Using the AXN/CBM/401

TEC/NOT/095

This paper discusses the following topics:

64.1 Introducing CAN bus traffic

It is essential to understand the structure of CAN bus traffic to fully comprehend how the AXN/CBM/401 monitors it.

64.1.1 The physical layer

The physical layer is a differential two-wire interface with CANH and CANL wires for each bus. Bit encoding used is Non-Return to Zero (NRZ) encoding (with bit-stuffing).

The use of NRZ encoding ensures compact messages with a minimum number of transitions and high resilience to external disturbance. Cable length depends on the data rate used (40 meters for 1 Mbps).

The CAN bus uses the following drive voltages:

- High - 2.75 to 4.5 volts
- Low - 0.5 to 2.25 volts
- Differential - 1.5v to 3.0 volts

64.1.2 Word definition for CAN 2.0 A/B

A CAN network can be configured to work with two different message (frame) formats: CAN 2.0 A (standard/base frame format) or CAN 2.0 B (extended frame format). The following two figures show the structure of the two formats.

The only difference between the two formats is the bit length supported for the identifier (ID). CAN 2.0 A supports a length of 11 bits for the ID; CAN 2.0 B supports a length of 29 bits for the ID—this comprises the 11-bit ID and an 18-bit extension (IDE). The IDE bit is transmitted as dominant in the case of an 11-bit frame, and transmitted as recessive in the case of a 29-bit frame.

CAN controllers that support CAN 2.0 B messages are also able to send and receive messages in CAN 2.0 A. All frames begin with a Start-Of-Frame (SOF) bit which denotes the start of the frame transmission. For definitions of fields shown in the following two figures, see Table 64-1 on page 2.

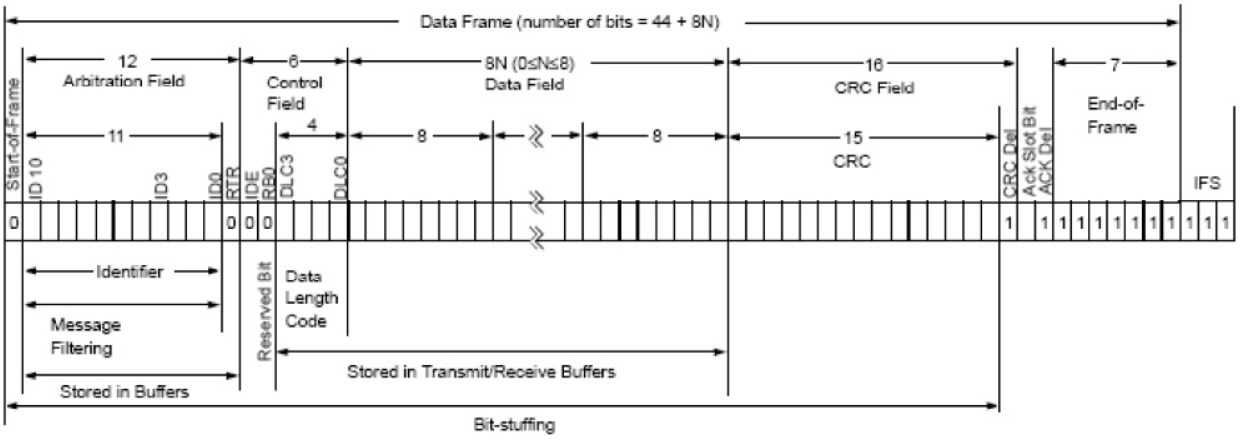


Figure 64-1: CAN 2.0 A

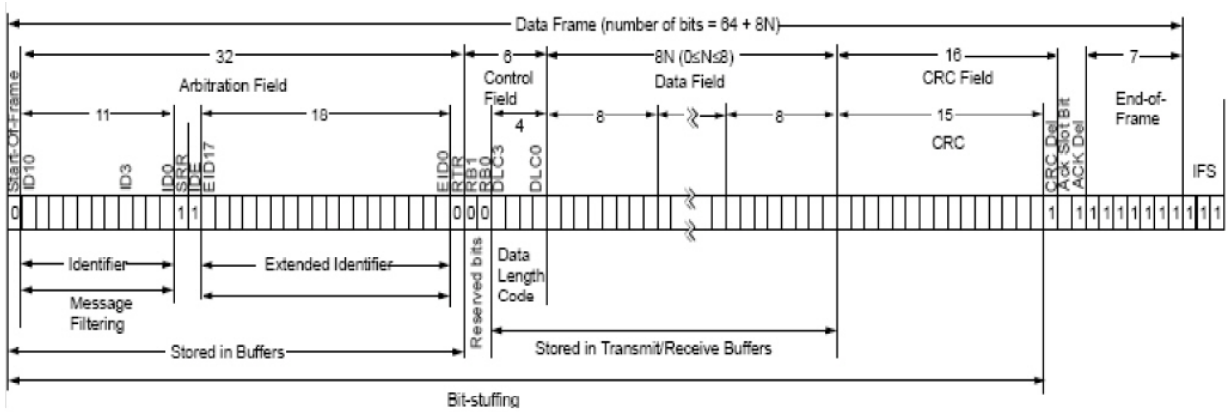


Figure 64-2: CAN 2.0 B

Table 64-1: Frame format fields

Frame format field	Description
ID (Identifier) / IDE (Extended IDentifier)	CAN2.0 A—11 bits; CAN2.0 B—29 bits
SSR	Substitute remote request - CAN2.0 B (masked by default)
RTR	Remote transmission request - CAN 2.0 A
DLC (Data Length Code)	Number of data bytes to be transmitted (0 to 8 bytes)
Data field	Data to be transmitted

64.1.3 Word definition for CAN FD

CAN FD is short for CAN with Flexible Data rate. There are three main differences between CAN FD and Classical CAN (CAN 2A/B explained above): Bit Rate Switching (BRS); maximum size of the data payload; and the coverage of the CRC.

