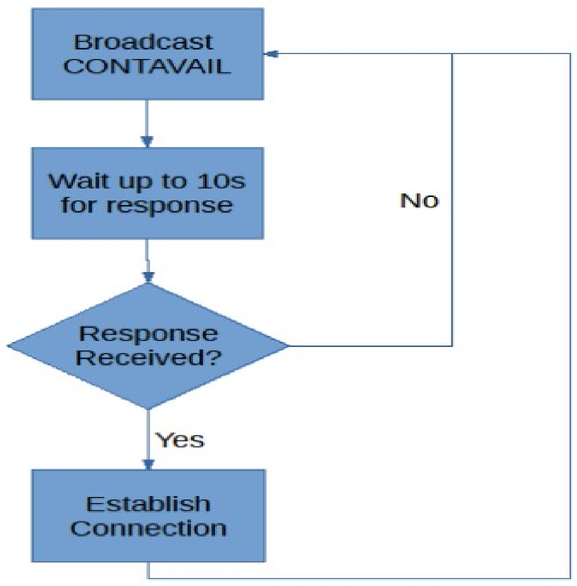

Figure 2 - Controller Broadcast Flow

The wait will be interrupted when a single loudspeaker responds, and the controller will immediately rebroadcast the CONTAVAIL message to connect to the next. This should allow the initial launch of the controller to connect to all available loudspeakers in just a few seconds. Once all connections are established, the 10-second timeout would go into effect, so any new loudspeakers that come online will be detected within that timeframe.

5 “Still Alive” Testing

In order to determine that the loudspeaker/controller connection is still available, once per minute, the controller will send an “ALIVE” message to each connected loudspeaker. The loudspeaker will ACK the message (with no data). Both the loudspeaker and controller can use this response to confirm that the connection is still valid. (i.e. The controller can determine the connection is invalid by not getting a response. The loudspeaker can determine the connection is invalid if it does not receive a message every minute.)

6 Messages

Following are the supported messages that can be sent between a controller and loudspeaker. It is not explicitly called out for brevity, but note that any messages that respond with an ACK can also respond with NACK and an appropriate error code.

6.1 ACK/NACK

6.1.1 ACK

```
KSM;ACK;<data>;<checksum>
```

This is the standard response when a message has been processed successfully. It can include data specific to the message being processed. (e.g. if requesting the loudspeaker name, the data will contain the name.) It can also contain no data.

6.1.2 NACK

```
KSM;NACK;<error code>;<checksum>
```

This is the standard response when a message has been received but has failed to process for some reason. The error code will be one of the numeric codes defined in the Errors section, below.

6.2 Connection-related Messages

The following messages are used to establish and maintain a connection between a controller and a loudspeaker.

6.2.1 Controller Available

```
KSM;CONTAVAIL;<IP address>,<port>;<checksum>
```

Description: This message is broadcast when a controller wishes to establish a connection with a loudspeaker.

Response: None (A loudspeaker that wishes to connect will send its own Speaker Available message.)

6.2.2 Speaker Available

```
KSM;SPKAVAIL;<IP address>,<port>;<checksum>
```

Description: When a speaker receives a CONTAVAIL message, it will open a socket on the specified IP address and port. Upon successful opening, it will send this message to the controller containing the loudspeaker’s IP address and port. The controller will then open a socket using that address, and communication is established.

Response: None

6.2.3 Controller Disconnect

```
KSM;DISCONNECT;<checksum>
```

Description: This message is sent when a controller is shut down or otherwise needs to disconnect from the loudspeaker.

The loudspeaker will send an ACK, then close its socket. It will then recreate the UDP server to begin listening for new connections. (See Figure 1.)

Response: ACK

6.2.4 Controller Alive

`KSM;ALIVE;<checksum>`

Description: This message is sent once per minute by the controller, to all connected loudspeakers. If the loudspeaker has not received the message within one minute, it can assume that the controller has become unresponsive.[2]

Response: KSM;ACK;<checksum>

[2] Technically there should be some leeway in this. Instead of exactly one minute, perhaps 75 seconds should be allowed in case the controller is momentarily late sending its messages.

6.3 Loudspeaker Parameters

Speakers that can be both set and retrieved are specified by the same name, with either _GET or _SET appended.

6.3.1 Speaker Control

```
KSM;SPKCONTROL_GET;<checksum>
```

Description: Returns the current speaker control enable state.

Response: KSM;ACK;[ON|OFF];<checksum>

```
KSM;SPKCONTROL_SET;[ON|OFF];<checksum>
```

Description: Sets the current speaker control enable state.

