REVISIONS

REV #	ECN #	Description of Change	Approved	Quality	Date
1	NDI	Initial Release	SJS	DFH	9/20/10
2	0111004	Added PA Present to MFGInfo.	SJS	DFH	12/02/10
3	0611002	Updated Surround Enable Audio Commands	SJS	DFH	6/5/11
4	E1604008	Update Appendix, Add Headphone & WOW Override, Diag Input, add reserved bytes to MFG info, expanded SPI and examples	SJS	RV	04/20/16
5	E1708013	modified unsolicited from Detailed Status, fix DB download in 5.3.1.1, Update SetOutputChannels with mask flags.	SJS	RV	08/23/17

Forte Series Digital Amplifier

Interface Design Description

Description

Forte Series Interface Design Description

1 Introduction

This document describes the protocol that is used for all system inter-board communication for the Alto line of Digital Audio amplifiers.

Data Dictionary

AMP/Amp	Amplifier
BCC	Block Check Character
DOC	Document
IDD	Interface Design Description
PA	Passenger Address
PRS	Product Requirements Specification
SRS	Software Requirements Specification
DataBase	A collection of data used to customize the EQ and audio function of the unit
Record	One of several of the custom data files stored in the DataBase
Configuration	A collection of user settable parameters of the system

2 References

106500PS	Product Requirements Specification for the DA-700 series
106500CT	Forte Series Configuration Table Definitions

3 Physical Layer

See Appendix Sections for Details regarding the Physical Layers.

4 Application Layer Protocol

4.1 Message Format

The external message format may vary depending upon the physical interface used. See the Appendix section for details regarding the message format.

The Internal System Message format is common to all physical interfaces, and it is described as follows (See Appendix for any Prefix or Suffix added to the External Message)**:**

Class	Oper	Func	Seq	Length	Data Payload	BCC
8 bits	8 bits	8 bits	8 bits	8 bits	<Data 1>…<Data 26>	8 bits

- The first five bytes of the message are the header which are **mandatory**
- Data payload is optional so a data **length** of ZERO is valid
- Unused payload bytes beyond the last byte of valid data are functionally ignored.
- **Seq**uence **Id**entifier is a numeric value that uniquely identifies a data transport protocol. It is a rolling number starting from 0 through 255. In the case of a parsing error or a missing number in the Seq series; the generated error message echoes the Seq of the offending message. When the Host is transmitting a message, it should increment the Seq for the next message. When the Client receives a message it will respond with the same Seq except for the Heartbeat. If a message is missing, the Host may resend the message based on the importance of the message.
- Client responds with a Heartbeat any time the Host sends the heartbeat function.
- Client responds with an AckNak message, for every "Set" command and a Response message for every "Get" command.
- There is a BCC-8 byte appended to the end of message.
- The Message Acknowledge system returns the following data in the Ack/Nak byte of all messages:
    - 0x00 (0) Good Status
    - 0x40 Busy, Waiting to Respond
    - 0xFF (-1) Invalid BCC
    - 0xFE (-2) Invalid Argument
    - 0xFD (-3) Function Not Executed
    - 0xFC (-4) Invalid CRC
    - 0xFB (-5) PA not responding
    - 0xDF TDB Format Mismatch
    - 0xDE Transfer Invalid Address
    - 0xDD Transfer Missing Data
    - 0xDC Erase Burn Failed
    - 0xDB Not Weight On Wheels
    - 0xDA Transfer Not Active
    - 0xD9 Transfer Invalid Record
    - 0xD8 Transfer Extra Data
    - 0xD7 Segment Not Active
    - 0xD6 Flow Control Violation
    - 0xD5 Segment Already Active
    - 0xD4 Hardware Fault
    - 0xCF PA Not indicated in Main Config
    - 0xCE No PA net connection
    - 0xCD Diag Session Not Active
    - 0xCC Flash Read Failure
    - 0xCB Record Failed Validation
    - 0xCA Default DB Loaded
    - 0xBF Requested DB Loaded
- The Message Acknowledge system returns the following data in the Seq byte of all messages:
    - Next Sequence Number for Ack
    - Echo the ID received for Nak
- Data (Endianess)
    - By convention, all scalar data types are presented in big Endian format.
    - In bit fields, bits are numbered from right to left:

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0

- In Byte fields, bytes are number from right to left:

Byte 1	Byte 0

- **Errors**: In the event of errors, for example, Invalid Argument, Function Not Executed and Invalid BCC, the Error message is returned using the same sequence ID as the offending message.
- BCC is calculated as a running XOR of each byte in the message, excluding the prefix (32 bytes).

4.2 Classes

Table of possible Class IDs is shown below:

Class ID	Class Name
0x00	ControlChannel
0x01	Diagnostic
0x02	SoftwareTransfer
0x03	AircraftInformation
0x04	Analog
0x07	Headphone
0x08	Amplifier
0x09	PAChime

4.3 Operations

Table of possible Operation IDs is shown below:

Operation ID	Name	Description
0x00	Set	Host Writes data
0x01	Get	Host Reads data
0x03	Inc	Sends Increment command to a control
0x04	Dec	Sends Decrement command to a control
0x05	Unsolicited	Sent by Client to Host
0x06	Response	Data returned by Client from a Get Operation
0x07	ACKNAK	Response to a message that was received
0x08	Status	Used for Heartbeat messages
0x80	Restart	Used for Restart messages

4.4 Functions

Each of the following sections lists all the possible Function tokens associated with a particular Class.

4.4.1 ControlChannel 0x00 Class Functions

The ControlChannel Class provides an interface to monitor and manage digital and analog device controls on the Main and PA boards.

Function ID	FunctionName
0x20	Heartbeat
0x21	HeartbeatTimeoutOverride
0x24	DataExchange
0x30	PowerOnInit

4.4.1.1 Heartbeat (Class 0x00, Function 0x20)

After the Host is powered up and ready, it starts transmitting a Heartbeat status to the Client at regular time intervals. If the Host does not receive a reply for a predetermined number of Heartbeat requests, the Host shall assume that the Client is not running. The Host, however, shall continue transmitting a Heartbeat status anticipating an eventual reply.

The Client acknowledges that it is ready by responding to the Host's Heartbeat status message with a message that includes a 4 byte data counter. If the Client does not receive a Heartbeat for the timeout period specified in the Configuration Table, it assumes that Host is not ready, goes into Mute mode. Additionally, the client may issue a hardware reset if the host hardware specification recommends it.

The Host shall stop transmitting a Heartbeat during Software Downloads and when updating the Active Configuration Database. The Client will not respond to the Heartbeat during these states.

The Client will respond to a Heartbeat command with a Heartbeat status. This transaction will be handled same as a SET command, so in the case of SPI interface, a response interrupt will be issued.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Counter Byte 0-3	Byte	4	0 – 0xFFFFFFFF	Counter

<u>Operations Sent</u>: *Status*

Heartbeat Status: Status (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x00	0x08	0x20	N	0	26 bytes x 0x00	0xyy

<u>Operations Returned</u>: *Status*

Heartbeat Status: Status (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Padding	BCC
0x00	0x08	0x20	N	4	Counter Byte 3 MSB	Counter Byte 2	Counter Byte 1	Counter Byte 0 LSB	22 bytes x 0x00	0xyy

4.4.1.2 HeartbeatTimeoutOverride (Class 0x00, Function 0x21)

The Host can temporarily override the HB timeout. If this timeout expires, then the unit will execute the prescribed timeout action and afterwards go back to using the default timeout value read from the configuration table

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Timeout	Byte	1	0 - 255	Timeout in Seconds

Table of HeartbeatTimeoutOverride Data Fields

Operations Sent: ***Set***

HostStatus Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x00	0x00	0x21	N	1	Timeout	25 bytes x 0x00	0xyy

Operations Returned: ***AckNak***

HostStatus AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x00	0x07	0x21	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.1.3 DataExchange (Class 0x00, Function 0x24)

Data exchange is a message that the SPI Master sends to the Slave whenever it is only interested in Receiving a message. Because SPI always executes a data exchange, some data must be sent to the Slave in the process of receiving a Slave message. In order to avoid transferring random or meaningless data, the following should be sent by the Host in cases where the action is only executed for reception of data.

A response message intended for the host will remain in the cue until a valid DataExchange sequence is received.

Operations Sent:

DataExchange: (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x00	0x02	0x24	Next	0	26 bytes x 0x00	0xyy

4.4.1.4 PowerOnInit (Class 0x00, Function 0x30)

Power-On Initialization is a response message that is sent once to the Host when the Client is initially powered on or if the Client has been reset during operation. The Power-On Initialization should indicate to the Host that the Client will require the commands necessary to establish (or re-establish) a session. Such commands might include setting the zone volume, channel Audio Format and types, etc.

The PowerOnInit message will only be sent after the Client receives a Heartbeat message in order to first insure that the Host is up and running.

Operations Sent: ***Unsolicited Status***

PowerOnInit: Unsolicited Status (Client to Host):

Class	Oper	Func	Seq	Length	Padding	BCC
0x00	0x05	0x30	Next	0	26 bytes x 0x00	0xyy

4.4.2 Diagnostic Class 0x01 Functions

The Diagnostic Class provides an interface to monitor and manage devices in the system.

Function ID	Function Name
0x09	PrepareForRestart
0x63	Restart
0x10	DeviceDetailedStatus
0x20	HostStatus
0x40	ConfigDataParameter
0x70	ManufacturingInformation
0x74	TuningDatabaseInfo
0x75	TuningDBRecordInfo
0x78	ActiveConfigDatabase

4.4.2.1 PrepareForRestart (Class 0x01, Function 0x09)

PrepareForRestart is a message that should be sent when preparing to send the Restart (0x63) message. This is implemented as a redundant command to insure that a Restart is truly intended.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Board	Enum	1	0 - 1	0: Main, 1: PA

Table of PrepareForRestsart Data Fields

Operations Sent: *Set*

PrepareForRestart: Set (Host to Client):

Class	Oper	Func	Seq	Length	Data1	Padding	BCC
0x01	0x00	0x09	Next	1	Board	25 bytes x 0x00	0xyy

Operations Returned: *AckNak*

PrepareForRestart: AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x09	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.2.2 Restart (Class 0x01, Function 0x63)

Restart puts the amplifier into Watch Dog Timeout and lets the hardware reset trigger. This command must be preceded by PrepareForRestart.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Board	Enum	1	0 - 1	0: Main, 1: PA

Table of Restsart Data Fields

Operations Sent: *Set*

Restart: Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x80	0x63	Next	1	Board	25 bytes x 0x00	0xyy

Operations Returned: *AckNak*

Restart: AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x63	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.2.3 DeviceDetailedStatus (Class 0x01, Function 0x10)

Power Supply voltage ranges are classified as Zones A through C. Voltages within Zone A are normal, and the system is fully functional. Voltages within Zone B indicate a decrease in power signaling the audio Power Amps to be muted and shutdown. Voltages within Zone C indicate imminent loss of power.