64.1.3.1 Bit Rate Switching

In CAN FD you can have two different bit-rates for different parts of the frame. This feature is called Bit Rate Switching (BRS) and is enabled by a new control bit called BRS, added to the existing control bits between the CAN ID and the Data Length Code (DLC).

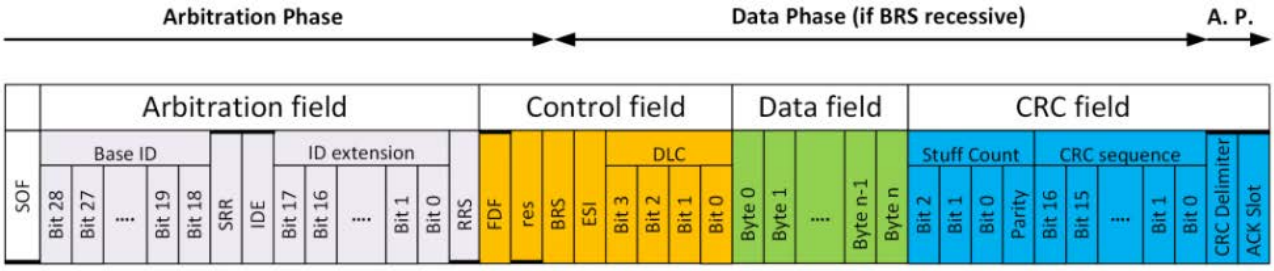


Figure 64-3: CAN FD

The new bits from CAN FD compared to classical CAN are:

- FDF: Bit to distinguish between classical CAN and CAN FD frame.
- Res: Reserved
- BRS: Bit Rate Switching. This indicates whether the bit-rate is going to stay the same or switch up to a faster rate. The arbitration field is always transmitted at the nominal bit-rate (in the AXN/CBM/40x, it's called the Arbitration Bit Rate), and if BRS is recessive, the bit-rate switches up to a higher data bit-rate at the sample point of the BRS bit (In the AXN/CBM/40x, it's called the Data Bit Rate).

- ESI: Error State Indicator (this bit is ignored on the AXN/CBM/40x).

64.1.3.2 Maximum size and CRC

The following table shows the relationship between DLC and the size of the data payload for both Classical CAN and CAN FD. CAN FD supports the classical CAN data length, but it also supports 12,16, 20, 24, 32, 48 and 64 bytes.

CAN FD uses either a 17-bit CRC for data fields up to and including 16 bytes, or a 21-bit CRC for data fields 20 bytes and over.

Table 64-2: Relationship between DLC and the size of the data payload for both Classical CAN and CAN FD

DLC Indicator	DLC (binary)	Classical CAN (bytes)	CAN FD (bytes)	Classical CAN CRC-	CAN FD CRC-
0	0000	0	0	15	17
1	0001	1	1	15	17
2	0010	2	2	15	17
3	0011	3	3	15	17
4	0100	4	4	15	17
5	0101	5	5	15	17
6	0110	6	6	15	17
7	0111	7	7	15	17
8	1000	8	8	15	17
9	1001	8	12	15	17
10	1010	8	16	15	17
11	1011	8	20	15	21
12	1100	8	24	15	21
13	1101	8	32	15	21
14	1110	8	48	15	21
15	1111	8	64	15	21

64.2 Module overview

The AXN/CBM/401 is an 8-channel CAN 2.0 A, CAN 2.0 B or CAN FD which can parse and/or packetize each channel at the same time. The AXN/CBM/402 is the same module as the AXN/CBM/401 but with acknowledgment support. While the AXN/CBM/401 physically cannot transmit anything to the bus, the AXN/CBM/402 can transmit an acknowledgment if the device monitored is requesting it.

Both modules are compatible with ARINC-825 and CANaerospace, that is, at the datalink layer level. In this document, reference to the AXN/CBM/401 can be applied to the AXN/CBM/402.

Screen shots and descriptions of settings shown in this technical note are from DAS Studio 3 software. DAS Studio 3 is used to create a configuration, which contains the various elements that make up your data acquisition system. You then use this configuration file to manage and program these elements.

To see how hardware is represented in the DAS Studio 3 graphical user interface, refer to the *DAS Studio 3 User Manual*.

64.3 Parser operation

64.3.1 How parsing works

Like other Curtiss-Wright bus monitors, the AXN/CBM/40x uses a triple buffer for parsing. The following figure illustrates the triple buffering of data words (green) and message tags (white) used for each bus in the AXN/CBM/401's parser.

