Research & Vehicle Technology
"Infotainment Systems Product Development"

A2B Command and Control API Specification

Infotainment Subsystem Part Specific Specification (SPSS)

Version: Revision 1.2.2

UNCONTROLLED COPY IF PRINTED

Version Date: 4/30/22

FORD CONFIDENTIAL

Revision History

Date	Ver	Notes	
June 24, 2021	**0.1**	**Initial DRAFT Release**	
August 5, 2021	0.2	Updated DRAFT Release	Modifications as follows: 1. Standardized message format to four bytes 2. Added Mailbox definition 3. Added Mailbox message sequence diagram and pseudocode example from ADI 4. Added Message Send Type to all APIs defined 5. Corrected errors in sending direction **6.** Updated all APIs for standard message size in order to accommodate.
November 4, 2021	1.0	Release 1.0	Modifications as follows: 1. Added new messages for Amplifier Diagnostic Information Request, Enable/Disable Frequency Hopping, Amplifier Direct Mute, Speaker Fault Status, Clip Detect Enable 2. Corrected message send type for Amplifier Enable State Message 3. Corrected data direction for A2B Stream Report Status 4. Modified Amplifier State of Health message to combine NVM Error state, add Thermal Shutdown and create general Speaker 1-4 Fault Warnings 5. Added D425 in combination with D245 in all sections. 6. Removed DSP AMP from Frequency Hopping Message 7. Removed ShortToExternalVoltageDetected as redundant in SOH Message 8. Added FreqHoppingEnabled and ClipDetectEnabled status to SOH Message 9. Added DID and DTC Mapping to CAC Message table
December 8, 2021	1.1	Release 1.1	Modifications as follows: 1. Modified AmpDiagStatus Command to allow for Amplifier Thermal Warning Status
February 21, 2022	1.2	Release 1.2	Modifications as follows: 1. Modified Error and Response Handling requirements 2. Added section for message Flow Control for ECU Serial Number and ECU Part Number 3. Modification to Read ECU Serial Number for Flow Control and CRC-16 modification 4. Modification to Read ECU Part Number for Flow Control and CRC-16 modification
February 22, 2022	1.2.1	Release 1.2.1	Modifications as follows: 1. Addition of Flow Control Frame Section 2. Modification to correct message format and description for Read ECU Part Number, Read ECU Serial Number, Message Format
March 30, 2022	1.2.2_DRAFT	Release 1.2.2 - DRAFT	Modifications as follows: 1. Removed Flow Control Frame Section and Message 2. Added single and multi-frame message designation to Protocol Overview and Message Format 3. Corrected applicability for Directed Mute signal 4. Corrected typo in Clip Detect message naming
April 30, 2022	1.2.2	Release 1.2.2 FINAL	Modifications as follows: 1. Modified multiframe protocol for 2 bits instead of 1 to identify frame type. 2. Clarified CRC contents to include all preceding bytes. Notes added to CRC-8 and CRC-16

			3. Removed TIDs and Total Frames from ECU Part Number and ECU Serial Number. Total Frames is Length and that is identified in note. TID is no longer required. 4. Updated retry timeout to 5 retries and timeout to 1200msec to match A2B SPSS 5. Added requirement for subnode behavior during multi-frame transaction. 6. Added definition of node read response vs. response in Abbreviations/Definitions 7. Addition of note to Interface Module Handling of Response as it relates to broadcast message and FIFO method. 8. Added new requirement for confirmation of node response prior to broadcast message send.

Table of Contents

1.1 Scope

This specification describes the command and control for communication between the PAC and external peripherals communicating on the I2C back channel over A2B.

The scope of this specification will be limited to protocol description, API definition for messages supported, and general description of communication flow.

[Note: For detailed information on message usage in the system, please refer to the Global A2B SPSS. For detailed information on hardware setup and software driver requirements, please refer to ADI documentation.]

1.2 Abbreviations and Definitions

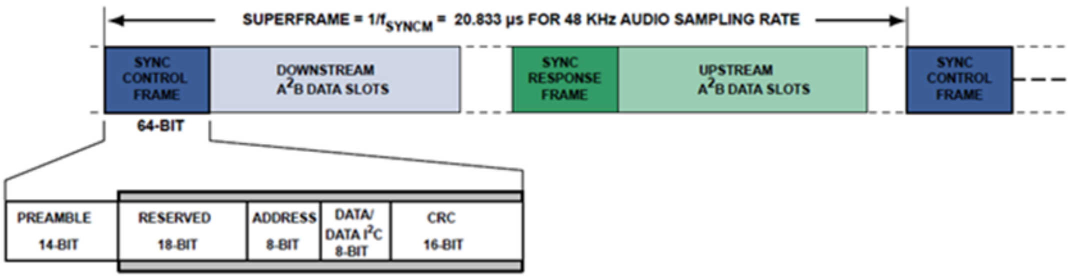
Abbreviation / Definition	Description
AM	Amplitude Modulation
AMP	Amplifier
ASCII	American Standard Code for Information Interchange
CAC	Command and Control
CRC	Cyclic Redundancy Check
dB	DeciBel
EOL	End of Line
EU	Europe
Fct	Function
FM	Frequency Modulation
GND	Ground
HW	Hardware
Hz	Hertz
ID	Identifier
imsbf	(signed) integer most significant bit first
JP	Japan
MCU	Micro Computer Unit
MSB	Most significant bit
PAC	Phoenix Audio Controller
PDC	Phoenix Domain Controller
ROW	Rest Of World

SW	Software or Switch
uimsbf	unsigned integer most significant bit first
US	United States (of America)
Node read response	Indication that an A2B mailbox message request has been read by setting the “empty” interrupt signal.
Response	A2B mailbox message response to an A2B mailbox message request where response is an actual A2B mailbox message sent from the receiving node of the request.

1.3 Protocol Overview

1.3.1 SWR-REQ-334780/E-Protocol Overview

All peripherals shall comply with the following message protocol and utilize A2B I2C mailbox communication *(Note: See page 58 of AD243x TRM for super-frame definition)*.