The on-board temperature sensor ranges for warning and shutdown are TBD. This information is reported in the overall status field. Power Amplifier temperature information is reported in the each individual power amplifier statuses.

The database status is self-evident. If the database has not been loaded, or it is corrupt, it needs to be loaded again. If the Main and PA module detailed status reports a serial Flash hardware fault, the database is considered not loaded.

The power amplifier channel status in regards to a speaker open circuit condition depends upon the speaker configuration. Channels not included in the configuration do not report opens. Shorts are always reported. The Main and PA initialization turn-on sequence detects opens on configured channels.

Name	Data Type	Size Bytes	Range	Description
Overall	Bit field	2	Bit 0: 0 – 1 Bit 1: 0 – 1 Bits 2 – 3: 0 – 2 Bits 4 – 5: 0 – 2 Bits 6 – 7: 0 – 2 Bits 8 – 9: 0 – 2 Bits 10 – 11: 0 – 2 Bits 12 – 13: 0 – 2 Bit 14: 0 – 1 Bit 15: 0 – 1	All: 0:normal, 1:fault Init: 0:normal, 1:initializing Voltage Zone Main: 0:A, 1:B, 2:C Voltage Zone PA: 0:A, 1:B, 2:C Temperature Main: 0:normal, 1:warning, 2:shutdown Temperature PA: 0:normal, 1:warning, 2:shutdown Database Main: 0:normal, 1:not loaded, 2:corrupt Database PA: 0:normal, 1:not loaded, 2:corrupt Power Amps Main: 0:normal, 1:fault Power Amps PA: 0:normal, 1:fault
MVolt	Scaled Short	2	0 – 280 + overrange	Amp Board Power Input voltage: units are tenths of a volt
PVolt	Scaled Short	2	0 – 280 + overrange	PA Board Power Input voltage: units are tenths of a volt
MTemp	Scaled Signed Short	2	-400 – +1200	Amp Board Temperature: units are tenths of degree C
PTemp	Scaled Signed Short	2	-400 – +1200	PA Board Temperature: units are tenths of degree C
Main Amp 1	Bit field	1	Bits 0 – 1: 0 - 2 Bits 2 – 3: 0 - 2 Bits 4 – 7: 0	Temperature: 0:normal, 1:warning, 2:shutdown Voltage: 0:normal, 1:under, 2:over Reserved TBD for DC offset and other status
Main Amp 2	Bit field	1	Bits 0 – 1: 0 - 2 Bits 2 – 3: 0 - 2 Bits 4 – 7: 0	Temperature: 0:normal, 1:warning, 2:shutdown Voltage: 0:normal, 1:under, 2:over Reserved TBD for DC offset and other status
PA Amp 1	Bit field	1	Bits 0 – 1: 0 - 2 Bits 2 – 3: 0 - 2 Bits 4 – 7: 0	Temperature: 0:normal, 1:warning, 2:shutdown Voltage: 0:okay, 1:under, 2:over Reserved TBD for DC offset and other status

Name	Data Type	Size Bytes	Range	Description
Main Channel 1 through Main Channel 8 PA Channel 1 through PA Channel 4	Bit field	1	Bit 0: 0 - 1 Bit 1: 0 - 1 Bit 2: 0 - 1 Bit 3: 0 - 1 Bit 4: 0 - 1 Bit 5: 0 - 1 Bit 6: 0 - 1 Bit 7: 0	Short to Ground: 0:not short, 1:short Short to VCC: 0:not short, 1:short Open Load: 0:not open, 1:open Short Load: 0:not short, 1:short DC Offset: 0:no error, 1:error Over Current: 0: no error, 1:error Current detected (tweeter test), 0:yes, 1:no Reserved
System Status	Bit field	1	Bit 0: 0 - 1 Bit 1: 0 - 1 Bit 2: 0 - 1 Bit 3: 0 - 1 Bit 4: 0 - 1 Bit 5: 0 - 1 Bit 6: 0 - 1 Bit 7: 0 - 1	Codec 1 Fault Codec 2 Fault DSP Fault EEProm Memory Fault Flash Memory Fault Fan Fault Network Host Fault PA no response Fault

Table of Detailed Status Data Fields

Operations Sent: ***Get***

DeviceDetailedStatus Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x01	0x01	0x10	N	0	26 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak, Unsolicited***

DeviceDetailedStatus Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8
0x01	0x06	0x10	N	26	Overall	Overall	MVolt	MVolt	PVolt	PVolt	MTemp	MTemp

Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17
PTemp	PTemp	Main Amp 1	Main Amp 2	PA Amp 1	Main Ch 1	Main Ch 2	Main Ch 3	Main Ch 4

Data 18	Data 19	Data 20	Data 21	Data 22	Data 23	Data 24	Data 25	Data 26	BCC
Main Ch 5	Main Ch 6	Main Ch 7	Main Ch 8	PA Ch 1	PA Ch 2	PA Ch 3	PA Ch 4	System Status	0xyy

DeviceDetailedStatus AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x10	N	1	Ack Nak	25 bytes x 0x00	0xyy

DeviceDetailedStatus Unsolicited (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7	Data 8
0x01	0x05	0x10	N	26	Overall -MSB	Overall -LSB	MVolt	MVolt	PVolt	PVolt	MTemp	MTemp

Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17
PTemp	PTemp	Main Amp 1	Main Amp 2	PA Amp 1	Main Ch 1	Main Ch 2	Main Ch 3	Main Ch 4

Data 18	Data 19	Data 20	Data 21	Data 22	Data 23	Data 24	Data 25	Data 26	BCC
Main Ch 5	Main Ch 6	Main Ch 7	Main Ch 8	PA Ch 1	PA Ch 2	PA Ch 3	PA Ch 4	System Status	0xyy

4.4.2.4 HostStatus (Class 0x01, Function 0x20)

Enables network host to communicate status information to Client for illumination of fault light.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Host Fault	Enum	1	0 - 1	0: No Faults 1: Fault condition
Host Timeout Detected	Enum	1	0 - 1	0: Normal operation 1: Host Timeout Detected

Table of HostStatus Data Fields

Operations Sent: ***Get, Set***

HostStatus Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x01	0x01	0x20	N	0	26 bytes x 0x00	0xyy

HostStatus Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x00	0x20	N	1	Host Fault	25 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

HostStatus Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x01	0x06	0x20	N	2	Host Fault	Host Timeout	24 bytes x 0x00	0xyy

HostStatus AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x20	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.2.5 ConfigDataParameter (Class 0x01, Function 0x40)

This command dynamically updates, in real time, the Active Configuration data values stored in EEPROM on the Main and PA boards, and sends a message to the DSP to update the parameters. This command also retrieves Active and Default Configuration data including their associated preset factory limits, data types, descriptions, etc., so that a list may be generated, viewed and/or printed. If a value stored in the database takes up more than one byte, the byte desired may be retrieved. Refer to Alto Dwg # 106500CT, *Forte Series Configuration Table Definitions.xls* for details pertaining to configuration table contents.

Name	Data Type	Size in Bytes	Range	Description
Board	Enum	1	0 - 1	0: Main, 1: PA
Index	Byte	1	0 - 255	Index of the Parameter Cell
Multiple	Byte	1	1-4	This parameter indicates the number of bytes in the value, normally set to 1. If Value does consist of more than one byte, then multiple messages are sent with the MSB value in the first message sent. Note that Low, High, and Default values are sent in the same manner.
Parameter	Byte	1	0 - 255	Parameter Value being set
ASCII description	Byte (ASCII)	20	Valid ASCII values	Up to 20 character description of the data field
Value	Byte	1	0 - 255	The current value in the DB for this parameter
Low limit	Byte	1	0 - 255	The factory lower limit for the parameter
High limit	Byte	1	0 - 255	The factory higher limit for the parameter
Default	Byte	1	0 - 255	The factory default value
DataType	Bitfield	1	0 - 255	Bits 0-3: Data type (see table below) Bit 4: Public = 0, 1 = Private Bit 5-8 reserved for future use

Table of ConfigDataParameter Data Fields

DataType, Bits 0-3	Datatype
0000	Spare
0001	On/Off type (0=off, 1=on) Unsigned
0010	Chime type (1-6 represent different chime tones) Unsigned
0011	Volume type (each increment represents +/- 1 db) Signed -128 - +127
0100	Ohm type (0=open, 1 = 1 ohm, 2 = 2 ohms, etc.) Unsigned
0101	Host type (0 = None, 1= Alto, 2 = HW, 3 = GAC) Unsigned
0110	Mute type (0 = None, 1-4 indicate the channels muted) Unsigned
0111	Delay type (0 – 255) Unsigned
1000	Spare
1001	Spare
1010	Spare
1011	Spare
1100	Spare
1101	Spare
1110	Spare
1111	Spare

DataType Bits Definition Table

Operations Sent: ***Get, Set***

ConfigDataParameter Get (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x01	0x01	0x40	N	3	Board	Index	Multiple	22 bytes x 0x00	0xyy

ConfigDataParameter Set (Host to Client): (Weight on Wheels active, sets Active Config values)

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Padding	BCC
0x01	0x00	0x40	N	4	Board	Index	Multiple	Parameter	21 bytes x 0x00	0xyy

Operations Returned: **Response, *AckNak***

ConfigDataParameter Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x01	0x06	0x40	N	26	Index	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII

Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17	Data 18	Data 19
ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII	ASCII

Data 20	Data 21	Data 22	Data 23	Data 24	Data 25	Data 26	BCC
ASCII	ASCII	Value	Low	High	Default	Datatype	0xyy

ConfigDataParameter AckNak:

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x40	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.2.6 ManufacturingInformation (Class 0x01, Function 0x70)

This DataBlock is located in Serial EEPROM. It contains the information about Software and Hardware, such as SW Code version, and HW version for PA and Main. 2 byte values represent a Major.Minor version. Value of each byte can be 00 – 99. Reporting of version information should always be padded to 2 digit (ex: 07.02 not 7.2).