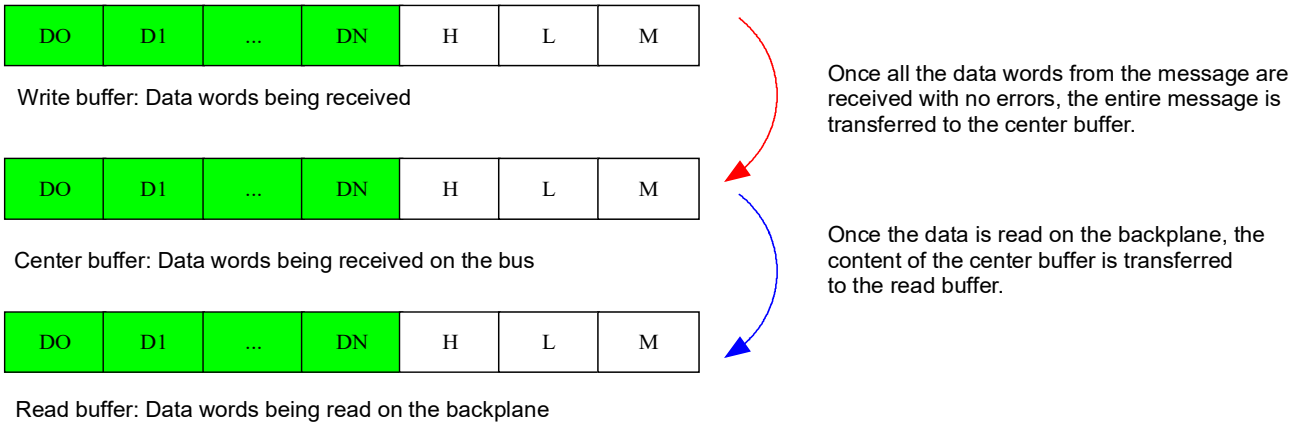


Figure 64-4: Triple buffering of traffic and associated message tags

In the previous figure, D0, D1, D2, Dn. corresponds to the traffic data words with n < 32.

The time tags H, L, M correspond to High time, Low time and Micro time, which is the last bit of the message with a 1-µsec resolution. Each buffer will also contain some message information such as the StandardID, DLC and ExtendedID. They are not represented on the above figure for simplification.

The way triple buffering works is as follows:

Time message tags are added to each message received and stored in separate buffers for each of the busses. As soon as a message is received with no errors, the contents of the write buffer is transferred to the center buffer. If the data in the center buffer has not been transferred to a read buffer, a skipped flag is set.

As soon as the last parameter of interest has been read from the read buffer by the backplane, the contents of the center buffer (if new) are transferred to the read buffer. If no new data word has been received, the stale flag is set. A center and read buffer exist for every message ID (parser slot). Skipped and stale bits can be found in the Message Info register to indicate whether messages are lost or repeated (under sampling or oversampling situations).

Additional tags such as MessageCount, MessageDLC, MessageStdId, MessageExtId and MessageInfo registers are also available as additional information and can be added from DAS Studio's CAN Builder application as explained in "64.5.1 Defining parsing rules" on page 6. For further information regarding these registers, refer to the *AXN/CBM/40x* data sheet.

64.4 Module Settings tab

64.4.1 Parser Data Endianness and Fill Value

In the parser, a total of up to 127 complete messages are triple buffered so that the stale indication is message wide. Each message can be up to 64 characters (bytes) long. Each message is tagged to 1 us resolution.

NOTE: To view the screen shots shown in this section in DAS Studio 3, ensure the AXN/CBM/401 module is in context and the Settings tab is selected.

To configure the parser, first you must decide on the endianness of the data you wish to receive then on the Fill Value. The Fill Value is the value for which the data will display until a message is received on this parser slot.

As shown in the following figure, there are two choices available for Parser Data Endianness.

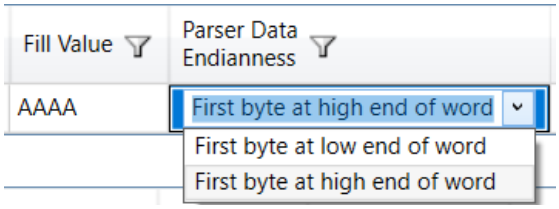


Figure 64-5: Parser Data Endianness and Fill Value settings

64.4.2 Setting up the incoming signal

In this section you define the signal type of the incoming data for that channel.

The following figure shows available CAN Settings.

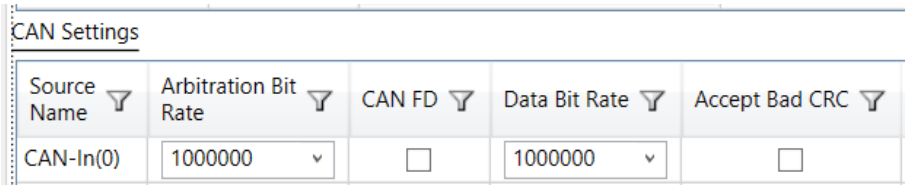


Figure 64-6: CAN settings on channel 0

Settings	Description
Arbitration Bit Rate	Arbitration bit-rate is for CAN 2.0 A/B but also for CAN FD non data bits, that is, it applies to the arbitration phase (see Figure 64-3 on page 2).
CAN FD	When selected, CAN FD is enabled on the channel.
Data Bit Rate	This field is not used for CAN 2.0 A/B. Data bit-rate is the same as the arbitration bit-rate for CAN 2.0 A/B. For CAN FD, it corresponds to the data phase (see Figure 64-3 on page 2).
Accept Bad CRC	When selected, packetization of messages is enabled where the CRC check fails. This field doesn't apply to parsing. The module will not be capable of parsing a message with the wrong CRC.

64.4.3 Global parameters settings

Like most bus monitors, the AXN/CBM/401 has global parameters, which can be used for debugging. The main global parameters are Report word and several counters such as module, message and error counters as described in the following table.