Response: KSM;ACK;<checksum>

6.3.2 Speaker Delay

```
KSM;SPKDELAY_GET;<checksum>
```

Description: Queries the loudspeaker for the current speaker delay, in milliseconds.

Response: KSM;ACK;[0-12];<checksum>

```
KSM;SPKDELAY_SET;[0-12];<checksum>
```

Description: Sets the loudspeaker delay to the specified value.

Response: KSM;ACK;<checksum>

6.3.3 Speaker Dim

`KSM;SPKDIM_GET;<checksum>`

Description: Queries the current dim state of the loudspeaker.

Response: KSM;ACK;[ON|OFF];<checksum>

`KSM;SPKDIM_SET;[ON|OFF];<checksum>`

Description: Sets the dim state of the loudspeaker.

Response: KSM;ACK;<checksum>

6.3.4 Speaker EQ Enable

`KSM;SPKEQENABLE_GET;<preset number>,<band>;<checksum>`

Description: Queries the current enable state of the specified band for the specified user preset. The preset can range from 1-8, and the band can range from 1-8.

Response: KSM;ACK;[ON|OFF];<checksum>

`KSM_SPKEQENABLE_SET;<preset number>,<band>,[ON|OFF];<checksum>`

Description: Enables or disables the specified band of the specified user preset.

Response: KSM;ACK;<checksum>

6.3.5 Speaker EQ Parameters

These messages are used to get and set preset values. The following values are available for presets:

Preset number: User preset number, from 1-8.

Band number: Band number within the preset, from 1-8.

Type:

Type	**Message Abbreviation**
Low-pass 1st order	LPF1
Low-pass 2nd order	LPF2
High-pass 1st order	HPF1
High-pass 2nd order	HPF2
Peaking EQ	PEQ
Low-shelf	LSH
High-shelf	HSH

Frequency: 10-40kHz

Gain: -24.00 to +18.00, in 0.05dB increments.

Q: 0.01-50.0, increments of 0.01.

6.3.5.1 Get

```
KSM;SPKEQPARAMS_GET;<F,U>,<preset number>,<band>;<checksum>
```

Description: Returns the current speaker EQ parameters. All filter parameters will be passed. The GET messages specifies the type of preset (Factory or User), the preset number, and the band number.

Response: KSM;ACK;<Type>,<Freq>,<Gain>,<Q>;<checksum>

6.3.5.2 Set

KSM;SPKEQPARAMS_SET;<preset number>,<band>,<Type>,<Freq>,<Gain>,<Q>;<checksum>

Description: Sets the current speaker EQ parameters for the preset number and band. It is implied that the preset is a user-preset, since factory presets cannot be changed.

Example:

```
KSM;SPKEQPARAMS_SET;4,3,LPF1,1000,3.4,0.9;<checksum>
```

Response: KSM;ACK;<checksum>

6.3.6 Speaker LED

`KSM;SPKLEDENABLE_GET;<checksum>`

Description: Queries the loudspeaker for the current LED enable state.

Response: KSM;ACK;[ON|OFF];<checksum>

`KSM;SPKLEDENABLE_SET;[ON|OFF];<checksum>`

Description: Enables or disables the loudspeaker LEDs.

Response: KSM;ACK;<checksum>

6.3.7 Speaker Locate

`KSM;SPKLOCATE;<checksum>`

Used to find a speaker through flashing LED, etc. so that the user recognizes which one it is. The loudspeaker will automatically go out of locate mode after a few seconds.

Response: KSM;ACK;<checksum>

6.3.8 Speaker Mute

`KSM;SPKMUTE_GET;<checksum>`

Description: Queries the loudspeaker for its current mute state.

Response: KSM;ACK;[ON|OFF];<checksum>

`KSM;SPKMUTE_SET;[ON|OFF];<checksum>`

Description: Sets the loudspeaker mute state to on or off. Note that ON is assumed to be muted (i.e. mute is on), and OFF is unmuted.

Response: KSM;ACK;<checksum>

6.3.9 Speaker Name

`KSM;SPKNAME_GET;<checksum>`

Description: Queries the loudspeaker for its name.

Response: KSM;ACK;<speaker name>;<checksum>

Example response:

`KSM;ACK;Left Speaker;<checksum>`

`KSM;SPKNAME_SET;New Name;<checksum>`

Description: Sets the new speaker name, up to 12 characters. Valid characters for the name are "A" - "Z", "a" - "z", "0" - "9", "-_()&", and SPACE.