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Main SW version	Byte	2	(00-99) . (00-99)	Main Software version (major.minor)
Main Signal Board HW version	Byte	1	00-255	Main Signal board Hardware version (Integer, should match Hardware BOM version)
Reserved	Byte	1	00-255	Reserved for Alto use
Main Amp Board HW version	Byte	1	00-255	Main Amp board Hardware version (Integer, should match Hardware BOM version)
Reserved	Byte	1	00-255	Reserved for Alto use
PA SW version	Byte	2	(00-99) . (00-99)	PA Software version (major.minor)
PA HW version	Byte	1	00-255	PA Hardware version (Integer, should match Hardware BOM version)
Reserved	Byte	1	00-255	Reserved for Alto use
Unit Serial Number	Unsigned Double	4	100000-999999	Unit serial number (6 decimal digits)
PA board Installed	Byte	1	00-01	Indicates that this unit is has PA option, 00 = no PA, 01 = PA
Amplifier Type	Byte	1	00-255	Code indicating Amplifier Type 1=PA 2=DA2XX etc. 10+ = DX 20=DZ
Amplifier Sub Type	Byte	1	00-255	Code indicating Amplifier Sub Type 10=DAX10 etc.
Amplifier Number	Byte	1	00-255	Indicates Amp# for multi-amp install 0 for undetermined
Active Record	Byte	1	00-255	Indicates Active record# in DB Used for tuning and diagnostic purposes
MP REV	Byte	1	00-255	Indicates the MP Rev (MOD Level) used to build unit
Main SW Part#	Unsigned Double	3	100000-999999	Main Software Alto Part# (6 decimal digits). The MSB is not sent, it should be set to 0.
Main SW Part#	Unsigned Double	3	100000-999999	PA Software Alto Part# (6 decimal digits) The MSB is not sent, it should be set to 0.

Table of Manufacturing Information Data Fields

Operations Sent: ***Get ManufacturingInformation Get (Host to Client):***

Class	Oper	Func	Seq	Length	Padding	BCC
0x01	0x01	0x70	N	0	26 bytes x 0x00	0xyy

ManufacturingInformation Set (Host to Client): (Factory Only!!!)

Class	Op	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
0x01	0x00	0x70	N	22	Main Sig HW ver	Reserved1	Main Amp HW ver	Reserved2	PA HW ver	Reserved3

Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17-22
Unit S/N	Unit S/N	Unit S/N	Unit S/N	PA Present	Amp Type	Amp Sub Type	Amp Number	Active Record	MP Rev	Ignored, read only

Padding	BCC
4 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

ManufacturingInformation Response from Get (Client to Host):

Class	Op	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6
0x01	0x06	0x70	N	26	Main SW ver	Main SW ver	Main Sig HW ver	Reserved1	Main Amp HW ver	Reserved

Data 7	Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15
PA SW ver	PA SW ver	PA HW ver	Reserved3	Unit S/N MSB	Unit S/N byte 3	Unit S/N byte 2	Unit S/N LSB	PA Present

Data 16	Data 17	Data 18	Data 19	Data 20	Data 21	Data 22	Data 23
Amp Type	Amp Sub Type	Amp Number	Active Record	MP Rev	Main SW PN Byte3	Main SW PN Byte2	Main SW PN LSB

Data 24	Data 25	Data 26	BCC
PA SW PN Byte3	PA SW PN Byte2	PA SW PN LSB	0xyy

ManufacturingInformation AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x70	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.2.7 TuningDatabaseInfo (Class 0x01, Function 0x74)

The Tuning Database is made up of tuning records (up to 25). Each record contains specific tuning information custom to a particular aircraft cabin or model and includes data and parameters for the DSP, Codecs, and hardware registers.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Response#	Byte	1	0 - 2	Which message of the 3 responses
Board	Enum	1	0 - 1	0: Main, 1: PA
VersionNo	Byte	1	0 - 255	Database Version number
Database Revision (major)	Byte	1	0 - 255	Database Revision (major)
Database Revision (minor)	Byte	1	0 - 255	Database Revision (minor)
Author's info	ASCII	6	NA	Author's Initials e.g. XZCD
Date	Hex	6	NA	Date: YYMMDD stored as 6 integers e.g. 0x14, 0x09, 0x01, 0x01, 0x02, 0x04 for 2009.11.24
Aircraft MfgID	Byte	1	0 - 255	Aircraft Manufacturer ID
Aircraft Model	Byte	1	0 - 255	Aircraft Model ID
Records	Byte	1	0 -48	Number of records in Database
RecordMask	Bit Field	6	Byte 6: records 1-8 Byte 5: records 9-16 Byte 4: records 17-24 Byte 3: records 25-32 Byte 2: records 33-40 Byte 1: records 41-48	Flag mask for which records are loaded and valid in the database. A '1' bit indicates record is loaded and valid.
Comment	ASCII	40	NA	Notes and comments on TDB content

Table of TuningDatabaseInfo Data Fields

Operations Sent: ***Get***

TuningDatabaseInfo Get (Host to Client):

Class	Oper	Func	Seq	Length	Data1	Padding	BCC
0x01	0x01	0x74	N	1	Board	25 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

TuningDatabaseInfo Response1 from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x01	0x06	0x74	N	26	Resp# (0)	Board	Ver no	DB rev major	DB rev minor	Author info	Author info

Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17
Author info	Author info	Author info	Author info	Date YY	Date YY	Date MM	Date MM	Date DD	Date DD

Data 18	Data 19	Data 20	Data 21	Data 22	Data 23	Data 24	Data 25	Data 26	BCC
Aircraft Mfg ID	Aircraft Model	Num Records	Record Mask1	Record Mask2	Record Mask3	Record Mask4	Record Mask5	Record Mask6	0xyy

TuningDatabaseInfo Additional Response2 from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x01	0x06	0x74	N	26	Resp# (1)	Board	Cmnt. Char 1	Cmnt. Char 2	Cmnt. Char 3	Cmnt. Char 4	Cmnt. Char 5

Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17
Cmnt. Char 6	Cmnt. Char 7	Cmnt. Char 8	Cmnt. Char 9	Cmnt. Char 10	Cmnt. Char 11	Cmnt. Char 12	Cmnt. Char 13	Cmnt. Char 14	Cmnt. Char 15

Data 18	Data 19	Data 20	Data 21	Data 22	Data 23	Data 24	Data 25	Data 26	BCC
Cmnt. Char 16	Cmnt. Char 17	Cmnt. Char 18	Cmnt. Char 19	Cmnt. Char 20	Cmnt. Char 21	Cmnt. Char 22	Cmnt. Char 23	Cmnt. Char 24	0xyy

TuningDatabaseInfo Additional Response3 from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x01	0x06	0x74	N	18	Resp# (2)	Board	Cmnt. Char 25	Cmnt. Char 26	Cmnt. Char 27	Cmnt. Char 28	Cmnt. Char 29

Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17	Data 18
Cmnt. Char 30	Cmnt. Char 31	Cmnt. Char 32	Cmnt. Char 33	Cmnt. Char 34	Cmnt. Char 35	Cmnt. Char 36	Cmnt. Char 37	Cmnt. Char 38	Cmnt. Char 39	Cmnt. Char 40

Padding	BCC
8 bytes 0x00	0xyy

TuningDatabaseInfo AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x74	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.2.8 TuningDBRecordInfo (Class 0x01, Function 0x75)

Each record in the Tuning Database is customized for a particular cabin or model installation. A check should be made using TuningDatabaseInfo to insure that the particular requested record is valid. If a record is invalid, then a Nak will be returned. NOTE: *A second and third message containing additional fields will be sent as an unsolicited message as a result of the GET*

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Response#	Byte	1	0 - 2	Which message of the 3 responses
Board	Enum	1	0 - 1	0: Main, 1: PA
Record #	Byte	1	1 – 25	Record number to fetch info from
EQ ID (major)	Byte	1	0 - 255	Record EQ number (unique number for each different EQ image)
EQ ID (minor)	Byte	1	0 - 255	Record EQ number (unique number for each different EQ image)
Record version	Byte	1	0 - 255	Version used for running change or improvement on record EQ
Author's info	ASCII	6	NA	Author's Initials e.g. XZAAA
Date	ASCII Hex	6	NA	Date: YYMMDD stored as 6 integers e.g. 0x14, 0x09, 0x01, 0x01, 0x02, 0x04 for 2009.11.24
Comment	ASCII	40	0 - 255	Notes and comments on Record content

Table of TuningDBRecordInfo Data Fields

Operations Sent: ***Get***

TuningDatabaseRecordInfo Get (Host to Client):

Class	Oper	Func	Seq	Length	Data1	Data2	Padding	BCC
0x01	0x01	0x75	N	2	Board	Record	24 bytes x 0x00	0xyy

Operations Returned: ***Response, Unsolicited additional response, AckNak***

TuningDatabaseRecordInfo Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x01	0x06	0x75	N	26	Resp# (0)	Board	Record	EQ ID (MSB)	EQ ID (LSB)	Record Version	Author Info 1

Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17
Author Info 2	Author info 3	Author Info 4	Author info 5	Author info 6	Date Y	Date Y	Date M	Date M	Date D

Data 18	Data 19	Data 20	Data 21	Data 22	Data 23	Data 24	Data 25	Data 26	BCC
Date D	Comment Char 1	Comment Char 2	Comment Char 3	Comment Char 4	Comment Char 5	Comment Char 6	Comment Char 7	Comment Char 8	0xyy

TuningDatabaseRecordInfo Additional Response1 from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x01	0x06	0x75	N	26	Resp# (1)	Board	Cmnt. Char 9	Cmnt. Char 10	Cmnt. Char 11	Cmnt. Char 12	Cmnt. Char 13

Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17
Cmnt. Char 14	Cmnt. Char 15	Cmnt. Char 16	Cmnt. Char 17	Cmnt. Char 18	Cmnt. Char 19	Cmnt. Char 20	Cmnt. Char 21	Cmnt. Char 22	Cmnt. Char 23

Data 18	Data 19	Data 20	Data 21	Data 22	Data 23	Data 24	Data 25	Data 26	BCC
Cmnt. Char 24	Cmnt. Char 25	Cmnt. Char 26	Cmnt. Char 27	Cmnt. Char 28	Cmnt. Char 29	Cmnt. Char 30	Cmnt. Char 31	Cmnt. Char 32	0xyy

TuningDatabaseRecordInfo Additional Response2 from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x01	0x06	0x75	N	10	Resp# (2)	Board	Cmnt. Char 33	Cmnt. Char 34	Cmnt. Char 35	Cmnt. Char 36	Cmnt. Char 37

Data 8	Data 9	Data 10	Padding	BCC
Cmnt. Char 38	Cmnt. Char 39	Cmnt. Char 40	16 bytes x 0x00	0xyy

TuningDatabaseRecordInfo AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x75	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.2.9 ActiveConfigDatabase (Class 0x01, Function 0x78)

Located within Serial Flash of Main and PA boards is a complete set of default Tuning Databases. The Client copies the requested Default Configuration data into EEPROM and then to the DSP. The values associated with the Active Database are located in the first block of EEPROM. This function only gets executed if the request is different from what is currently active.