Source Name	Parameter Type	Parameter Name
MyAXN_CBM_401	Report	***P_MyAXN_CBM_401_Report***
MyAXN_CBM_401	ModuleMessageCount	***P_MyAXN_CBM_401_ModuleMessageCount***
MyAXN_CBM_401	ReadCounter	***P_MyAXN_CBM_401_ReadCounter***
CAN-In(0)	ChannelMessageCount	***P_MyAXN_CBM_401_CAN-In(0)_ChannelMessageCount***
CAN-In(0)	ChannelErrCount	***P_MyAXN_CBM_401_CAN-In(0)_ChannelErrCount***

Figure 64-7: Global parameters in DAS Studio 3 Settings tab

Settings	Description
Report	The Report word is a 16-bit register, which provides information regarding errors detected. The Report word is recommended to be monitored as a debug register when abnormal conditions are detected. Refer to the *AXN/CBM/401* data sheet for further information.
ModuleMessageCount	ModuleMessageCount increments by one each time the parser logic detects a complete message. This register counts how many messages the module has parsed by any of the channels.
ChannelMessageCount (0 to 7)	Increments by one each time the parser logic detects a complete message on this channel. Note: Do not confuse this register with MessageCount, which corresponds to a counter added to each message.
ChannelErrCount (0 to 7)	Count of errors detected on this bus. See “64.4.4 Errors” on page 6 for error definitions.
ModuleTemperature	Temperature of the AXN/CBM/401 module. Refer to the *AXN/CBM/401* data sheet.

64.4.4 Errors

There are several errors reported by the AXN/CBM/401. As explained in the previous section, the Report word provides an indicator of the different errors detected in any of the busses, and ChannelErrCount provides an indicator of the number of errors detected on each bus. Additional packetizer headers contains an error flag Er (1 bit) and a 6 bits Error Code indicator.

The supported errors are:

- Bit error. Only for the AXN/CBM/402. Recessive bit detected when sending dominant bit. (Only possible if transmitting is enabled.)
- Stuff Error. Too many consecutive equal bit levels without detection of stuff bit.
- CRC Error. Calculated CRC does not match received CRC.
- Form Error. Fixed form bit field contains illegal bits. It indicates a problem with the format of the message.
- Ack Error. Acknowledgement dominant bit not detected in Ack Slot.

All errors are defined in the CAN Specification.

64.5 CAN-Bus Builder

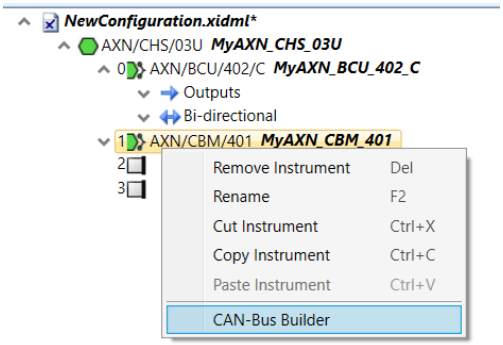
For the following sections, use the CAN-Bus Builder application in DAS Studio 3. Refer to the “Builder application GUI overview” section of the *DAS Studio 3 User Manual* for a brief overview of navigating the application.

64.5.1 Defining parsing rules

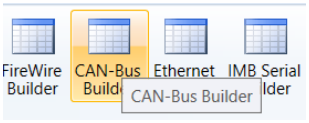
After you have all channel settings configured, refer to the following to define rules to identify messages.

1. Do one of the following.

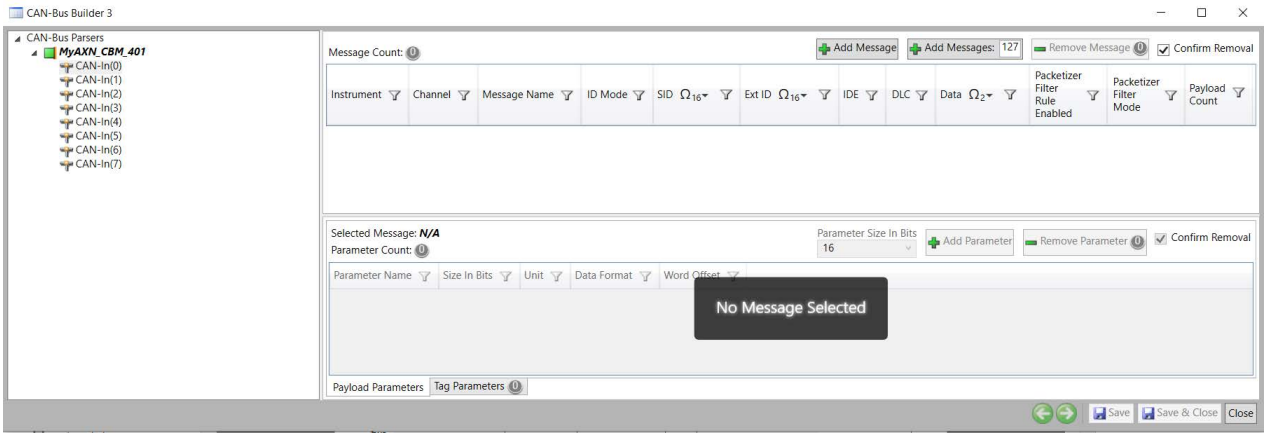
- In the Navigator, right-click the AXN/CBM/401 module and then click **CAN-Bus Builder**.



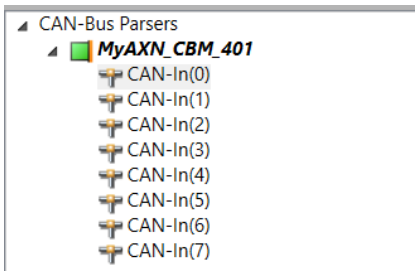
- On the **Applications** tab click **CAN-Bus Builder**.



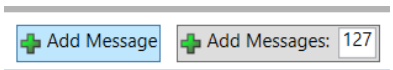
The CAN-Bus Builder application opens.



2. In the Navigator (left pane), select the channel on the AXN/CBM/401 that you want to parse data on.



3. Click **Add Message** to add a single message. To add multiple messages (up to 127), click **Add Messages** (typing the number of messages in the field).



64.5.2 Message definition

Now you must define the rules to identify or parse the desired message.