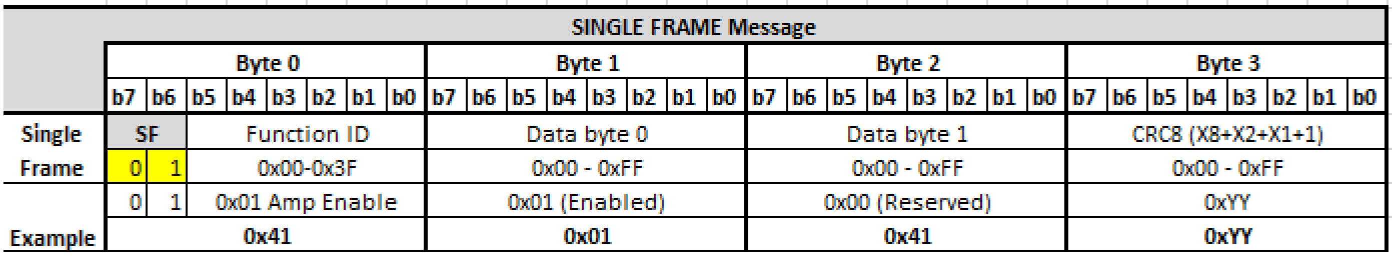


4-3: SCF

Using the mailbox method, a setup would look like the below with an example for Mailbox 0:

Mailbox Identifier	Data Definition
MBOX0B0	Byte 0
MBOX0B1	Byte 1
MBOX0B2	Byte 2
MBOX0B3	Byte 3

The A2B I2C Command and Control (CAC) single frame message shall consist of a 1-byte Function ID with single frame designator, two bytes of subsequent Payload Data, and an 8-bit CRC as follows:

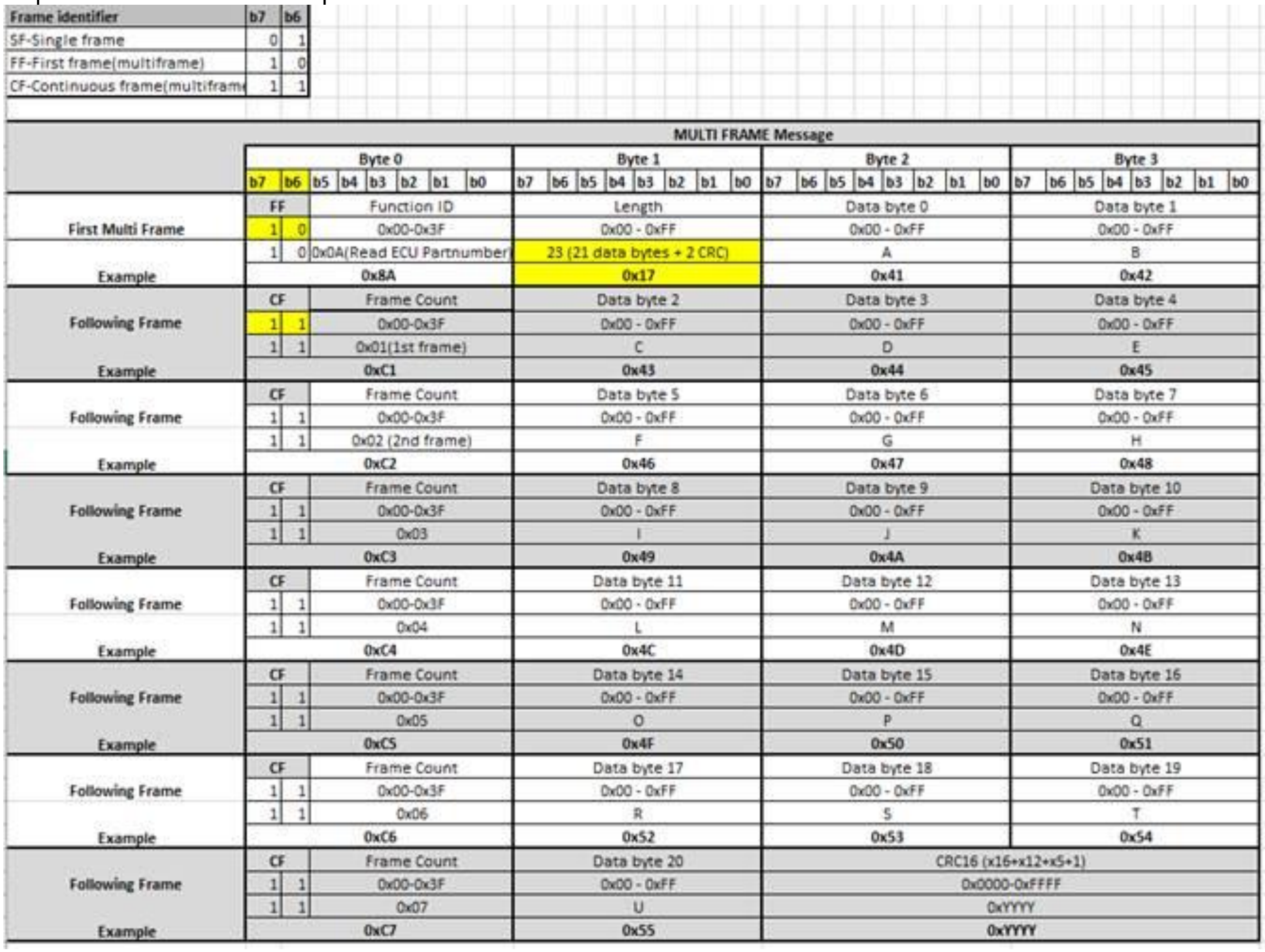


<table>
<tr><th></th><th colspan="32">SINGLE FRAME Message</th></tr>
<tr><th></th><th colspan="8">Byte 0</th><th colspan="8">Byte 1</th><th colspan="8">Byte 2</th><th colspan="8">Byte 3</th></tr>
<tr><th></th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th><th>b7</th><th>b6</th><th>b5</th><th>b4</th><th>b3</th><th>b2</th><th>b1</th><th>b0</th></tr>
<tr><td>Single</td><td colspan="2">SF</td><td colspan="6">Function ID</td><td colspan="8">Data byte 0</td><td colspan="8">Data byte 1</td><td colspan="8">CRC8 (X8+X2+X1+1)</td></tr>
<tr><td>Frame</td><td>0</td><td>1</td><td colspan="6">0x00-0x3F</td><td colspan="8">0x00 - 0xFF</td><td colspan="8">0x00 - 0xFF</td><td colspan="8">0x00 - 0xFF</td></tr>
<tr><td></td><td>0</td><td>1</td><td colspan="6">0x01 Amp Enable</td><td colspan="8">0x01 (Enabled)</td><td colspan="8">0x00 (Reserved)</td><td colspan="8">0xYY</td></tr>
<tr><td>Example</td><td colspan="8">0x41</td><td colspan="8">0x01</td><td colspan="8">0x41</td><td colspan="8">0xYY</td></tr>
</table>

Note: The CRC8 considers the contents of the entire message preceding the CRC8.