For Default Configuration data refer to section Configurable Data Tuning Database.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Board Info	Enum	1	0 - 1	0: Main, 1: PA
DB_ID	Byte	1	0 – 25	Tuning and Configuration Database Identification Number Note: Selecting 0 for the DB_ID will cause the unit to use the Default Record hard coded into the firmware.

Table of ActiveConfigDatabase Data Fields

Operations Sent: ***Get, Set***

ActiveConfigDatabase Get (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x01	0x78	N	1	Board Info	25 bytes x 0x00	0xyy

ActiveConfigDatabase Set (Host to Client): (Weight on Wheels must be active if available)

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x01	0x00	0x78	N	2	Board Info	DB_ID	24 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

ActiveConfigDatabase Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x01	0x06	0x78	N	2	Board Info	DB_ID	24 bytes x 0x00	0xyy

ActiveConfigDatabase AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x01	0x07	0x78	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.3 SoftwareTransfer Class 0x02 Functions

Func	Name
0x10	DownloadStart
0x11	DownloadSegment
0x12	DownloadEnd
0x13	DownloadAbort
0x20	UploadStart (reserved)
0x21	UploadSegment (reserved)
0x22	UploadEnd (reserved)
0x23	UploadAbort (reserved)
0x30	TransferData
0x31	TransferStatus

Oper	Name
0x80	MethodStart
0x82	MethodAction
0x84	MethodStop
0x86	MethodResult

The SoftwareTransfer Class supports download and upload of memory spaces both physical and logical. Examples of physical space are: Serial Flash, Code Flash, EEPROM and RAM. An example of logical is Tuning Database.

Execution of Function within this Class requires that “Weight on Wheels” is active.

The following section(s) describe the Func available (not reserved) to the SoftwareTransfer Class.

4.4.3.1 DownloadStart (Class 0x02, Function 0x10)

In order to initiate a download, a DownloadStart (Oper MethodStart) must be sent. A set of arguments instructs the DAPA700 how to prepare for the download. The DAPA700 should enter a Diagnostic Session in Programming Mode before sending this message

If “Weight on Wheels” is not active or a transfer (download or upload) is already active or if implemented a diagnostic session in programming mode is not active, a NAK message with error code “Functioned Not Executed” is sent.

The sender of this message must wait for a response before continuing to send SoftwareTransfer messages. In the case of preparing the Serial Flash for a download of a complete Tuning DB (Header and all Records) it typically may take 15 seconds before returning the ACK if the erase entire memory unit is selected.

Name	Data Type	Byte Size	Range	Description
Board Network ID	Enum	1	0 – 1	0: Main, 1: PA
Memory Type	Enum	1	 0x00 0x01 0x02 0x03 0x10 0x11 0x12 0x13 0x18	Physical Memory Type Serial Flash (supported) Serial EEPROM (reserved) Internal Code Flash (reserved) Internal RAM (reserved) DSP Program (reserved) DSP Parameter (reserved) DSP Data (reserved) DSP Register (reserved) CODEC Register (reserved)
Memory Unit	Enum	1	1-1	Placeholder currently always 1
Flags	Bitfield	1	 Bit 0 Bit 1 Bits 2 - 7	Flags 0: ignore 1: auto-abort transfer session 0: ignore 1: erase memory unit Reserved

Table of DownloadStart Data Fields

Operations Sent: *MethodStart*

DownloadStart MethodStart (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Padding	BCC
0x02	0x80	0x10	N	4	Board Net ID	Memory Type	Memory Unit	Flags	22 bytes 0x00	0xyy

Operations Returned: *AckNak*

DownloadStart AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x02	0x07	0x10	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.3.2 DownloadSegment (Class 0x02, Function 0x11)

DownloadSegment facilitates scatter-load. Linked output is not always located in contiguous memory. DowloadStart determined the memory space of image. This function via the Action argument provides a sentinel to signal that transfer of data to a previous segment has been completed. Furthermore, it supports initiation of transfer data to a new segment with the optional arguments Segment Type, Address / ID and Size similar to DownloadStart. Because an Ack (AckNak) message is not mandatory Flags allow for an option Acknowledge.
If "Weight on Wheels" is not active or a download is not active or if implemented a diagnostic session in programming mode is not active, a NAK message with error code is sent.

Data Fields

Name	Data Type	Byte Size	Range	Description
Board Network ID	Enum	1	0 – 1	0: Main, 1: PA
Segment Type	Enum	1	 0x00 0x01 0x02 0x03 0x04 0x10 0x11	Segment Data Type Byte, 8-Bit data, U8 Short 16-Bit Data, U16 Long 32-Bit Data, U32 DSP Program 48-Bit Data, U48 Longlong 64-Bit Data, U64 Tuning DB Header (Byte, U8) Tuning DB Record (Byte, U8)
Segment Address or ID	U32	4	Segment Dependent	Address if a N-Bit Segment Type; Tuning DB Header NA but Record the ID.
Segment Size	U32	4	Segment Dependent	Size in BYTES not segment type units used for addressing, alignment & transfer size check
Flags	Bitfield	1	 Bit 0 Bit 1 Bit 2 Bit 3 Bits 4 – 7	Flags 0: ignore 1: Finish previous segment 0: ignore 1: Start new segment 0: ignore 1: send Ack 0: ignore 1: reserved for erase segment Reserved

Table of DownloadSegment Data Fields

Operations Sent: *MethodAction*

DownloadSegment MethodAction (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4
0x02	0x82	0x11	N	11	Board Net ID	Segment Type	Seg Addr / ID MSB	Seg Addr / ID MMSB

Data 5	Data 6	Data 7	Data 8	Data 9	Data 10
Seg Addr / ID LLSB	Seg Addr / ID LSB	Seg Size MSB	Seg Size MMSB	Seg Size LLSB	Seg Size LSB

Data 11	Padding	BCC
Flags	15 bytes x 0x00	0xyy

Operations Returned: *AckNak*

DownloadSegment AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x02	0x07	0x11	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.3.3 DownloadEnd (Class 0x02, Function 0x12)

DownloadEnd is sent to finish the download process. If a DownloadSegment had not been sent to finish the last segment, DownloadEnd automatically finishes, cleans up the segment. Flags allow for validation of the download before AckNak (highly recommended) and reset after transmission of the AckNak.

If "Weight on Wheels" is not active or a download is not active or if implemented a diagnostic session in programming mode is not active, a NAK message with error code "Functioned Not Executed" is sent.

The sender of this message must wait for a response, an AckNak message. This may take seconds. If the download is invalid the AckNak byte error will be a CRC-32 error.

After the DownloadEnd is received, the amplifier searches for the first valid record in the database and sets the ActiveConfigDatabase to that record. Then the unit is reset such that the newest load and changes take effect.

Data Fields

Name	Data Type	Byte Size	Range	Description
Board Network ID	Enum	1	0 – 1	0: Main, 1: PA
Flags	Bitfield	1	 Bit 0 Bit 1 Bits 2 – 7	Flags 0: ignore 1: Validate download before AckNak 0: ignore 1: Reset after transmission of Ack Reserved

Table of DownloadEnd Data Fields

Operations Sent: ***MethodStop***

DownloadEnd MethodStop (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x02	0x84	0x12	N	2	Board Net ID	Flags	24 bytes x 0x00	0xyy

Operations Returned: ***AckNak or better MethodResult***

DownloadEnd AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x02	0x07	0x12	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.3.4 DownloadAbort (Class 0x02, Function 0x13)

DownloadAbort causes an immediate halt to the download process. This could eventually result in an unstable software operational state. It is strongly suggested that the "Reset after sending AckNak" is set.

If "Weight on Wheels" is not active or a download is not active or if implemented a diagnostic session in programming mode is not active, a NAK message with error code "Functioned Not Executed) is sent.

The sender of this message must wait for a response, an AckNak message.

Data Fields

Name	Data Type	Byte Size	Range	Description
Board Network ID	Enum	1	0 – 1	0: Main, 1: PA
Flags	Bitfield	1	 Bit 0 Bits 1 – 7	Flags 0: ignore 1: Reset after transmission of AckNak Reserved

Table of DownloadAbort Data Fields

Operations Sent: ***MethodStop***

DownloadAbort MethodStop (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x02	0x84	0x13	N	2	Board Net ID	Flags	24 bytes x 0x00	0xyy

Operations Returned: ***AckNak or better MethodResult***

DownloadAbort AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x02	0x07	0x13	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.3.5 TransferData (Class 0x02, Function 0x30)

The direction of TransferData is determined by whether a download or upload session is in process.