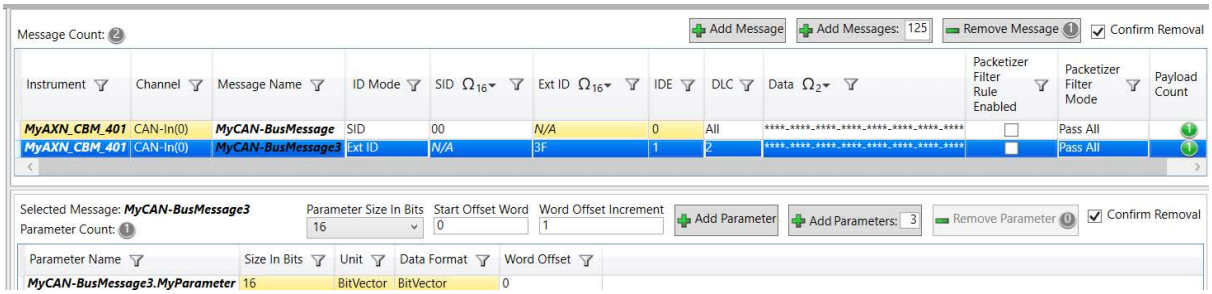
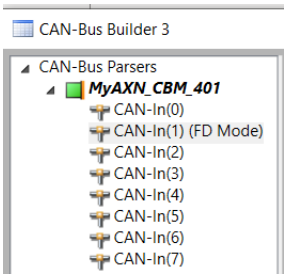


Figure 64-8: Example of the setup of each message type available in the AXN/CBM/401 using CAN2.0 A/B

If the message is set to **CAN FD** on the Settings tab, the CAN-Bus Builder shows the channel is in **FD Mode**.



It also indicates that you can pick a DLC higher than 8.

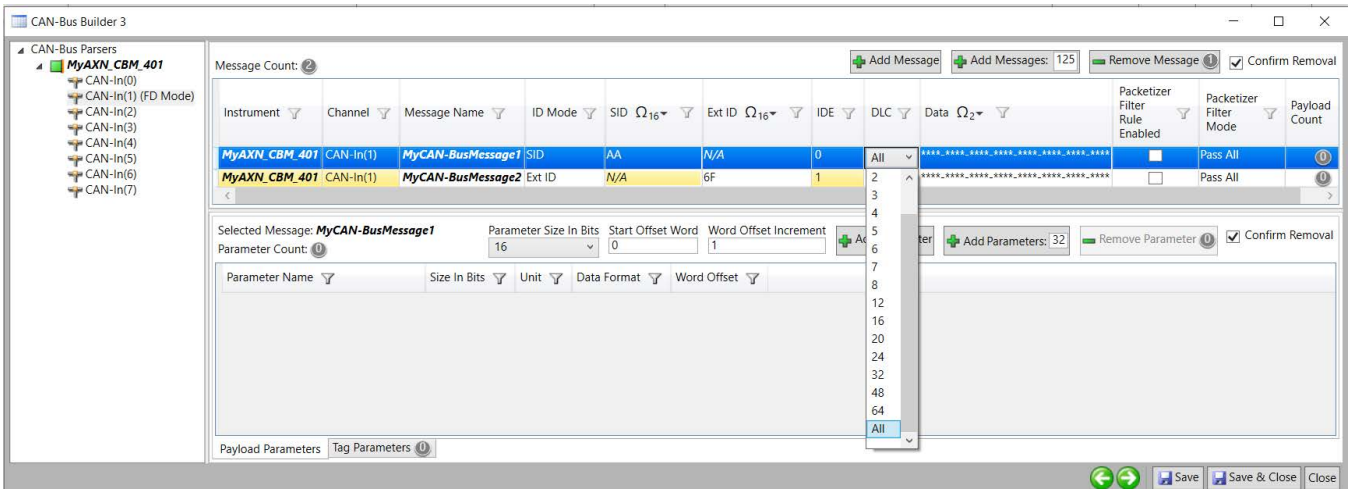


Figure 64-9: Example of the setup of each message type available in the AXN/CBM/401 in FD mode

64.5.3 Packetizer rule per message

The module supports filtered packetizer per channel. To configure the messages you want to filter, you need to enable both fields below:

Select the **Packetizer Filter Rule Enabled** check box to allow the message to be either Blocked or Passed in the packetizer (depending on the channel **Packetizer Filter Mode** set in the Settings tab) when they meet the packetizer filtering condition referenced by this process.

NOTE: **Packetizer Filter Rule Enabled** and **Packetizer Filter Mode** are only applicable when Packetization is enabled.

Packetizer Filter Mode can then be specified:

- **Block By Rule** means that messages that match defined filter rules are blocked and all other messages are passed through in the packetizer.
- **Pass By Rule** means that messages that match defined filter rules are passed through and all other messages are blocked in the packetizer.
- **Pass All** means that all messages are passed through the packetizer.

Packetizer Filter Mode is a channel setting, and this field must be set to the same rules in both the Settings tab and CAN-Bus Builder. If there is a discrepancy between CAN-Bus Builder and the channel settings, DAS Studio automatically changes Packetizer Filter Mode for all the messages on the same channel and in the Settings tab as shown in the following example. This prevents user error.