The A2B I2C Command and Control (CAC) multi-frame message shall consist of a unique first frame where Byte 0 is a 1-byte Function ID with multi-frame designator, Byte 1 shall be the length of the message, and Bytes 2 and 3 shall be the first two data bytes. All subsequent frames shall begin with Byte 0 as a 1-byte Frame Counter with multi-frame designator and with remaining bytes reserved for data. The last two bytes of the last frame sent shall contain a CRC16. [Note: The CRC16 considers the contents of the entire message preceding the CRC16]

An example for the multi-frame response is shown below:



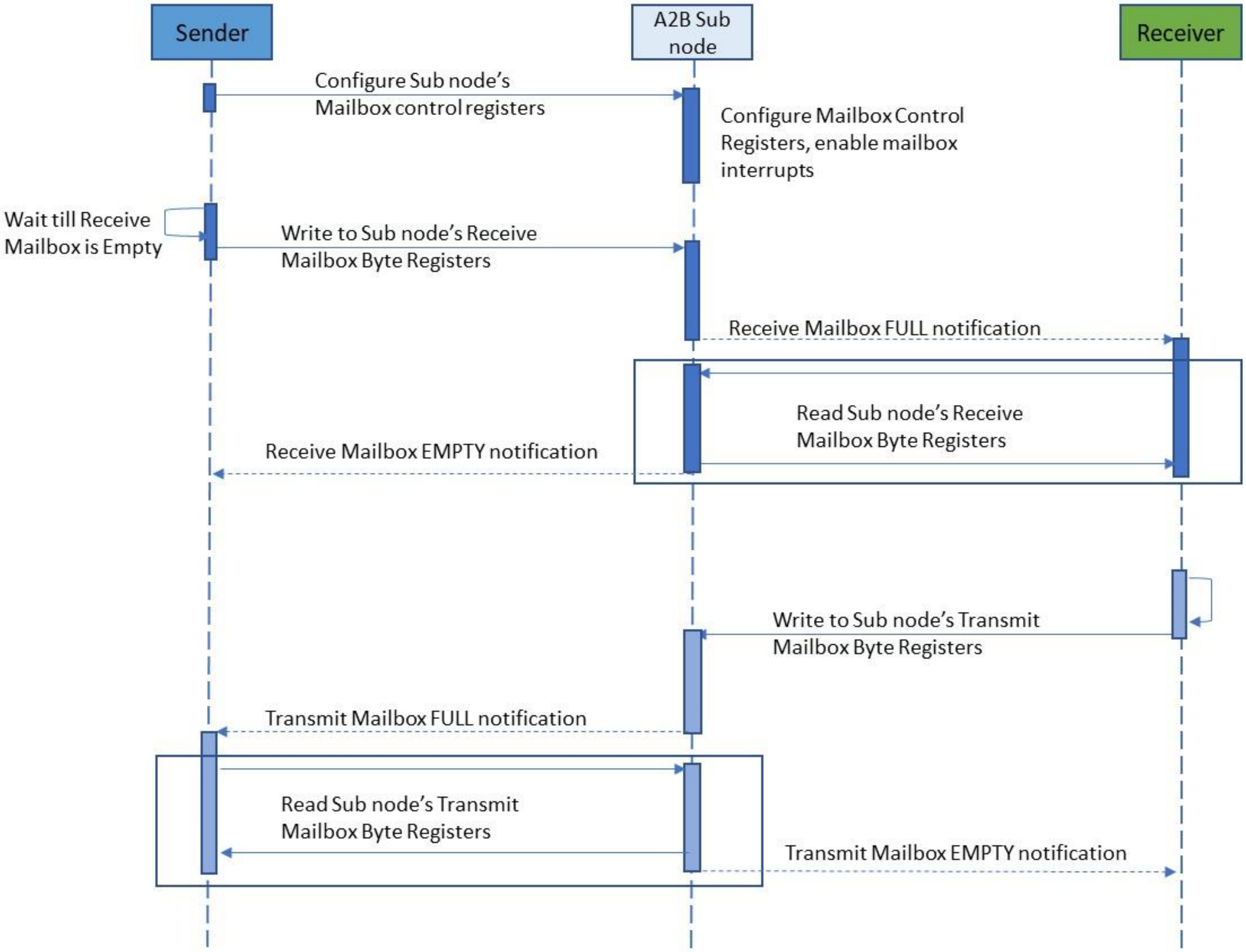
Frame identifier	b7	b6
SF-Single frame	0	1
FF-First frame(multiframe)	1	0
CF-Continuous frame(multiframe	1	1

MULTI FRAME Message

	Byte 0			Byte 1	Byte 2	Byte 3
	b7	b6	b5 b4 b3 b2 b1 b0	b7 b6 b5 b4 b3 b2 b1 b0	b7 b6 b5 b4 b3 b2 b1 b0	b7 b6 b5 b4 b3 b2 b1 b0
First Multi Frame	FF		Function ID	Length	Data byte 0	Data byte 1
	1	0	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
Example	1	0	0x0A(Read ECU Partnumber)	23 (21 data bytes + 2 CRC)	A	B
	0x8A			0x17	0x41	0x42
Following Frame	CF		Frame Count	Data byte 2	Data byte 3	Data byte 4
	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
Example	1	1	0x01(1st frame)	C	D	E
	0xC1			0x43	0x44	0x45
Following Frame	CF		Frame Count	Data byte 5	Data byte 6	Data byte 7
	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
Example	1	1	0x02 (2nd frame)	F	G	H
	0xC2			0x46	0x47	0x48
Following Frame	CF		Frame Count	Data byte 8	Data byte 9	Data byte 10
	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
Example	1	1	0x03	I	J	K
	0xC3			0x49	0x4A	0x4B
Following Frame	CF		Frame Count	Data byte 11	Data byte 12	Data byte 13
	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
Example	1	1	0x04	L	M	N
	0xC4			0x4C	0x4D	0x4E
Following Frame	CF		Frame Count	Data byte 14	Data byte 15	Data byte 16
	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
Example	1	1	0x05	O	P	Q
	0xC5			0x4F	0x50	0x51
Following Frame	CF		Frame Count	Data byte 17	Data byte 18	Data byte 19
	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
Example	1	1	0x06	R	S	T
	0xC6			0x52	0x53	0x54
Following Frame	CF		Frame Count	Data byte 20	CRC16 (x16+x12+x5+1)	
	1	1	0x00-0x3F	0x00 - 0xFF	0x0000-0xFFFF	
Example	1	1	0x07	U	0xYYYY	
	0xC7			0x55	0xYYYY	

1.3.2 SWR-REQ-435456/A-A2B Mailbox Communication Sequence

The A2B mailbox communication sequence shall be as follows:


Sender
A2B Sub node
Receiver
Configure Sub node's Mailbox control registers
Configure Mailbox Control Registers, enable mailbox interrupts
Wait till Receive Mailbox is Empty
Write to Sub node's Receive Mailbox Byte Registers
Receive Mailbox FULL notification
Read Sub node's Receive Mailbox Byte Registers
Receive Mailbox EMPTY notification
Write to Sub node's Transmit Mailbox Byte Registers
Transmit Mailbox FULL notification
Read Sub node's Transmit Mailbox Byte Registers
Transmit Mailbox EMPTY notification

Analog Devices Confidential Information. ©2021 Analog Devices, Inc. All rights reserved.