Response: KSM;ACK;<checksum>

6.3.10 Speaker Preset Select

`KSM;SPKPRESET_GET;<checksum>`

Returns the currently selected preset. The message contains whether the current preset is a factory or user preset, followed by the preset number, separated by a comma. The preset number is 1-based.

Response: The response will be as shown, depending on whether a factory preset, a user preset, or no preset is currently selected. If no preset is selected, the preset number is ignored.

Factory preset: KSM;ACK;F,3;<checksum>

User preset: KSM;ACK;U,2;<checksum>

No preset: KSM;ACK;N,1;<checksum>

`KSM;SPKPRESET_SET;[U|F],[Preset number];<checksum>`

Sets the current preset to the specified preset.

6.3.11 Speaker Enable DIP Switches

Normally, when the controller connects to a loudspeaker, the DIP switches on the speaker are disabled until the user explicitly re-enables them. This message allows the user to specify that the DIP switch settings should take precedence over the controller.

`KSM;SPKDIPSWENABLE_GET;<checksum>`

Description: Gets the current enable/disable switch state. For consistency with other messages, “ON” implies that the switches are enabled, and “OFF” implies disabled.

Response: KSM;ACK;[ON|OFF];<checksum>

`KSM;SPKDIPSWENABLE_SET;[ON|OFF];<checksum>`

Description: Instructs the speaker to enable or disable the DIP switches.

Response: KSM;ACK;<checksum>

6.3.12 Speaker Standby Delay

`KSM;SPKSTANDBYDLY_GET;<checksum>`

Description: Returns the current standby delay, an integer in minutes, ranging from 5 to 120 (2 hours).

Response: KSM;ACK;<standby delay>;<checksum>

`KSM;SPKSTANDBYDLY_SET;<minutes>;<checksum>`

Description: Sets the current standby delay.

Response: KSM;ACK;<checksum>

6.3.13 Speaker Standby Enable

`KSM;SPKSTANDBYENABLE_GET;<checksum>`

Description: Returns the current standby-enable state, either ON or OFF.

Response: KSM;ACK;[ON|OFF];<checksum>

`KSM;SPKSTANDBYENABLE_SET;[ON|OFF];<checksum>`

Description: Sets the standby enable to the specified state.

Response: KSM;ACK;<checksum>

6.3.14 Speaker Trim Increment/Decrement

`KSM;SPKTRIM_INC;<increment/decrement>;<checksum>`

Description: Increments or decrements the speaker trim level in 0.5 dB increments. The value specified should be either 0.5 or -0.5. The new trim level is returned as data in the ACK message. If the increment or decrement would put the trim level out of range, the trim level will be set to the closest value that is in range. (For example, if the trim level is 5.9 and asked to increment, the new level will be 6.0.)

Example:

`KSM;SPKTRIM_INC;-0.5;<checksum>`

Response: KSM;ACK;<new trim level>;<checksum>

6.3.15 Speaker Trim Level Set

`KSM;SPKTRIMLEVEL_GET;<checksum>`

Description: Returns the current trim level in dB, from -12.0 to +6.0. The format will always be floating point, with a single digit after the decimal, in 0.5dB increments.

Response: KSM;ACK;##.#;<checksum>

Where “##.#” is the dB level, such as 4.0, -8.5, etc.

`KSM;SPKTRIMLEVEL_SET;<dB value>;<checksum>`

Description: Sets the current trim level for the loudspeaker.

Response: KSM;ACK;<checksum>

6.3.16 Speaker Firmware Version

`KSM;SPKFIRMWAREVER_GET;<checksum>`

Description: Returns the speaker firmware version. It is assumed that the version is of the form “#.#”, with a major and minor version number.

Response: KSM;ACK;[Product Code],#.#;<checksum>

“Product Code” is a three letter value identifying the product. For the current project, this value will be “57X”.

“#.#” is the version, such as 1.8, 2.3, etc.

6.3.17 Trigger Speaker Message

`KSM;SPKTRIGGER;[AMP|LIM|PAR];<checksum>`

Description: This message requests that the loudspeaker send the requested message at some point within the next 3 seconds. The data portion specifies which message to send.

AMP	Trigger AMP fault message on within 3 seconds, then off within another 3 seconds.
LIM	Trigger Limiter Engaged within 3 seconds, then off within another 3 seconds.
PAR	Trigger a Parameter Changed message within 3 seconds.

Response: KSM;ACK;<checksum>, then the requested message(s) in the specified time.