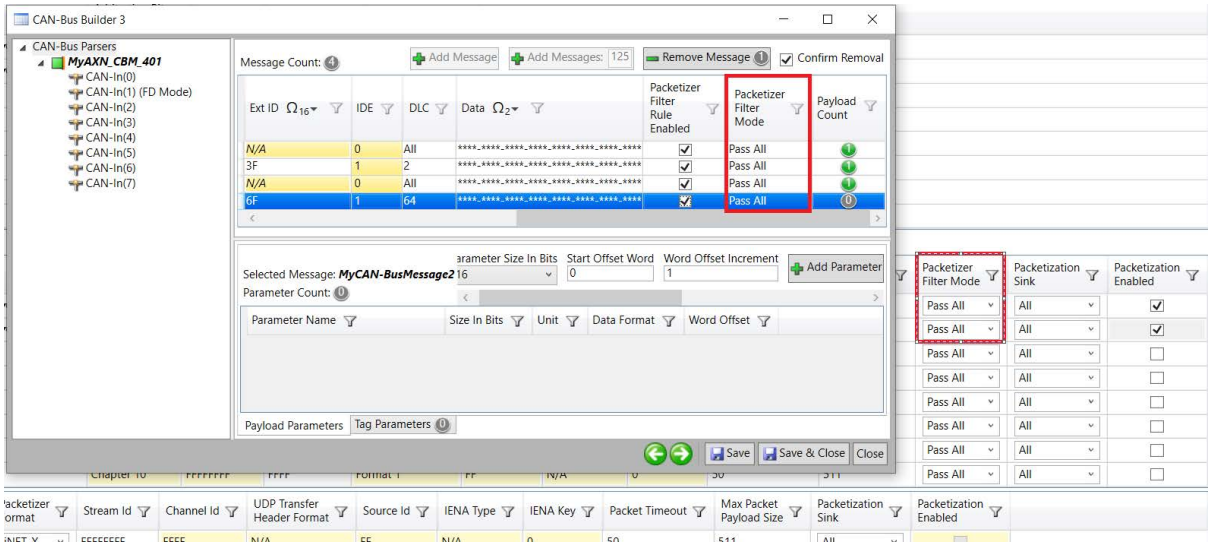


Figure 64-10: DAS Studio AXN/CBM/401 settings and CAN-Bus Builder showing the Packetizer Filter Mode settings propagation.

64.5.4 Adding parameters and tags

To add parameters to a message, type the number of parameters in the **Add Parameters** field and then click **Add**.

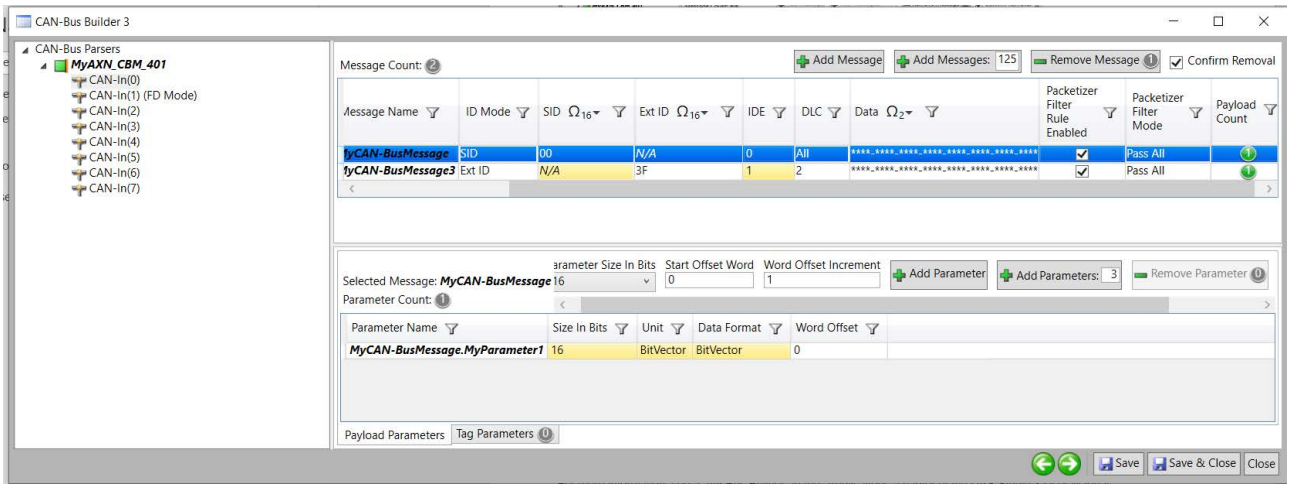
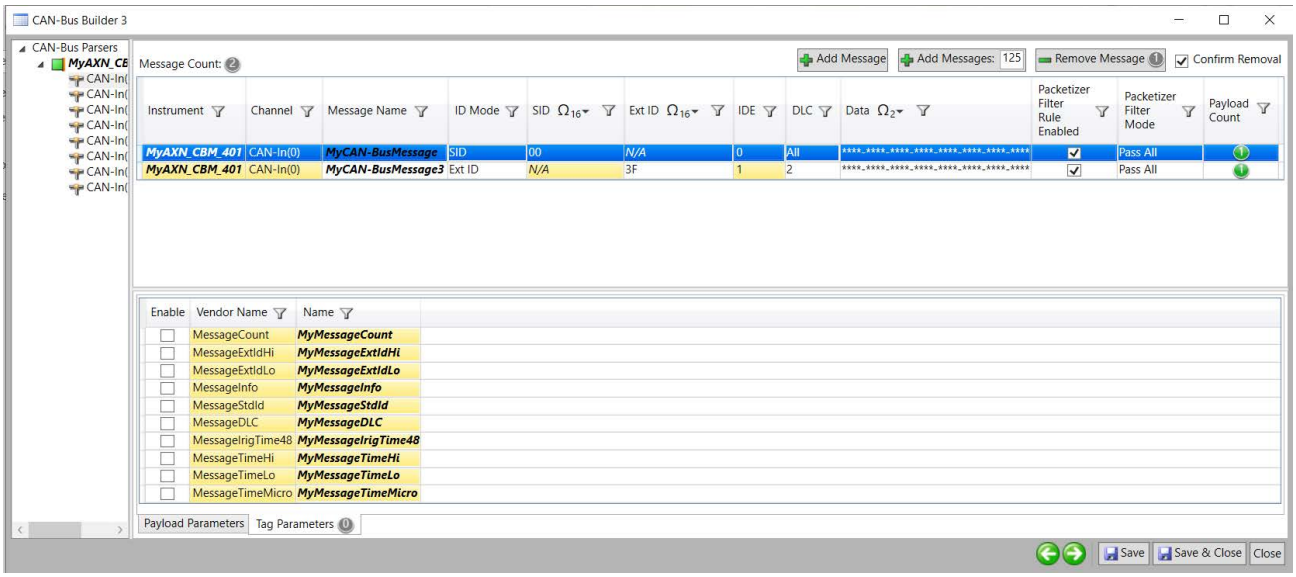