Example pseudocode:

```
//Mailbox communication - between Master and a Slave node
------------
MASTER SEND:
------------
//MBOXxCTL.MBxFIEN = 1 and MBOXxCTL.MBxEIEN = 1
FIRST TIME: //As there is no interrupt indication at the beginning
if (MBOXxSTAT.MBxEMPTY == 1)
{
    SEND MBOXxB0-(LEN-1) Bytes //all bytes written sequentially
}

NEXT TIME:
if (INTSTAT == 1)
{
    READ INTTYPE and INTSRC
    if (INTTYPE == MBOXx_EMPTY) && (INTSRC.INODE == SlaveNum)
    {
         SEND MBOXxB0-(LEN-1) Bytes //all bytes written sequentially
    }
}


---------------
MASTER RECEIVE:
---------------
//WHEN MBOXyCTL.MByFIEN = 1 and MBOXyCTL.MByEIEN = 1
if (INTSTAT.IRQ == 1)
{
    READ INTTYPE and INTSRC
    if (INTTYPE == MBOXy_FULL) && (INTSRC.INODE == SlaveNum)
    {
          READ MBOXyB0-(LEN-1) Bytes //all bytes read sequentially
    }
}

--------------------------------------------------------------------------------------------------
--------------
SLAVE RECEIVE:
--------------
// WHEN MBOXxCTL.MBxFIEN = 1 and MBOXxCTL.MBxEIEN = 1
if (MBOXxSTAT.MBxFULL == 1)
{
    if (LINTTYPE == MBOXx_FULL)
    {
         READ MBOXxB0-(LEN-1) Bytes //all bytes read sequentially
    }
}

--------------
SLAVE SEND:
--------------
//WHEN MBOXyCTL.MByFIEN = 1 and MBOXyCTL.MByEIEN = 1
FIRST TIME: //As there is no interrupt indication at the beginning
if (MBOXySTAT.MByEMPTY == 1)
{
    SEND MBOXyB0-(LEN-1) Bytes  //all bytes written sequentially
}

NEXT TIME:
If (MBOXySTAT.MByEMPTY == 1)
{
    if (LINTTYPE == MBOXy_EMPTY)
     {
          SEND MBOXyB0-(LEN-1) Bytes //all bytes written sequentially
     }
}
```

1.3.3 SWR-REQ-435457/A-A2B Broadcast Message Description

To send a A2B broadcast message to all sub-nodes, the Main Node (sender) shall issue a broadcast write where the same message is sent to all A2B sub-nodes in the system.

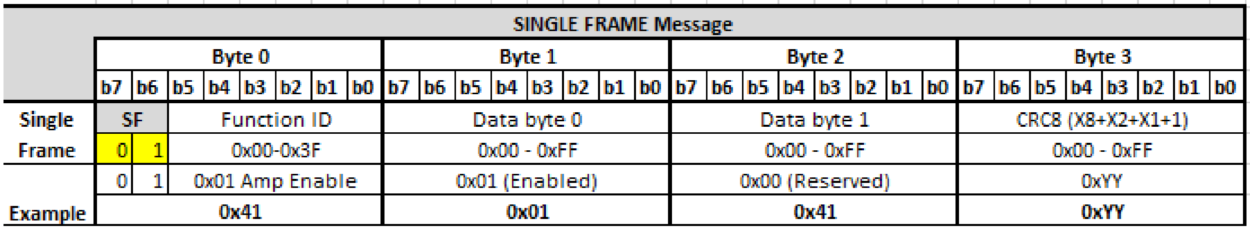
The node closest to the Main Node shall generate the mailbox full interrupt first followed by the next node (e.g. Sub-node 0 shall generate the mailbox full interrupt first followed by Sub-node 1, Sub-node 2 and so on (assuming they all have mailbox configured and mailbox interrupts enabled).

1.3.4 SWR-REQ-483777/D-Single and Multi-Frame Designator Definition

The protocol shall utilize b7 and b6 of Byte 0 as a frame designator where if:

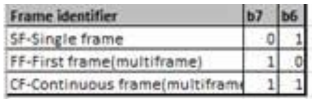
a. b7=0, b6=1, then this is a single frame message AND;
b. b7=1, b6=0, then this is the first frame of a multi-frame message AND;
c. b7=1, b6=1, then this is a continuing frame of a multi-frame message.

An example of a single frame designator use in a single frame message is as follows:

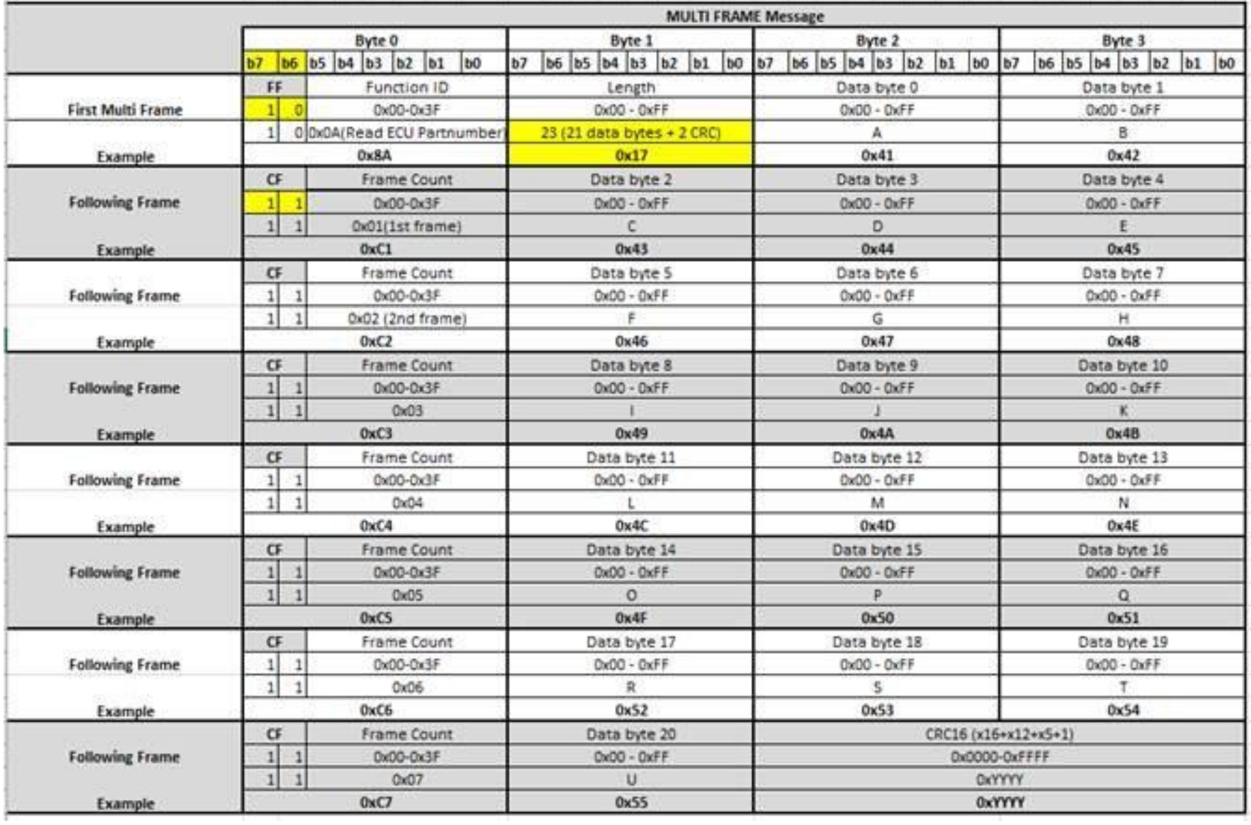


SINGLE FRAME Message						
	Byte 0			Byte 1	Byte 2	Byte 3
	b7	b6	b5 b4 b3 b2 b1 b0	b7 b6 b5 b4 b3 b2 b1 b0	b7 b6 b5 b4 b3 b2 b1 b0	b7 b6 b5 b4 b3 b2 b1 b0
Single	SF		Function ID	Data byte 0	Data byte 1	CRC8 (X8+X2+X1+1)
Frame	0	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
	0	1	0x01 Amp Enable	0x01 (Enabled)	0x00 (Reserved)	0xYY
Example	0x41			0x01	0x41	0xYY

An example of a multi-frame designator use in a multi-frame message is as follows:



Frame identifier	b7	b6
SF-Single frame	0	1
FF-First frame(multiframe)	1	0
CF-Continuous frame(multiframe	1	1



MULTI FRAME Message						
	Byte 0			Byte 1	Byte 2	Byte 3
	b7	b6	b5 b4 b3 b2 b1 b0	b7 b6 b5 b4 b3 b2 b1 b0	b7 b6 b5 b4 b3 b2 b1 b0	b7 b6 b5 b4 b3 b2 b1 b0
	FF		Function ID	Length	Data byte 0	Data byte 1
First Multi Frame	1	0	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
	1	0	0x0A(Read ECU Partnumber	23 (21 data bytes + 2 CRC)	A	B
Example	0x8A			0x17	0x41	0x42
	CF		Frame Count	Data byte 2	Data byte 3	Data byte 4
Following Frame	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
	1	1	0x01(1st frame)	C	D	E
Example	0xC1			0x43	0x44	0x45
	CF		Frame Count	Data byte 5	Data byte 6	Data byte 7
Following Frame	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
	1	1	0x02 (2nd frame)	F	G	H
Example	0xC2			0x46	0x47	0x48
	CF		Frame Count	Data byte 8	Data byte 9	Data byte 10
Following Frame	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
	1	1	0x03	I	J	K
Example	0xC3			0x49	0x4A	0x4B
	CF		Frame Count	Data byte 11	Data byte 12	Data byte 13
Following Frame	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
	1	1	0x04	L	M	N
Example	0xC4			0x4C	0x4D	0x4E
	CF		Frame Count	Data byte 14	Data byte 15	Data byte 16
Following Frame	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
	1	1	0x05	O	P	Q
Example	0xC5			0x4F	0x50	0x51
	CF		Frame Count	Data byte 17	Data byte 18	Data byte 19
Following Frame	1	1	0x00-0x3F	0x00 - 0xFF	0x00 - 0xFF	0x00 - 0xFF
	1	1	0x06	R	S	T
Example	0xC6			0x52	0x53	0x54
	CF		Frame Count	Data byte 20	CRC16 (x16+x12+x5+1)	
Following Frame	1	1	0x00-0x3F	0x00 - 0xFF	0x0000-0xFFFF	
	1	1	0x07	U	0xYYYY	
Example	0xC7			0x55	0xYYYY	

1.3.5 Message Format

1.3.5.1 SWR-REQ-335597/E-Message Format

All peripherals shall support the common datagram representation as follows for messages that are not multi-frame:

Syntax	Nr of bits	
`Message()`		
`{`		
`    FunctionID`	8	uimsbf
`    for (i=0; i<N; i++)`		
`    {`		
`        PayloadDataByte0`	8	uimsbf
`        PayloadDataByte1`	8	uimsbf
`    }`		
`    CRC8`	8	uimsbf
`}`		

Name	Byte	Range	Description
FunctionID	1	Full range	Specifies the function (Note: b0=0 for Single Frame Designation)
PayloadDataByte0 PayloadDataByte1	1 1	Full range Full range	Payload Data Payload Data
CRC8	1	Full range	1-byte CRC following polynomial: X8+x2+x1+1 *[Note: This includes all preceding bytes starting with FunctionID]*

All peripherals shall support the common datagram representation as follows for multi-frame messages:

Frame 0:

Syntax	Nr of bits	
`MessageMultiFrame0()`		
`{`		
`    FunctionID`	8	uimsbf
`    Length`	8	uimsbf
`    PayloadDataByte0`	8	uimsbf
`    PayloadDataByte1`	8	uimsbf
`}`		

Subsequent Frames prior to Last Frame:

Syntax	Nr of bits	
MessageMultiFrameSub()		
{		
FrameCtr	8	uimsbf
PayloadDataByte2	8	uimsbf
PayloadDataByte3	8	uimsbf
PayloadDataByte4	8	uimsbf
}		

Last Frame w/Last Data Byte:

Syntax	Nr of bits	
MessageMultiFrameLast()		
{		
FrameCtr	8	uimsbf
PayloadDataByte(N)	8	uimsbf
CRC16	16	uimsbf
}		

Name	Byte	Range	Description
FunctionID	1	Full range	Specifies the function (Note: b0=1 for Multi-Frame Designation)
FrameCtr	1	0x1-0xFF	Specifies current frame
Length		0x1-0xFF	Specifies total number of frames in message to be received (including CRC16 and preceding bytes)
PayloadDataByte0 PayloadDataByte1 PayloadDataByten	1 1 1	Full range Full range Full range	Payload Data Payload Data End of Payload Data
CRC16	1	Full range	2-byte CRC following polynomial: X16+x12+x5+1 *[Note: This includes all preceding bytes starting with FunctionID]*

1.3.6 Error and Response Handling

1.3.6.1 SWR-REQ-335602/D-Interface Module Handling of Response

The PAC shall induce a retry strategy where it shall retry up to five times before determining that an A2B Communication Error exists IF:

a. A message that is NOT a multiframe message AND a response is NOT received within 1200msec OR;

b. A message is a multiframe message AND a response is NOT received within 50msec.