If “Weight on Wheels” is not active or a download or upload is not active or if implemented a diagnostic session in programming mode is not active, a NAK message with error code is sent.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Network ID	Enum	1	0 – 1	0: Main, 1: PA
Data	Byte	1 - 26	0 – 0xFF	Data

Operations Sent: *MethodAction*

TransferData MethodAction (Database Header) to DA780 Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2-26	Padding	BCC
0x02	0x82	0x30	N	2 - 26	Board Net ID	Data	(25 – length) bytes 0x00	0xyy

Operation returned: AckNak

TransferData AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x02	0x07	0x30	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.3.6 TransferStatus (Class 0x02, Function 0x31)

TransferStatus allows the examination of whether a session is active and the status and state of the currently or previously active session.
If "Weight on Wheels" is not active or a download or if implemented a diagnostic session in programming mode is not active, a NAK message with error code "Functioned Not Executed" is sent.
Note that the segment address and data remaining (size) fields have different meaning than in DownloadSegment.

Data Fields

Name	Data Type	Byte Size	Range	Description
Board Network ID	Enum	1	0 – 1	0: Main, 1: PA
Status	Bitfield	1	 Bit 0 Bit 1 Bit 2 Bit 3	Active? Okay? Direction? Erasing 0:Not Active 1: Active 0: Not Okay 1: Okay 0: Download 1:Upload 0: Not Erasing 1:Erasing
State	Enum	1	 0 1 2 3 4	 Done Starting Stopping (Verifying, Aborting &c) Paused Transferring Data (includes TransferData and DownloadSegment)
Board Network ID		1		See DownloadStart
Memory Type		1		See DownloadStart
Memory Unit		1		See DownloadStart
Segment Type		1		See DownloadStart
Segment Address or ID	U32	4		Current not start address like in DownloadStart – in case of ID requested and current are same.
Segment Data Remaining	U32	4		The size in BYTES remaining not total size like in DownloadStart

Table of TransferStatus Data Fields

Operations Sent: ***Get***

TransferStatus Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x02	0x01	0x31	N	0	26 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

TransferStatus Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Data 6	Data 7
0x02	0x06	0x31	N	14	Board Net ID	Status	State	Board Net ID	Memory Type	Memory Unit	Segment Type

Data 8	Data 9	Data 10	Data 11	Data 12	Data 13	Data 14
Seg Addr / ID MSB	Seg Addr / ID MMSB	Seg Addr / ID LLSB	Seg Addr / ID LSB	Seg Remain MSB	Seg Remain MMSB	Seg Remain L LSB

Data 15	Padding	BCC
Seg Remain. LLSB	11 bytes x 0x00	0xyy

TransferStatus AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x02	0x07	0x31	N	1	Ack Nak	25 bytes x 0x00	0xyy

4.4.4 AircraftInformation Class 0x03 Functions

Func	Name
0x0B	Weight-On-Wheels
0x11	AircraftInfo

4.4.4.1 AircraftInfo (Class 0x03, Function 0x11)

The AircraftInfo Class provides general aircraft information like engine speed, air speed, Weight-on-wheels and other parameters that may be useful to noise adaptation and other audio control algorithms. These values must be written to the Client from the Host. If a PA device is present, then the Weight On Wheels signal should be read from the PA Event command.

Name	Data Type	Size in Bytes	Range	Description
Altitude	Byte	2	0 – 65535	Current Altitude in Feet
Air Speed	Byte	2	0 – 65535	Current Air Speed in MPH
Weight On Wheels	Byte	1	0-1	Indicates if aircraft is on ground. Value 1 = on ground, 0 = in flight. Will get overwritten by PA if present by next PA Event.

Table of AircraftInfo Data Fields

Operations Sent: ***Get, Set***

AircraftInfo Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x03	0x01	0x11	N	0	26 bytes x 0x00	0xyy

AircraftInfo Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Padding	BCC
0x03	0x00	0x11	N	5	Altitude (MSB)	Altitude (LSB)	Air Speed (MSB)	Air Speed (LSB)	Weight On Wheels	21 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

AircraftInfo Data Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Padding	BCC
0x03	0x06	0x11	N	5	Altitude (MSB)	Altitude (LSB)	Air Speed (MSB)	Air Speed (LSB)	Weight On Wheels	21 bytes x 0x00	0xyy

AircraftInfo AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x03	0x07	0x11	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.4.2 WOWOverride (Class 0x03, Function 0x0B)

The WOWOverride Class provides the option to control the WOW status via software, thereby overriding the actual pin state if desired.

Name	Data Type	Size in Bytes	Range	Description
Override-Enable	Byte	1	0 – 1	Enables the WOW override (0=No-Override, 1=Override)
Override-State	Byte	1	0 – 1	Determines the state of the override (0=NO-WOW, 1=WOW)

Table of WOWOverride Data Fields

Operations Sent: ***Get, Set***

WOWOverride Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x03	0x01	0x0B	N	0	26 bytes x 0x00	0xyy

WOWOverride Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x03	0x00	0x0B	N	2	Override-Enable	Override-State	24 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

WOWOverride Data Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x03	0x06	0x0B	N	2	Override-Enable	Override-State	24 bytes x 0x00	0xyy

WOWOverride AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x03	0x07	0x0B	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.5 Analog Class 0x04 Functions

The Analog Class provides an interface to monitor and manage analog device controls (where applicable) on the Main amplifier.

Function Token	FunctionName
0x01	Select AB_1
0x02	Select AB_2
0x03	Select Diag

4.4.5.1 Analog Select AB_1 (Class 0x04, Function 0x01)

This function is used to set the state of the analog switches on the Main board for input 1. Either A or B may be selected.

Name	Data Type	Size in Bytes	Range	Description
AB	Enum	1	0 - 1	0: A, 1: B Selects A or B for input zone 1

Operations Sent: ***Get, Set***

SelectAB_12 Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x04	0x01	0x01	N	0	26 bytes x 0x00	0xyy

SelectAB_12 Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x04	0x00	0x01	N	1	AB	25 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

SelectAB_12 Data Response from Get (Client to Host):

Class	Op	Func	Seq	Length	Data 1	Padding	BCC
0x04	0x06	0x01	N	1	AB	25 bytes x 0x00	0xyy

SelectAB_12 AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x04	0x07	0x01	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.5.2 Analog Select AB_2 (Class 0x04, Function 0x02)

This function is used to set the state of the analog switches for input 2 on the Main board. Either A or B may be selected.

Name	Data Type	Size in Bytes	Range	Description
AB	Enum	1	0 - 1	0: A, 1: B Selects A or B for input zone 2

Operations Sent: ***Get, Set***

SelectAB_12 Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x04	0x01	0x02	N	0	26 bytes x 0x00	0xyy

SelectAB_12 Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x04	0x00	0x02	N	1	AB	25 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

SelectAB_12 Response from Get (Client to Host):

Class	Op	Func	Seq	Length	Data 1	Padding	BCC
0x04	0x06	0x02	N	1	AB	25 bytes x 0x00	0xyy

SelectAB_12 AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x04	0x07	0x02	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.5.3 Analog Select Diag (Class 0x04, Function 0x03)

This function is used to set the state of the analog switch which routes the Diagnostic analog input (via the diagnostic connector) on the Main board to channel 4 for diagnostic purposes.

Name	Data Type	Size in Bytes	Range	Description
Diag	Enum	1	0 - 1	0: Normal, 1: Diag Selects Diag input

Operations Sent: ***Get, Set***

SelectDiag Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x04	0x01	0x03	N	0	26 bytes x 0x00	0xyy

SelectDiag Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x04	0x00	0x03	N	1	Diag	25 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

SelectDiag Response from Get (Client to Host):

Class	Op	Func	Seq	Length	Data 1	Padding	BCC
0x04	0x06	0x03	N	1	Diag	25 bytes x 0x00	0xyy

SelectDiag AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x04	0x07	0x03	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.6 Headphone Class 0x07 Functions

Func	Name
0x09	HPInputSelect
0x11	HPVolume
0x14	HPMute

4.4.6.1 HPInputSelect (Class 0x07, Function 0x09)

HPInput Select chooses the input source on the headphone inputs. The amplifier may have an input active mask and if a non-active input is selected then an error will be returned. Inc and Dec wrap around to any active inputs.

Name	Data Type	Size in Bytes	Range	Description
Headphone#	Byte	1	0 – 6	Selects which headphone output (1-6). 0 is a global broadcast to all headsets
Input	Byte	1	1 - 4	Selects the input

Table of HPInputSelect Data Fields

Operations Sent: ***Get, Set, Inc, Dec***

HPInputSelect Get (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x07	0x01	0x09	N	1	Headphone#	25 bytes x 0x00	0xyy

HPInputSelect Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x07	0x00	0x09	N	2	Headphone#	Input	24 bytes x 0x00	0xyy

HPInputSelect Inc (Host to Client):

Class	Oper	Func	Seq	Length	Data1	Padding	BCC
0x07	0x03	0x09	N	1	Headphone#	25 bytes x 0x00	0xyy

HPInputSelect Dec (Host to Client):

Class	Oper	Func	Seq	Length	Data1	Padding	BCC
0x07	0x04	0x09	N	1	Headphone#	25 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

HPInputSelect Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x07	0x06	0x09	N	2	Headphone#	Selected Input	24 bytes x 0x00	0xyy

HPInputSelect AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x07	0x07	0x09	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.6.2 HPVolume (Class 0x07, Function 0x11)

HPVolume affects the volume for one or all headphone outputs. Volume is a 32 position control which uses a pre-determined taper.

Name	Data Type	Size in Bytes	Range	Description
Headphone#	Byte	1	0 – 6	Select the headphone output (1-6) or 0 for global broadcast to all
Volume	Byte	1	0 - 31	0: mute, 31: maximum volume

Table of HPVolume Control Data Fields

Operations Sent: ***Get, Set, Inc, Dec***

HPVolume Get (Host to Client):

Class	Oper	Func	Seq	Length	Data1	Padding	BCC
0x07	0x01	0x11	N	1	Headphone#	25 bytes x 0x00	0xyy

HPVolume Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x07	0x00	0x11	N	2	Headphone#	Volume	24 bytes x 0x00	0xyy

HPVolume Inc (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x07	0x03	0x11	N	1	Headphone#	25 bytes x 0x00	0xyy

HPVolume Dec (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x07	0x04	0x11	N	1	Headphone#	25 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

HPVolume Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x07	0x06	0x11	N	2	Headphone#	Volume	24 bytes x 0x00	0xyy

HPVolume AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x07	0x07	0x11	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.6.3 HPMute (Class 0x07, Function 0x14)

HPMute affects the mute for one or all headphone outputs.