6.4 Loudspeaker-Initiated Messages

This section details messages that can be initiated by the loudspeaker, without a prompt from the controller. To make a clearer distinction between these and standard messages, the standard message prefix will be LSM (for “Loudspeaker Message”)

6.4.1 Limiter Engaged

`KSM;LSM_LIMITERENG;[ON|OFF];<checksum>`

When the state of the limiter changes (i.e. engaged/not engaged), the loudspeaker will send this message to the controller.

Response: KSM;ACK;<checksum>

6.4.2 Amp Fault

`KSM;LSM_AMPFAULT;[ON|OFF];<checksum>`

This message is sent when the loudspeaker detects an amp fault. The controller will display an appropriate message to the user. When the fault is cleared, the loudspeaker will send another message with OFF as the data.

Response: KSM;ACK;<checksum>

6.4.3 Parameter Changed

`KSM;LSM_PARAMCHANGE;<checksum>`

Description: This message indicates to the controller that the user has taken some action on the loudspeaker that has caused its state to change. The controller should refresh its settings by re-querying the loudspeaker for its current settings and updating its display accordingly.

Response: KSM;ACK;<checksum>

6.4.4 Local Firmware Update

`KSM;LSM_FWUPDATE;<checksum>`

Description: This message is sent by the loudspeaker when the user initiates a firmware download by inserting a USB flash drive into the speaker. The controller will assume that the speaker is going offline to do the upgrade, and will disconnect. When the update is complete, the loudspeaker will restart its UDP server and automatically reconnect to the controller using the standard process.

6.5 Firmware Updates

Firmware updates consist of downloading a new binary file to the loudspeaker in 1KB chunks. The workflow is described below.