Notes:

1. *There is no special handling for a broadcast message in terms of retry. Retry will still occur even when a single node does not respond to a broadcast as per the A2B SPSS requirements and the broadcast will be reissued.*

2. *Message transmission is inherently done in a FIFO manner even when retry handling is required.*

1.3.6.2 SWR-REQ-425417/A-Error Handling

A cyclic redundancy check (CRC) is done at the hardware level on each data byte. If the CRC doesn’t match the data, the data is flagged with an error (i.e. CRC ERROR) and is discarded.

Any peripheral on the A2B bus which does not receive an entire message shall consider this an error and take no further action.

1.3.6.3 SWR-REQ-497817/A-Behavior During Multi-Frame Response

During a multi-frame transaction, a subnode shall not send other messages until the multi-frame transaction is complete.

1.3.6.4 SWR-REQ-499058/A-Confirmation of Node Response Prior to Broadcast Message Send

Prior to sending a broadcast message, the Master Node shall confirm a node read response from all sub nodes.

1.4 Messages

1.4.1 Commands Overview

The following command set is defined for control and diagnostic status between the A2B Main Node and Sub-Nodes for modules that communicate using the A2B CAC Protocol [Note: Mapping to the IDS Specification for DID or DTC is given for implementation]:

Function ID	Command	DID/DTC Mapping
0x00	*Not used*	*N/A*
0x01	Amplifier Enable State	DID FD52
0x02	A2B Broadcast State	DID FD53
0x03	A2B Node ID Report Out	DID FD54
0x04	AMP Speaker Output Status	DID FD55
0x05	A2B Stream Content Status	DID FD57
0x06	Amplifier State of Health	DID FD58
0x07	Amplifier Diagnostic Information Request	N/A
0x08	Enable/Disable Frequency Hopping	DID FD59
0x09	Amplifier Direct Mute	DID FD5B
0x0A	Read ECU Part Number	DID F129
0x0B	Read ECU Serial Number	DID F0E8
0x0C	Speaker Fault Status	DTC 9238xx
0x0D	Clip Detect Enable	DID FD5C
0x0E-0x7E	*Reserved for Future Use*	*N/A*
0x80-0xFF	*Reserved for Future Use*	*N/A*

1.4.2 Amplifier Enable State

1.4.2.1 SWR-REQ-371060/B-Amplifier Enable State Message Definition

The *AMPEnableSet* command shall be utilized as a directed command, based on event, from the PAC to Enable or Disable the DSP AMP Module, D425, and D245 AMP Module and to get status on the current Enable/Disable state.

Syntax Description:

Syntax	Nr of Bits	
AMPEnableSet()		
{		
FunctionID	8	uimsbf
AMPEnableState	8	uimsbf
ReservedByte	8	uimsbf
CRC8	8	uimsbf
}		

Command Description:

FunctionID	Data Direction	Parameters	Message Send Type
0x01	PAC to DSP AMP PAC to D245 PAC to D425	AMPEnableState	Event

Parameter Description:

Parameters	Bytes	Range	Description
AMPEnableState	1	0x00-0xFF	0x00: Disabled 0x01: Enabled 0x02-FF: Invalid/Reserved
ReservedByte	1	0x00-0xFF	Reserved Payload Data Byte

1.4.3 A2B Broadcast State

1.4.3.1 SWR-REQ-387217/B-A2B Broadcast State Message Definition

All peripherals shall support the *A2BBroadcast* command.

The PAC shall broadcast the *A2BBroadcast* command to all sub-nodes for communication of the:

a. Muted state of the PAC speakers.
b. Status of audio content (e.g. whether it has been sent or not).
c. Periodic heartbeat for the system; OR
d. Request for A2B Node ID.

<u>Syntax Description:</u>

Syntax	Nr of Bits	
A2BBroadcastState ()		
{		
FunctionID	8	uimsbf
A2BBroadcastState	8	uimsbf
ReservedByte	8	uimsbf
CRC8	8	uimsbf
}		

<u>Command Description:</u>

FunctionID	Data Direction	Parameters	Message Send Type
0x02	PAC to DSP AMP PAC to PDC PAC to D245 PAC to D425	A2BBroadcastState ReservedByte	Broadcast

<u>Parameter Description:</u>

Parameters	Bytes	Description
A2BBroadcastState	1	**Bit 0**: UnMuted (0x1), Muted (0x0) **Bit 1**: Audio Sent (0x1), Audio Not Sent (0x0) **Bit 2:** A2B Node Request (0x1), Default (0x0) **Bits 3-7**: Reserved
ReservedByte	1	Reserved Payload Data Byte

1.4.4 A2B Node ID Report Out

1.4.4.1 SWR-REQ-387223/B-A2B Node ID Report Out Message Definition

The *A2BNodeReportOut* command shall be supported by all sub-nodes to communicate the A2B Node ID to the PAC module as a response to the request of the *A2BBroadcastState* (Bit 2=1, A2B Node Request) command.

Syntax Description:

Syntax	Nr of Bits	
A2BNodeReportOut ()		
{		
FunctionID	8	uimsbf
A2BNodeID	8	uimsbf
ReservedByte	8	uimsbf
CRC8	8	uimsbf
}		

Command Description:

FunctionID	Data Direction	Parameters	Message Send Type
0x03	DSP AMP to PAC PDC to PAC D245 to PAC D425 to PAC	A2BNodeID ReservedByte	Event

Parameter Description:

Parameters	Bytes	Description
A2BNodeID	1	Node ID assigned to the Sub-Node
ReservedByte	1	Reserved Payload Data Byte

1.4.5 AMP Speaker Output Status

1.4.5.1 SWR-REQ-387224/B-AMP Speaker Output Status Message Definition

The *AMPSpeakerOutputStatus* shall be supported by the PAC and Amplifier modules as an event-based message. This message is sent from the AMP Module/Modules (DSP AMP or D245/D425) to give the PAC status on the Mute or Unmute of the system correlating to whether it is ready to play sounds.