Name	Data Type	Size in Bytes	Range	Description
Headphone#	Byte	1	0 – 6	Select the headphone output (1-6) or 0 for global broadcast to all
Mute	Byte	1	0 - 1	0: unmute, 1: mute

Table of HPMute Control Data Fields

Operations Sent: ***Get, Set, Inc, Dec***

HPVolume Get (Host to Client):

Class	Oper	Func	Seq	Length	Data1	Padding	BCC
0x07	0x01	0x14	N	1	Headphone#	25 bytes x 0x00	0xyy

HPVolume Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x07	0x00	0x14	N	2	Headphone#	Mute	24 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

HPVolume Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x07	0x06	0x14	N	2	Headphone#	Volume	24 bytes x 0x00	0xyy

HPVolume AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x07	0x07	0x14	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.7 Amplifier Class 0x08 Functions

Func	Name
0x03	Diag Input Select
0x08	AudioFormat
0x09	InputSelect
0x11	Volume
0x12	Bass
0x13	Treble
0x14	Mute
0x20	Compressor
0x21	Loudness
0x22	Spatial
0x30	SurroundEnable

4.4.7.1 Diag Input Select (Class 0x08, Function 0x03)

This function is used to set the state of the analog switch which routes the Diagnostic analog input (via the diagnostic connector) on the 4X6 Combo Amp to input channel 4 for diagnostic purposes.

Name	Data Type	Size in Bytes	Range	Description
Diag	Enum	1	0 - 1	0: Normal, 1: Diag Selects Diag input

Operations Sent: ***Get, Set***

SelectDiag Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x08	0x01	0x03	N	0	26 bytes x 0x00	0xyy

SelectDiag Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x00	0x03	N	1	Diag	25 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

SelectDiag Response from Get (Client to Host):

Class	Op	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x06	0x03	N	1	Diag	25 bytes x 0x00	0xyy

SelectDiag AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x07	0x03	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.7.2 AudioFormat (Class 0x08, Function 0x08)

AudioFormat is used to specify the audio data format (stereo or 5.1) for one or more zones. A bit field selects the zones in which the audio format is changed. The audio routing is determined in the DSP image and is not remotely configurable.

Name	Data Type	Size in Bytes	Range	Description
Zone Select	Bit field	1	Bit 0: Zone 1: 0 / 1 Bit 1: Zone 2: 0 / 1	Set the zone bit(s) affected by the audio format
Zone N Format	Bit field	1	Bit 0: 0 / 1 Bit 1 - 7: 0	0:stereo, 1: 5.1 PCM 2.0, Reserved for other formats, N = Zone

Table of AudioFormat Data Fields

Operations Sent: ***Get, Set***

AudioFormat Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x08	0x01	0x08	N	0	26 bytes x 0x00	0xyy

AudioFormat Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x08	0x00	0x08	N	3	Zone Select	Zone 1 Format	Zone 2 Format	23 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

AudioFormat Data Response from Get (Client to Host):

Class	Op	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x08	0x06	0x08	N	2	Zone 1 Format	Zone 2 Format	24 bytes x 0x00	0xyy

AudioFormat AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x07	0x08	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.7.3 InputSelect (Class 0x08, Function 0x09)

Input Select chooses the input source on amplifier with this function implemented for one or more zones. A bit field is set to determine the zones in which the selection is made. The amplifier may have an input active mask and if a non-active input is selected then an error will be returned. Inc and Dec wrap around to active inputs.

Name	Data Type	Size in Bytes	Range	Description
Zone Select	Bit field	1	Bit 0: Zone 1: 0 / 1 Bit 1: Zone 2: 0 / 1	Set the zone bit(s) affected by the select
Input	Byte	1	1 - 4	Selects the input

Table of InputSelect Data Fields

Operations Sent: ***Get, Set, Inc, Dec***

InputSelect Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x08	0x01	0x09	N	0	26 bytes x 0x00	0xyy

InputSelect Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x08	0x00	0x09	N	3	Zone Select	Input Z1	Input Z2	23 bytes x 0x00	0xyy

InputSelect Inc (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x03	0x09	N	1	Zone Select	25 bytes x 0x00	0xyy

InputSelect Dec (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x04	0x09	N	1	Zone Select	25 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

InputSelect Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x08	0x06	0x09	N	2	Selected Input Z1	Selected Input Z2	24 bytes x 0x00	0xyy

InputSelect AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x07	0x09	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.7.4 Volume (Class 0x08, Function 0x11)

Volume affects the entertainment volume for one or more zones. A bit field is set to determine the zones in which the volume is changed. Volume is a 32 position control which uses a pre-determined taper.

Name	Data Type	Size in Bytes	Range	Description
Zone Select	Bit field	1	Bit 0: Zone 1: 0 / 1 Bit 1: Zone 2: 0 / 1	Set the zone bit(s) affected by the volume
Zone N Volume	Byte	1	0 - 31	0: mute, 31: maximum volume, N = Zone

Table of Volume Control Data Fields

Operations Sent: ***Get, Set, Inc, Dec***

Volume Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x08	0x01	0x11	N	0	26 bytes x 0x00	0xyy

Volume Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x08	0x00	0x11	N	3	Zone Select	Zone 1 Volume	Zone 2 Volume	23 bytes x 0x00	0xyy

Volume Inc (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x03	0x11	N	1	Zone Select	25 bytes x 0x00	0xyy

Volume Dec (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x04	0x11	N	1	Zone Select	25 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

Volume Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x08	0x06	0x11	N	2	Zone 1 Volume	Zone 2 Volume	24 bytes x 0x00	0xyy

Volume AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x07	0x11	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.7.5 Bass (Class 0x08, Function 0x12)

Bass affects the entertainment bass for one or more zones. A bit field is set to determine the zones in which the bass is changed. Bass is a 15 position control. Center position is 0, maximum cut -7, maximum boost +7. Cut or boost for each position is stored in a database using a pre-determined taper.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Zone Select	Bit field	1	Bit 0: Zone 1: 0 / 1 Bit 1: Zone 2: 0 / 1	Set the zone bit(s) affected by the bass
Zone N Bass	Signed	1	-7 to +7	0: center, -7: max cut, +7: max boost, N= Zone

Table of Bass Control Data Fields

Operations Sent: ***Get, Set***

Bass Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x08	0x01	0x12	N	0	26 bytes x 0x00	0xyy

Bass Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x08	0x00	0x12	N	3	Zone Select	Zone 1 Bass	Zone 2 Bass	23 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

Bass Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x08	0x06	0x12	N	2	Zone 1 Bass	Zone 2 Bass	24 bytes x 0x00	0xyy

Bass AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x07	0x12	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.7.6 Treble (Class 0x08, Function 0x13)

Treble affects the entertainment treble for one or more zones. A bit field is set to determine the zones in which the treble is changed. Treble is a 15 position control. Center position is 0, maximum cut -7, maximum boost +7. Cut or boost for each position is stored in a database using a pre-determined taper.

Name	Data Type	Size in Bytes	Range	Description
Zone Select	Bit field	1	Bit 0: Zone 1: 0 / 1 Bit 1: Zone 2: 0 / 1	Set the zone bit(s) affected by the treble
Zone N Treble	Signed	1	-7 to +7	0: center, -7: max cut, +7: max boost, N = Zone

Table of Treble Control Data Fields

Operations Sent: ***Get, Set***

Treble Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x08	0x01	0x13	N	0	26 bytes x 0x00	0xyy

Treble Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x08	0x00	0x13	N	3	Zone Select	Zone 1 Treble	Zone 2 Treble	23 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

Treble Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x08	0x06	0x13	N	2	Zone 1 Treble	Zone 2 Treble	24 bytes x 0x00	0xyy

Treble AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x07	0x13	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.7.7 Mute (Class 0x08, Function 0x14)

Mute attenuates entertainment audio completely (with a soft envelope so there is no pop sound) for one or more zones. A bit field is set to determine the zones in which mute takes effect.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Zone Select	Bit field	1	Bit 0: Zone 1: 0 / 1 Bit 1: Zone 2: 0 / 1	Set the zone bit(s) affected by the Mute
Zone N Mute	Enum	1	Bit 0: 0/1	0:demute, 1:mute

Table of Mute Data Fields

Operations Sent: ***Get, Set***

Mute Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x08	0x01	0x14	N	0	26 bytes x 0x00	0xyy

Mute Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x08	0x00	0x14	N	3	Zone Select	Zone 1 Mute	Zone 2 Mute	23 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

Mute Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x08	0x06	0x14	N	2	Zone 1 Mute	Zone 2 Mute	24 bytes x 0x00	0xyy

Mute AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x07	0x14	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.7.8 Compressor (Class 0x08, Function 0x20)

Compressor settings affect entertainment audio in one or more zones. Enabling the compressor limits the dynamic range of the incoming signal allowing loud and quiet passages to achieve a better balance in a noisy environment. A bit field is set to determine the zones in which the compression is enabled.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Zone Select	Bit field	1	Bit 0: Zone 1: 0 / 1 Bit 1: Zone 2: 0 / 1	Set the zone bit(s) affected by the compressor
Zone N Compressor	Byte	1	0 = Disabled 1 = Enabled	Set Enable or Disable

Table of Compressor Data Fields

Operations Sent: ***Get, Set***

Compressor Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x08	0x01	0x20	N	0	26 bytes x 0x00	0xyy

Compressor Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x08	0x00	0x20	N	3	Zone Select	Zone 1 setting	Zone 2 setting	23 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

Compressor Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x08	0x06	0x20	N	2	Zone 1 setting	Zone 2 setting	24 bytes x 0x00	0xyy

Compressor AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x07	0x20	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.7.9 Loudness (Class 0x08, Function 0x21)

Loudness settings affect entertainment audio in one or more zones based upon the Fletcher-Munson curves. A bit field is set to determine the zones in which the loudness settings are applied. Both Bass and Treble depth can be set independently.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Zone Select	Bit field	1	Bit 0: Zone 1: 0 / 1 Bit 1: Zone 2: 0 / 1	Set the zone bit(s)
ZoneN Boost	Bit Mask	1	Bit 0:1 – Bass boost 00 = None 01 = 6 dB 10 = 9 dB 11 = 12 dB Bit 2:3 – Treb boost 00 = None 01 = 4 dB 10 = 6 dB 11 = 8 dB	Set Max amount of boost at low volume