Figure 64-11: DAS Studio AXN/CBM/401 showing a parameter added to a message

For more information, see “CAN-Bus Builder” in the “Applications” chapter of the *DAS Studio 3 User Manual*.

To tag a message, select the message and then click the **Tag Parameters** tab.



The tags associated with the message are described as follows:

Settings	Description
MessageIrigTime48	MessageIrigTime48 is a 48-bit register consisting of three 16-bit time registers: TimeHi, TimeLo, and TimeMicro. MessageIrigTime48 represents the time stamp of a valid parsed message.
TimeHi, TimeLo and TimeMicro	Same information as MessageIrigTime48 but split in three 16-bit registers. Note: these registers are implemented to provide compatibility with legacy systems.
MessageCount	Counter of the received messages, however this counter might not increase smoothly. As explained in section “64.4.3 Global parameters settings” on page 5, the ModuleMessageCount is the only counter in the module counting messages received in all channels of the module. ModuleMessageCount stores the current value of that counter in the slot with each message, therefore ModuleMessageCount increases smoothly. MessageCount indicates where, in the arrival order, the particular message was received. For this reason consecutive messages on a single bus may not have consecutive values as messages on other channels may have been received, however it increases. Individual MessageCount from different channels can be correlated with ModuleMessageCount.
MessageInfo	Stale/skipped indication for each parsed message. These flags indicate whether messages are repeated or lost, that is, oversampling or undersampling situations respectively.

Refer to the *AXN/CBM/401* data sheet for further information on these tags and regarding additional tags such as MessageExtID, MessageStdID, and MessageDLC (which are not explained above).

To save your changes and close CAN-Bus Builder, click **Save & Close**.

When message building in CAN-Bus Builder is complete, these parameters become available to be placed in a PCM stream or placed packet.

64.6 Packetizer operation

Independently of the parser, when packetizer is enabled, a packet stream is generated for each channel. For iNET-X, all received bytes are encapsulated in an iNET-X parser-aligned payload structure. These parser-aligned packets may be transmitted aperiodically to optimize network bandwidth utilization and memory usage when recording traffic.

There are many settings available to configure or tune packetizer behavior.

The packetizer settings are shown in the following figure and described in the table that follows.

Packetizer

Source Name	Packetizer Format	Stream Id	Channel Id	UDP Transfer Header Format	Source Id	IENA Type	IENA Key	Packet Timeout	Max Packet Payload Size	Packetizer Filter Mode	Packetization Sink	Packetization Enabled
CAN-In(0)	iNET-X	FFFFFFFF	FFFF	N/A	FF	N/A	0	50	511	Pass All	All	☐
CAN-In(1)	iNET-X	FFFFFFFF	FFFF	N/A	FF	N/A	0	50	511	Pass All	All	☐
CAN-In(2)	iNET-X	FFFFFFFF	FFFF	N/A	FF	N/A	0	50	511	Pass All	All	☐
CAN-In(3)	iNET-X	FFFFFFFF	FFFF	N/A	FF	N/A	0	50	511	Pass All	All	☐
CAN-In(4)	iNET-X	FFFFFFFF	FFFF	N/A	FF	N/A	0	50	511	Pass All	All	☐
CAN-In(5)	iNET-X	FFFFFFFF	FFFF	N/A	FF	N/A	0	50	511	Pass All	All	☐
CAN-In(6)	iNET-X	FFFFFFFF	FFFF	N/A	FF	N/A	0	50	511	Pass All	All	☐
CAN-In(7)	iNET-X	FFFFFFFF	FFFF	N/A	FF	N/A	0	50	511	Pass All	All	☐

Figure 64-12: Packetizer settings

Settings	Description
Source Name	Channel to packetize.
Packetizer Format	iNET-X, IENA or Chapter 10. This setting defines the packetizer format for all channels.
Stream ID	iNET-X stream identifier for selected channel.
Channel ID	Chapter 10 channel ID for selected channel.
UDP Transfer Header Format	UDP transfer header format used to wrap Chapter 10 packets for streaming.
SourceID	Source Id used when streaming Chapter 10 packets in UDP transfer header format 3.
IENA Type	Q or M. Describes the IENA parameter type of the packet payload.
IENA Key	IENA Key for selected channel.
Packet Timeout	The timeout in milliseconds before a packet is generated if insufficient messages have been received to reach the Packet Size. Packets generated due to Packet Timeout are tagged in the iNET-X header. The Packet Timeout ranges from 10 ms to 999 ms (default value is 50 ms). Reducing this value results in more frequent and generally smaller packets. Increasing the value results in less frequent, but generally bigger packets.
Max Packet Payload Size	The number of words in the packet buffer, ranges from 200 words to 511 words. The default value is 511 words; reducing this value results in smaller and therefore generally more frequent packets.
Packetization Filter Mode	Specifies the filtering mode for the channel. **Block By Rule** means that messages that match defined filter rules are blocked and all other messages are passed through. **Pass By Rule** means that messages that match defined filter rules are passed through and all other messages are blocked. **Pass All** means that all messages are passed through. This is only applicable when Packetization is enabled.
Packetization Sink	Selects which modules the packetizer package is sent to for transmission or storage. The choices are Controller only, All slots or Slot in which a sink module that supports packetizer logging.