Syntax Description:

Syntax	Nr of Bits	
AMPSpeakerOutputStatus()		
{		
FunctionID	8	uimsbf
SpeakerStatus	8	uimsbf
ReservedByte	8	uimsbf
CRC8	8	uimsbf
}		

Command Description:

Function ID	Data Direction	Parameters	Message Send Type
0x04	DSP AMP to PAC D245 to PAC D425 to PAC	SpeakerStatus ReservedByte	Event

Parameter Description:

Parameters	Bytes	Description
SpeakerStatus	1	0x00: Default (Muted) 0x01: UnMuted 0x02-0xFF: Invalid/Reserved
ReservedByte	1	Reserved Payload Data Byte

1.4.6 A2B Stream Report Status

1.4.6.1 SWR-REQ-387225/B-A2B Stream Report Status Message Definition

The *A2BStreamReportStatus* message shall be supported by both the PDC and the PAC. This message is an event-based status command from the PDC to the PAC to communicate audio status.

Syntax Description:

Syntax	Nr of Bits	
A2BStreamReportStatus()		
{		
FunctionID	8	uimsbf
AudioStatus	8	uimsbf
ReservedByte	8	uimsbf
CRC8	8	uimsbf
}		

Command Description:

FunctionID	Data Direction	Parameters	Message Send Type
0x05	PDC to PAC	AudioStatus ReservedByte	Event

Parameter Description:

Parameters	Bytes	Description
AudioStatus	1	0x00: AudioNotSent 0x01: AudioSent 0x02-FF: Invalid/Reserved
ReservedByte	1	Reserved Payload Data Byte

1.4.7 Amplifier State of Health

1.4.7.1 SWR-REQ-387226/C-Amplifier State of Health Message Definition

The *AmplifierSOH* command shall be supported by the D245/D425 Amplifiers and the PAC. This message shall communicate diagnostic status (e.g. state of health) to the PAC on an event-periodic basis for purposes of logging amplifier DTCs and taking fault or functional action accordingly. The message is bit encoded where bit=1 indicates that the condition is ACTIVE and bit=0 indicate that the condition is INACTIVE.

Syntax Description:

Syntax	Nr of Bits	
AmplifierSOH ()		
{		
FunctionID	8	uimsbf
AmpDiagStatus	16	uimsbf
CRC-8	8	uimsbf
}		

Command Description:

FunctionID	Data Direction	Parameters	Message Send Type
0x06	D245 to PAC D425 to PAC DSP AMP to PAC (Optional)	AmpDiagStatus	Event-Periodic (5 sec period)

Parameter Description:

Parameters	Byte #	Description
AmpDiagStatus	Byte 1	**Bit 0:** OverTemperatureProtectionActive **Bit 1:** OverVoltageProtectionActive **Bit 2:** NVMErrorActive **Bit 3:** MPUPeripheralErrorDetected **Bit 4:** WatchdogTimerResetDetected **Bit 5:** UnderVoltageProtectionActive **Bit 6:** FreqHoppingEnabled **Bit 7:** ClipDetectEnabled
	Byte 2	**Bit 0:** Speaker1FltActive **Bit 1:** Speaker2FltActive **Bit 2:** Speaker3FltActive **Bit 3:** Speaker4FltActive **Bit 4:** ThermalWarningActive **Bit 5:** Reserved **Bit 6:** Reserved **Bit 7:** Reserved

1.4.8 Amplifier Diagnostic Info Request

1.4.8.1 SWR-REQ-456117/A-Amplifier Diagnostic Info Request Message Definition

The *AmpDiagInfoReq* message shall be supported by both the PAC and the amplifier modules (D245/D425). This message is an event-based request command from the PAC to the amplifiers to request specific diagnostic information.
Note: As D245 amplifier will only support two channels, Speaker 3 and Speaker 4 fault requests are not supported.

Syntax Description:

Syntax	Nr of Bits	
AmpDiagInfoReq()		
{		
FunctionID	8	uimsbf
DiagInfoReq	8	uimsbf
ReservedByte	8	uimsbf
CRC8	8	uimsbf
}		

Command Description:

FunctionID	Data Direction	Parameters	Message Send Type
0x07	PAC to D245 PAC to D425	DiagInfoReq ReservedByte	Event

Parameter Description:

Parameters	Bytes	Description	Notes
DiagInfoReq	1	0x00: Invalid/Null 0x01: ECUPartNumber 0x02: ECUSerialNumber 0x03: Speaker1Faults 0x04: Speaker2Faults 0x05: Speaker3Faults (not supported for D245) 0x06: Speaker4Faults (not supported for D245) 0x07-0xFF: Invalid/Reserved	The DiagInfoReq byte is meant to handle one request at a time. It is not possible to request all diagnostic information at the same time or in combination.
ReservedByte	1	Reserved Payload Data Byte	

1.4.9 Enable/Disable Frequency Hopping

1.4.9.1 SWR-REQ-456118/A-Enable/Disable Frequency Hopping Message Definition

The *FrequencyHopSet* command shall be utilized as a direct event command from the PAC to Enable or Disable frequency hopping in the D245/D425 AMP modules and if enabled, the frequency to be used shall be given.

Syntax Description:

Syntax	Nr of Bits	

FreqHopSet()
{		

FunctionID	8	uimsbf
FreqHopState	8	uimsbf
AssignedFreq	8	uimsbf
CRC8	8	uimsbf
}		

Command Description:

FunctionID	Data Direction	Parameters	Message Send Type
0x08	PAC to D245 PAC to D425	FreqHopState AssignedFreq	Event Message

Parameter Description:

Parameters	Bytes	Range	Description
FreqHopState	1	0x00-0xFF	0x00: Disabled 0x01: Enabled 0x02-FF: Invalid/Reserved
AssignedFreq	1	0x00-0xFF	0x00: 8 x fS (352.8 kHz/384 kHz) 0x01: 10 x fS (441 kHz/480 kHz) 0x02: Reserved 0x03: Reserved 0x04: Reserved 0x05: 38 x fS (1.68 MHz/1.82 MHz) 0x06: 44 x fS (1.94 MHz/2.11 MHz) 0x07: 48 x fS (2.12 MHz/ not supported) 0x08-0xFE: Reserved 0xFF: IF FreqHopState=Disabled

1.4.10 Amplifier Directed Mute

1.4.10.1 SWR-REQ-456119/B-Amplifier Directed Mute Message Definition

The *AMPDirectMute* message shall be supported by both the PAC, D425, and the D245 AMP modules. This message is an event-based status command from the PAC to the amplifiers to directly mute or unmute.