Table of Loudness Data Fields

Operations Sent: ***Get, Set***

Loudness Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x08	0x01	0x21	N	0	26 bytes x 0x00	0xyy

Loudness Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x08	0x00	0x21	N	3	Zone Select	Zone 1 setting	Zone 2 setting	23 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

Response:

Loudness Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x08	0x06	0x21	N	2	Zone 1 setting	Zone 2 setting	24 bytes x 0x00	0xyy

Loudness AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x07	0x21	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.7.10 Spatial (Class 0x08, Function 0x22)

Spatial is a spatial enhancement audio processing algorithm designed to improve the soundstage. A bit field selects in which zones the spatial enhancement is enabled.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Zone Select	Bit field	1	Bit 0: Zone 1: 0 / 1 Bit 1: Zone 2: 0 / 1	Set the zone bit(s) affected by the spatial
Zone N Setting	Byte	1	0 = Disabled 1 = Intensity min 2 = Intensity med 3 = Intensity max	Set Intensity of Spatial enhancement

Table of Spatial Data Fields

Operations Sent: ***Get, Set***

Spatial Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x08	0x01	0x22	N	0	26 bytes x 0x00	0xyy

Spatial Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x08	0x00	0x22	N	3	Zone Select	Zone 1 setting	Zone 2 setting	23 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

Spatial Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x08	0x06	0x22	N	2	Zone 1 setting	Zone 2 setting	24 bytes x 0x00	0xyy

Spatial AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x07	0x22	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.7.11 SurroundEnable (Class 0x08, Function 0x30)

SurroundEnable allows control to disable the output to the surround channels (Center, Surround L&R). A bit field selects in which zones the command acts upon. If the surround is to be disabled, it is recommended to set the AudioFormat to Stereo to insure that no audio information is lost.

Data Fields

Name	Data Type	Size in Bytes	Range	Description
Zone Select	Bit field	1	Bit 0: Zone 1: 0 / 1 Bit 1: Zone 2: 0 / 1	Set the zone bit(s) affected by the Surround Enable
Zone N Setting	Byte	1	0 = Enabled 1 = Center Muted 2 = Surrounds Muted 3 = Center & Surrounds Muted	Set the Channel Enable parameters

Table of SurroundEnable Data Fields

Operations Sent: ***Get, Set***

SurroundEnable Get (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x08	0x01	0x30	N	0	26 bytes x 0x00	0xyy

SurroundEnable Set (Host to Client):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Data 3	Padding	BCC
0x08	0x00	0x30	N	3	Zone Select	Zone 1 setting	Zone 2 setting	23 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

SurroundEnable Response from Get (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x08	0x06	0x30	N	2	Zone 1 setting	Zone 2 setting	24 bytes x 0x00	0xyy

SurroundEnable AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x08	0x07	0x30	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.8 PAChime Class 0x09 Functions

Func	Name
0x10	SetOutputChannels
0x11	ChimeAudioSequence
0x12	PAEvent

4.4.8.1 SetOutputChannels (Class 0x09, Function 0x10)

SetOutputChannels prepares the PA for a software ChimeAudioSequence by specifying a channel output mask for all PA outputs. Settings will be stored for the current session until command is re-issues or unit is power cycled.

Name	Data Type	Size in Bytes	Range	Description
OutputMute	Bit field	1	Bit 0: Speakers1&2: 0/1 Bit 1: Speakers 3&4: 0/1 Bit 2: Speaker Aux1: 0/1 Bit 3: Speaker Aux2: 0/1 Bit 4: Line L : 0/1 Bit 5: Line R: 0/1 Bit 6: Chime1: 0/1 Bit 7: Chime2: 0/1	0:ON, 1:MUTE. Set the outputs for the audio / chime event output(s) ON or MUTED
SidetoneMute	Bit field	1	Bit 0: Sidetone 1: 0/1 Bit 1: Sidetone 2: 0/1 Bit 2: Sidetone 3: 0/1 Bit 3: Relay Inhibit 0/1 Bit 4: PA Event Inhibit 0/1 Bit 5: PA Message Inhibit 0/1 Bit 6: PA Active Inhibit 0/1	0:ON, 1:MUTE. Set the sidetone outputs for the audio / chime stream ON or MUTED Relay Inhibit prevents the Speaker relays on the PA board from switching. PA Event Inhibit prevents the discrete line to the main amp from triggering PA Message Inhibit prevents the PA Event message from being generated PA Active Inhibit prevents the PA Active output on the connector from activating.

Table of PA SetOutputChannels Data Fields

Operations Sent: ***Get, Set***

SetOutputChannels Get (Host to Client)

Class	Oper	Func	Seq	Length	Padding	BCC
0x09	0x01	0x10	N	0	26 bytes x 0x00	0xyy

SetOutputChannels Set (Host to Client

Class	Op	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x09	0x00	0x10	N	2	Output Mute	Sidetone Mute	24 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

SetOutputChannels Response (Client to Host)

Class	Oper	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x09	0x06	0x10	N	2	Output Mute	Sidetone Mute	24 bytes x 0x00	0xyy

SetOutputChannels AckNak (Client to Host)

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x09	0x07	0x10	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.8.2 ChimeAudioSequence (Class 0x09, Function 0x11)

ChimeAudioSequence is used to generate chimes and stream announcement audio using a network software call. All other announcements are triggered by hard-wired ordinance.
Sequences initiated through this command take the lowest priority, beneath all hard-wired ordinances or announcements.
Be sure to close an audio channel once it is opened.

A cue of 3 commands will be stored and executed when the PA is free. It is suggested that a Get PAEvent should be executed first to check if the PA is free and determine the status of events. There is no Get for this command.
If interrupted by a higher priority event, any open audio channel will be re-opened after the higher priorities clear.

Name	Data Type	Size in Bytes	Range	Description
Command	Byte	1	0 - 2	0: Audio Close, 1: Audio Open 2: Do Chime
Sound	Byte	1	1 - 6	1: Tone1, 2: Tone2, 3: Tone3, 4:Tone4, 5: Tone5, 6: Tone6,
Volume	Signed	1	-40 - +20	Gain in dB for Audio channel

Table of ChimeAudioSequence Data Fields

Operations Sent: ***Set***

ChimeAudioSequence Set (Host to Client): Initiate a Chime

Class	Op	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x09	0x00	0x11	N	2	DoChime	Tone	24 bytes x 0x00	0xyy

ChimeAudioSequence Set (Host to Client): Open the Audio Channel

Class	Op	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x09	0x00	0x11	N	2	AudioOpen	Volume	24 bytes x 0x00	0xyy

ChimeAudioSequence Set (Host to Client): Close the Audio Channel

Class	Op	Func	Seq	Length	Data 1	Data 2	Padding	BCC
0x09	0x00	0x11	N	2	AudioClose	Volume	24 bytes x 0x00	0xyy

Operations Returned: ***Response, AckNak***

ChimeAudioSequence AckNak (Client to Host):

Class	Oper	Func	Seq	Length	Data 1	Padding	BCC
0x09	0x07	0x11	N	1	AckNak	25 bytes x 0x00	0xyy

4.4.8.3 PA Event (Class 0x09, Function 0x12)

PA Event is an unsolicited message that is sent to the Host whenever a MIC or Ordinance event occurs on the PA Board. Host can also request the current status.

Name	Data Type	Size in Bytes	Range / Value	Description
Event	Byte	1	1 = Software Chime 2 = Software Briefer 3 = Aux2 PTT 4 = Aux1 PTT 5 = Mic 3 PTT 6 = Mic 2 PTT 7 = Mic 1 PTT 8 = Ordinance 6 9 = Ordinance 5 10 = Ordinance 4 11 = Ordinance 3 12 = Ordinance 2 13 = Ordinance 1	Indicates which event is being reported
Event State	Boolean	1	0 - 1	0: Not-Active, 1: Active
Weight On Wheels	Byte	1	0 - 1	0:OFF, 1:ON Set to 1 when WOW active Set to 0 when WOW ends
KLI Status1	Bit field	1	Bit 0 = Ordinance 1 Bit 1 = Ordinance 2 Bit 2 = Ordinance 3 Bit 3 = Ordinance 4 Bit 4 = Ordinance 5 Bit 5 = Ordinance 6 Bit 6 = Not Used Bit 7 = Not Used	each bit represents status of corresponding Key Line input (or virtual SW KLI) Bit = 1: KLI set Bit = 0: KLI not set
KLI Status2	Bit field	1	Bit 0 = Mic 1 PTT Bit 1 = Mic 2 PTT Bit 2 = Mic 3 PTT Bit 3 = Aux1 PTT Bit 4 = Aux2 PTT Bit 5 = Software Briefer Bit 6 = Software Chime Bit 7 = Not Used	each bit represents status of corresponding Key Line input (or virtual SW KLI) Bit = 1: KLI set Bit = 0: KLI not set

Operations Sent: ***Unsolicited, Get***

PAEvent Unsolicited (Client to Host):

Class	Op	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Padding	BCC
0x09	0x05	0x12	N	5	Event	Event State	WOW	KLI Status1	KLI Status2	21 bytes x 0x00	0xyy

PAEvent Get (Host to Client): Request status of current event (if any)

Class	Oper	Func	Seq	Length	Padding	BCC
0x09	0x01	0x12	N	0	26 bytes x 0x00	0xyy

Operations Returned: ***Response***

PAEvent Response (Client to Host):

Class	Op	Func	Seq	Length	Data 1	Data 2	Data 3	Data 4	Data 5	Padding	BCC
0x09	0x06	0x12	N	5	Event	Event State	WOW	KLI Status1	KLI Status2	21 bytes x 0x00	0xyy

5 Appendix

5.1 AltoNET™ Interface Details

The AltoNET™ protocol functions over the RS-422 interface on the Main and PA Boards. The intent of AltoNET™ is to offer a proprietary interface for inter-board communication. This interface allows for probing board health, controlling audio parameters in real-time, and uploading and downloading files to the board.