6.5.1 Controller Firmware Update

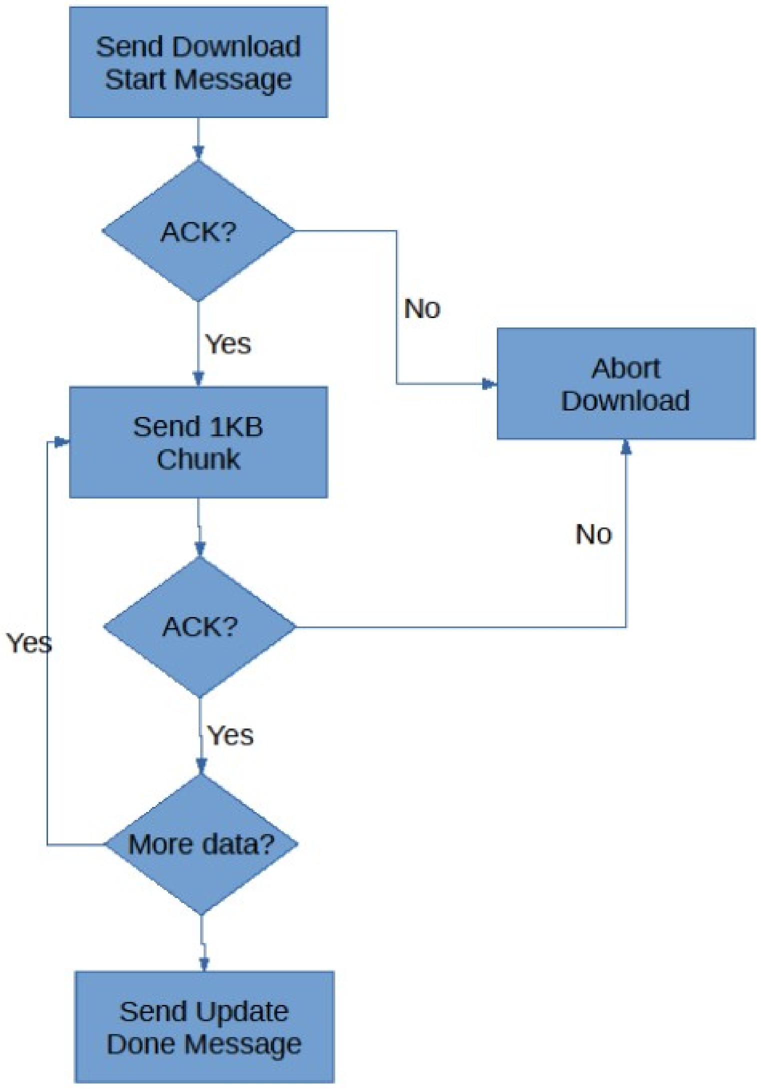

Figure 3 - Firmware Upgrade, Controller

6.5.2 Speaker Firmware Update

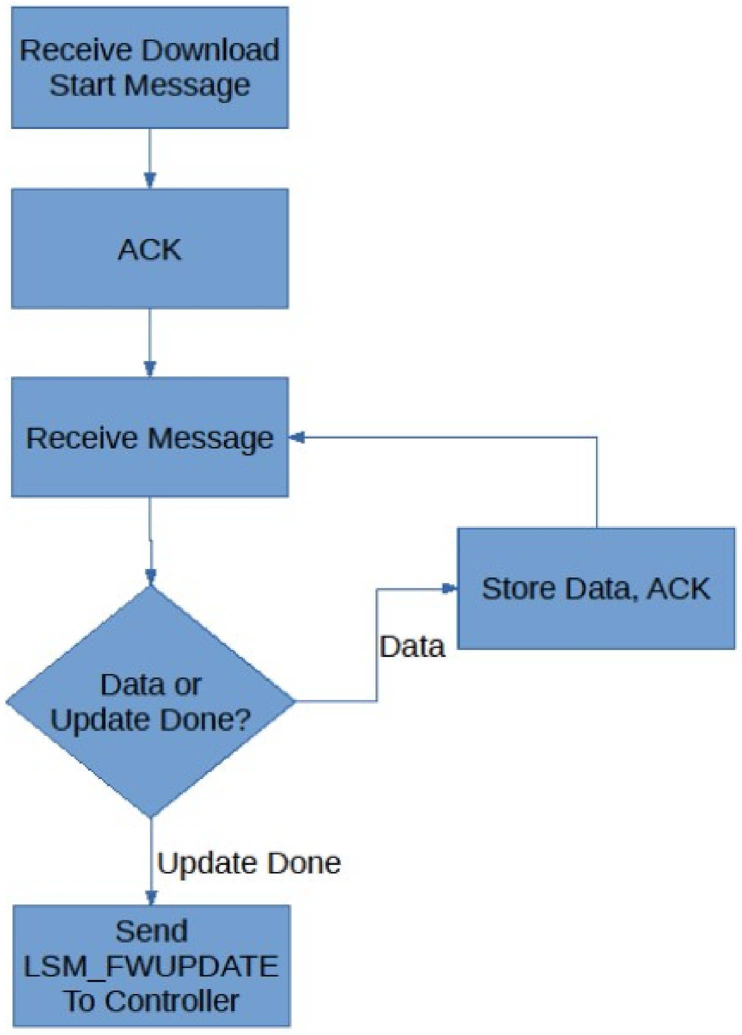

Figure 4 - Firmware Upgrade, Speaker

6.5.3 Firmware Update Messages

6.5.3.1 Start Download

```
KSM;SPKFWSTART;<num blocks>;<checksum>
```

Informs the loudspeaker that a firmware download is about to start. The parameter specifies the number of 1KB blocks that will be sent.

Response: KSM;ACK;<checksum>

6.5.3.2 Firmware Block

The firmware block message is the only message that does not follow the standard message format, since it contains binary data. The message is the following format:

```
KSM;FWDATA;<block number>,<data length>,<crc32>;[binary data]
```

Parameters:

Block number: The current block being sent.

Data length: The number of data bytes contained in the message. This is typically 1024, but may be less for the final block.

Crc32: The CRC32 calculation of the data.

Binary data: The data, as a binary blob.

There is no final checksum for this message.

Response: KSM;ACK;<checksum>

6.5.3.3 Download Complete

KSM;SPKFWCOMPLETE;[PGM|NPGM]<checksum>

This message is sent when the download is complete. The loudspeaker can begin to flash its memory and perform whatever steps necessary to complete the upgrade.

The data parameter indicates whether the loudspeaker should immediately begin to program the new firmware. If NPGM is specified, the loudspeaker should not program the firmware. (This is typically only used for testing. The normal method will indicate PGM.

Response: KSM;ACK;<checksum>

7 Errors

The following errors can be returned by a NACK message. The code is a 16-bit value, with the two upper bits reserved to indicate the type of error:

Bit 15: The message itself is mal-formed.

Bit 14: The message was correct, but an error occurred while processing it.

Note that the code is sent as a decimal integer.

Code (Hex)	**Code (Decimal)**	**Error**	**Meaning**
0x8001	32769	Checksum error	The checksum of the message was incorrect.
0x8002	32770	Unrecognized message	The specified message was correctly formed but is unrecognized by the recipient.
0x8003	32771	Malformed message	The message is formed incorrectly.
0x4001	16385	Invalid data	The data specified in the message is invalid. (e.g. expects an integer argument, received a string.)
0x4002	16386	Out of range	The value specified in the data is out of range.
0x4003	16387	Data too long	The specified data is too long.
0x4004	16388	CRC32 error	The CRC32 value in a firmware download block did not match.