Syntax Description:

Syntax	Nr of Bits	
AMPDirectMute()		
{		
FunctionID	8	uimsbf
DirectMute	8	uimsbf
ReservedByte	8	uimsbf
CRC8	8	uimsbf
}		

Command Description:

FunctionID	Data Direction	Parameters	Message Send Type
0x09	PAC to D245 PAC to D425	DirectMute ReservedByte	Event

Parameter Description:

Parameters	Bytes	Description
DirectMute	1	0x00: AudioMute 0x01: AudioUnmute 0x02-FF: Invalid/Reserved
ReservedByte	1	Reserved Payload Data Byte

1.4.11 Read ECU Part Number

1.4.11.1 SWR-REQ-456120/D-Read ECU Part Number Message Definition

The *ECUPartNumber* message shall be supported by both the PAC and D245/D425 Modules. This message is an event-based command from to the PAC from D245/D425 module to communicate the ECU Part Number based on a *DiagInfoRequest* where it is requested.
[Note: This message is a multi-frame response, the first frame will include a Length byte, not shown in the Syntax Description below]

Syntax Description:

Syntax	Nr of Bits	
ECUPartNumber()		
{		
FunctionID	8	uimsbf
FrameCtr	8	uimsbf
ECUPNByte1	8	uimsbf
ECUPNByte2	8	uimsbf
ECUPNByte3	8	uimsbf
ECUPNByte4	8	uimsbf
ECUPNByte5	8	uimsbf
ECUPNByte6	8	uimsbf
ECUPNByte7	8	uimsbf
ECUPNByte8	8	uimsbf
ECUPNByte9	8	uimsbf
ECUPNByte10	8	uimsbf
ECUPNByte11	8	uimsbf
ECUPNByte12	8	uimsbf
ECUPNByte13	8	uimsbf
ECUPNByte14	8	uimsbf
ECUPNByte15	8	uimsbf
ECUPNByte16	8	uimsbf
ECUPNByte17	8	uimsbf
ECUPNByte18	8	uimsbf
ECUPNByte19	8	uimsbf
ECUPNByte20	8	uimsbf
CRC16	16	uimsbf
}		

Command Description:

FunctionID	Data Direction	Parameters	Message Send Type
0x0A	D245 to PAC D425 to PAC	ECUPNByte1-20	Event

Parameter Description:

Parameters	Bytes	Description
FrameCtr	1	Count of current frame of multi-frame response
ECUPNByte[n]	1-20	ECU Part Number (20-Bytes) ASCII

1.4.12 Read ECU Serial Number

1.4.12.1 SWR-REQ-456121/D-Read ECU Serial Number Message Definition

The *ECUSerialNumber* message shall be supported by both the PAC and D245/D425 modules. This message is an event-based command from to the PAC from the D245/D425 module to communicate the ECU Serial Number based on a *DiagInfoRequest* where it is requested.
[Note: This message is a multi-frame response, the first frame will include a Length byte, not shown in the Syntax Description below]

Syntax Description:

Syntax	Nr of Bits	
ECUSerialNumber()		
{		
FunctionID	8	uimsbf
FrameCtr	8	uimsbf
ECUSNByte1	8	uimsbf
ECUSNByte2	8	uimsbf
ECUSNByte3	8	uimsbf
ECUSNByte4	8	uimsbf
ECUSNByte5	8	uimsbf
ECUSNByte6	8	uimsbf
ECUSNByte7	8	uimsbf
ECUSNByte8	8	uimsbf
ECUSNByte9	8	uimsbf
ECUSNByte10	8	uimsbf
ECUSNByte11	8	uimsbf
ECUSNByte12	8	uimsbf
ECUSNByte13	8	uimsbf
ECUSNByte14	8	uimsbf
ECUSNByte15	8	uimsbf
ECUSNByte16	8	uimsbf
CRC16	16	uimsbf
}		

Command Description:

FunctionID	Data Direction	Parameters	Message Send Type
0x0B	D245 to PAC D425 to PAC	ECUSNByte1-16	Event

Parameter Description:

Parameters	Bytes	Description
FrameCtr	1	Count of current frame of multi-frame response
ECUSNByte[n]	1-16	ECU Serial Number (16-Bytes) ASCII

1.4.13 Speaker Fault Status

1.4.13.1 SWR-REQ-456137/A-Speaker Fault Status Message Definition

The *SpeakerFltStatus* command shall be supported by the PAC and the amplifier (D245/D425). This message shall be in response to a DiagInfoRqst where a specific speaker fault status is requested AND upon qualification of a speaker fault. This message is sent for a single speaker.

Syntax Description:

Syntax	Nr of Bits	
SpeakerFltStatus () {		
FunctionID	8	uimsbf
SpeakerID	8	uimsbf
SpeakerStatus	8	uimsbf
CRC-8	8	uimsbf
}		

Command Description:

FunctionID	Data Direction	Parameters	Message Send Type
0x0C	D245 to PAC D425 to PAC	SpeakerID SpeakerStatus	Event

Parameter Description:

Parameters	Byte #	Description
SpeakerID	Byte 1	**Bit 0:** Speaker1 **Bit 1:** Speaker2 **Bit 2:** Speaker3 (not used for D245) **Bit 3:** Speaker4 (not used for D245) **Bit 4-7:** Reserved
SpeakerStatus	Byte 2	**Bit 0:** ShortAcrossLoadDetected **Bit 1:** ShortToSupplyDetected **Bit 2:** ShortToGroundDetected **Bit 3:** CurrentLimitingActive **Bit 4:** OutputDCOffsetDetected **Bit 5:** OpenCircuitDetected **Bit 6:** Reserved **Bit 7:** Reserved

1.4.14 Clip Detect Enable

1.4.14.1 SWR-REQ-456147/B-Clip Detect Enable Message Definition

The *ClipDetectSet* command shall be utilized as a direct event command from the PAC to Enable or Disable clip detect in the amplifier (D245/D425 module) and if enabled, one of the thirteen settings specified shall be utilized. These thirteen settings are defined by the EQ settings stored within the PAC.

Syntax Description:

Syntax	Nr of Bits	
ClipDetectSet()		
{		
FunctionID	8	uimsbf
ClipDetectState	8	uimsbf
ClipDetectSetting	8	uimsbf
CRC8	8	uimsbf
}		

Command Description:

FunctionID	Data Direction	Parameters	Message Send Type
0x0D	PAC to D245 PAC to D425	ClipDetectState ClipDetectSetting	Event

Parameter Description:

Parameters	Bytes	Range	Description
ClipDetectState	1	0x00-0xFF	0x00: Disabled 0x01: Enabled 0x02-FF: Invalid/Reserved
ClipDetectSetting	1	0x00-0xFF	0x00: Invalid 0x01-0x0D: Setting1-Setting13 0x0E-0xFE: Reserved 0xFF: IF ClipDetectState=Disabled