5.1.1 Physical Layer Details

The RS422 interface is differential with up to -10 to +10 Volt input range.
Differential signal should be at least +/- 2V fully loaded.
The protocol supports full duplex and expects a baud rate of 115,200, 8 bits of data, no parity, 1 stop bit and no flow control.

The physical connections required implementing this interface as Inputs and Outputs are:

Signal	PA700 BD	Main BD	Description
Tx+	J5-27	J2-9	Transmit +
Tx-	J5-26	J2-8	Transmit -
Rx+	J5-25	J2-7	Receive +
Rx-	J5-24	J2-6	Receive -

5.1.2 Message Format

All messages set to either Main or PA boards via AltoNET will have the same basic format shown below:

	Prefix	Class	Operation	Function	Sequence	Length	Data	BCC	CR/LF
Actual bits	16 bits	8 bits	8 bits	8 bits	8 bits	8 bits	<Data 1>…<Data N>	8 bits	16 bits
ASCII Xmitted bits	32 bits	16 bits	16 bits	16 bits	16 bits	16 bits	<Data 1>…<Data N>	16 bits	16 bits

- The messages are sent and received as an ASCII string, so ascii “AA550008” is actually sent and received as 0x41 0x41 0x35 0x35 0x30 0x30 0x30 0x38 ….
- ASCII string must be terminated with CR/LF (0x0D 0x0A)
- The prefix is 0xAA 0x55
- The next Five bytes form the header.
- Unused bytes beyond the last byte of valid data are functionally ignored.
- BCC is calculated as a running XOR of each HEX byte in the message before conversion to ASCII, excluding the prefix (32 bytes).
- Message header, data length, and BCC are checked for validity in returned ACK/NACK.
- Each Data Field is 8 bits unless otherwise indicated throughout the document.
- There must be a minimum delay between sent messages of 20mS.

5.2 AltoSPI Interface Details

5.2.1 AltoSPI Physical Layer

The physical connections required implementing this interface as Inputs and Outputs are:

Pin	Amp	Host	Description
SPI_ENET_SS*	I	O	Chip Select
ENET_MUTE*	I	O	Mute
SPI_ENET_REQUEST*	O	I	Service
SPI_ENET_MOSI	I	O	Master Out Slave In
SPI_ENET_MISO	O	I	Master In Slave Out
SPI_ENET_SCK	I	O	Serial Clock

The AltoSPI Control Channel communications interface is a standard four wire SPI. It uses pins SPI_ENET_SCK, SPI_ENET_MOSI, SPI_ENET_MISO and SPI_ENET_SS*.

The SPI_ENET_REQUEST* is a special purposes pin used by the Data Link Layer to manage a Half-Duplex interface.

The ENET_MUTE* pin is also a special purpose used to instruct the application to mute immediately.

Please reference DOC 106900PS for more detailed electrical specifications.

5.2.2 AltoSPI Data Link Layer

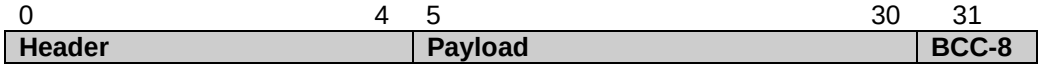
- The SPI Host Daughtercard is Master of the SPI and the DA-700 Series Amplifier is the Slave.
- Exchange of Data uses SPI Mode 0 which means the SCK idles LOW and data on MOSI and MISO are valid upon SCK transition from LOW to HIGH
- Data are octets, 8-bits so there are 8 SCK per character.
- The BAUD is 3 MHz.
- Data transfers begin when the master asserts CS LOW and end when the master de-asserts CS HIGH.
- The AltoSPI interface uses a 32 byte word fixed length message. All the unused data bytes are set to zero.
- In order to insure Framing Synchronization, all 32 bytes of the SPI exchange must be executed during one CS assertion cycle. If CS is released during an exchange the buffer is reset.
- The Alto Amp (Slave) asserts LOW SPI_ENET_REQUEST* if it needs service, that is, if it has a message to transmit. The slave will not assert SPI_ENET_REQUEST* while a SPI exchange is active.
- It is possible that the Master begins its own transmission immediately after the Slave asserts the SPI_ENET_REQUEST*. In this case, both transmissions will be simultaneous. When Master is sending a message, Slave is receiving and exchanging data with the same clocks. Master shall check for the exchanged received data at all time.
- If Slave has an unsolicited message, it will initiate a SPI_ENET_REQUEST* and Master shall generate clock after asserting the CS and start data exchange. Master shall send an Acknowledge to Slave after the completion of data exchange.
- All SPI events are a Data Exchange, therefore the receive buffer should be evaluated after every transfer event to check for a valid message. This is necessary in case a race condition occurs such that a scheduled send message is coincident with a receive request.

5.2.3 Application Layer

All low-level errors are elevated to and handled by the Application Layer.

The message format is outlined below. Reference the Application Protocol for details. Effectively, chip select and chip deselect frame the messages.

A simple BCC-8 Block Check Character should be applied to all the messages. This is an 8-bit Character created by performing consecutive exclusive OR (XOR) operation of the entire message byte-by-byte. The BCC is appended to the last byte of the message as showed in the next diagram. It is always placed as the last byte (position 31). Any unused bytes between the payload and BCC should be padded to 0x00



Reference the application protocol for handling of the BCC-8 errors. This also applies to Slave DMA communication timeouts on the fixed length messages.

5.2.4 DataExchange

Data exchange is a message that the SPI Master sends to the Slave whenever it is only interested in Receiving a message. Because SPI always executes a data exchange, some data must be sent to the Slave in the process of receiving a Slave message. In order to avoid transferring random or meaningless data, the following should be sent by the Host in cases where the action is only executed for reception of data.

A response message intended for the host will remain in the cue until a valid DataExchange sequence is received.

Operations Sent:

DataExchange: (Host to Client):

Class	Oper	Func	Seq	Length	Padding	BCC
0x00	0x02	0x24	Next	0	26 bytes x 0x00	0xyy

5.2.5 SPI Communication Examples

- Assume the amp has been reset
- The SPI Master (Daughtercard) sends the Heartbeat message to the amp.
  - Ex: 00 08 20 NN 00 00 ….. BCC (message is always exactly 32 bytes)
- The SPI_ENET_REQUEST* line will go active (Low) indicating a response is pending
- SPI Master sends the DataExchange Message.
- SPI Master receives the PowerOnReset message (this indicates amp has been reset and will need to have settings such as vol, src, mute sent.)
- The SPI_ENET_REQUEST* line will go active (Low) indicating a response is pending
- SPI Master sends the DataExchange Message.
- SPI Master receives the Heartbeat Status message indicating amp is alive and well
- SPI Master sends Volume, Unmute, Source, etc. normal operating mode

5.3 DataBase Download

A database file will be presented to the Customer for download to the Alto system via one of the above interfaces. In this case all pertinent commands and data shall be packaged in a single file. The Host need only execute the procedure outlined below to download the database to the unit. Databases intended for the MAIN (Entertainment) amplifier board or PA amplifier board are sent through the same channel. The source files are separate for Main and PA.

5.3.1.1 DataBase Source File Format and naming

The source Database files presented for Host consumption are binary files consisting of a string of single-Byte integers. Because the Alto message structure is built around messages of 32 byte length, the file size shall be a multiple of 32. A tuning database can have up to 25 records. This approach is commonly used in an aircraft with multiple amplifiers, although it is also common to have 1 record per Database. It is up to the Host and customer to determine a method that best suits the implemented system.

The loaded record shall be identified as follows: (refer to 4.4.2.5 TuningDatabaseInfo)

- Database Revision (major.minor), Database Version, Aircraft Manufacturer ID, and Aircraft Model ID are the key identifiers that uniquely identify a tuning database and allow for matching to a particular unit in an installation. Alto will keep a cross-reference table recording which Databases correspond with which aircraft installations. A particular Database may be used for multiple installations

- A file containing a single Tuning Database will be delivered to the customer. This should be loaded into the installed unit by the Host CMS. File naming will be: DESCRIPTOR-DAGG-DD-XX-YY-ZZ.atdb, where DA indicates Entertainment amp (or PA indicating PA amp), GG = the Manufacturer code, DD = the Aircraft Model, XX = Database Revision (major), YY = Database Revision (minor), and ZZ = Database Version. The DESCRIPTOR can be any relevant text description of the aircraft program. Revision / Version Numbers can be between 00 and 99. Example file name G650_FCGU-DA06-01-01-01-02.atdb

5.3.1.2 Database download procedure

1) Read the first 32 bytes of the .atdb Database file. All messages will contain any required sequence or BCC information so no processing is required
2) The first message is always DownloadStart. Send this message to the client
3) Wait for Ack/Nack Success message (0x00). If success message is not received after 2 seconds then abort download.
4) Gather next 32 bytes from the database file, send to client, wait an inter-message delay of 10mS.
5) Repeat step 4 till end of file.
6) After last message wait for Ack/Nack Success message to confirm successful download. If the wait lasts more than 1 second then Busy, Waiting to Respond message will be sent each second until final result is sent.
7) If any errors are detected during download by the Client, an error message will be sent to the host. Host can choose to abort download or continue until the end.
8) If a successful Ack/Nack message is not received after transmission of the final message then the download should be considered unsuccessful and repeated.

5.3.1.3 Database Verification

The Host should query the TuningDatabaseInfo (see 4.4.2.5) both after a download, and each time at system boot up to confirm that the correct database is loaded into the amplifier. The loaded database should match Database files assigned to the aircraft in question. Database information of the assigned file can be extracted from the filename (refer to section 5.3.1.1). The Host should compare Aircraft Model, Database Revision (Major), Database Revision (Minor), and Database Version between the queried loaded info and the assigned file info. If these do not match then a load procedure should be executed (see 5.3.1.2). Keep in mind that the PA and Main(entertainment) are separate units therefore the database needs to be verified in both.

5.3.1.4 Configuration Settings

Configuration table data for both Main and PA is available to be adjusted either by the host (4.4.2.3) or manually using the Alto Forte Manager tool. There are spare fields at the end of the Main Configuration table that the Host may use for keeping track of any desired information relating to the load. Each Database record contains default settings for these configuration parameters. When a database is loaded, or the active record is changed then any changes to the active Configuration settings that had been made will be overwritten.