Settings	Description
Packetization Enabled	Enables the generation of a stream of packets containing messages from this channel. DAS Studio 3 automatically creates a packetizer packet after verification/programming.

For further information regarding iNET-X Placed packets used by the packetizer refer to *TEC/NOT/067 - IENA and iNET-X packet payload formats*. Additionally the *AXN/CBM/401* data sheet provides several examples of packetizer parser blocks.

The *AXN/CBM/401* data sheet explains the IENA Q/M and Chapter 10 packetizer format generated by the module.

NOTE: IADS does not currently support this Chapter 10 format nor IENA Q/M.

64.7 Enabling packetizer

To turn on packetizer operation on any channel, define a unique stream ID for that channel and then select the Packetization Enabled check box for that channel as shown in the following figure. The packetizer is enabled the next time the module is programmed.

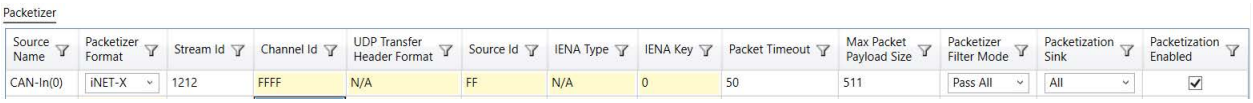


Figure 64-13: Packetization Enabled setting

NOTE: DAS Studio automatically creates packetizer packets on the aperiodic transmitter (such as the AXN/BCU/402). The packet rate is always N/A. This value is not used by the Axon hardware.

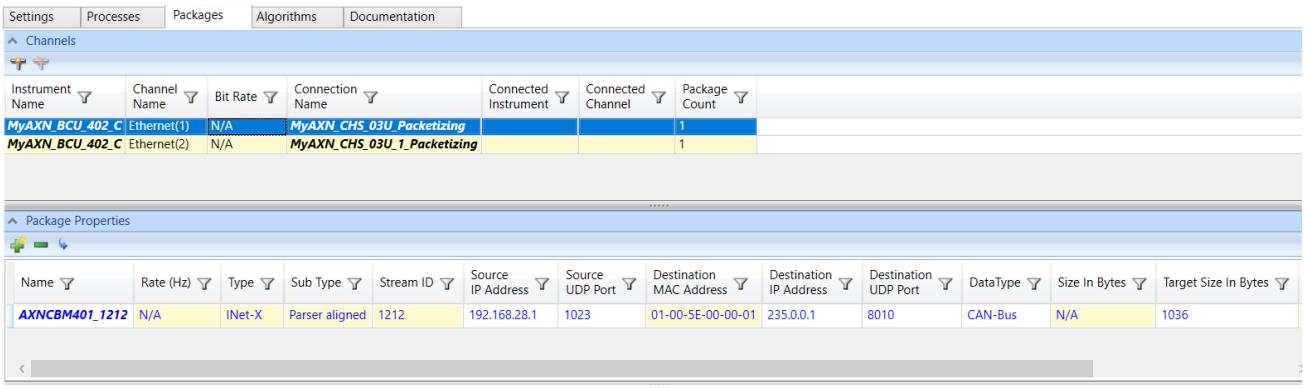


Figure 64-14: AXN/BCU/402 packages showing a packetizer created by DAS Studio after verification/programming

64.8 Appendix

64.8.1 Termination

For a CAN bus, the transmission line should be terminated at the last transceiver on the line (bus). Transmission line termination schemes is a topic of discussion beyond the scope of this technical note. The section gives a basic overview of the termination schemes.

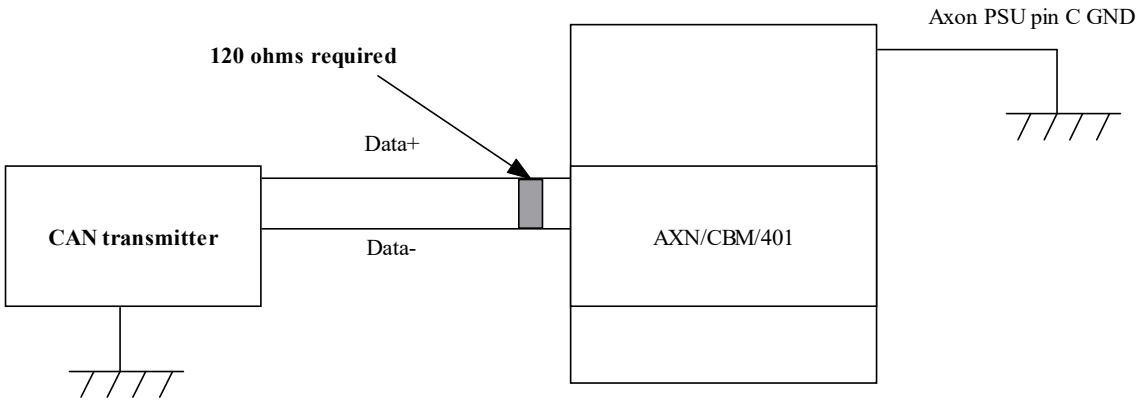


Figure 64-15: Example of termination required on the AXN/CBM/401

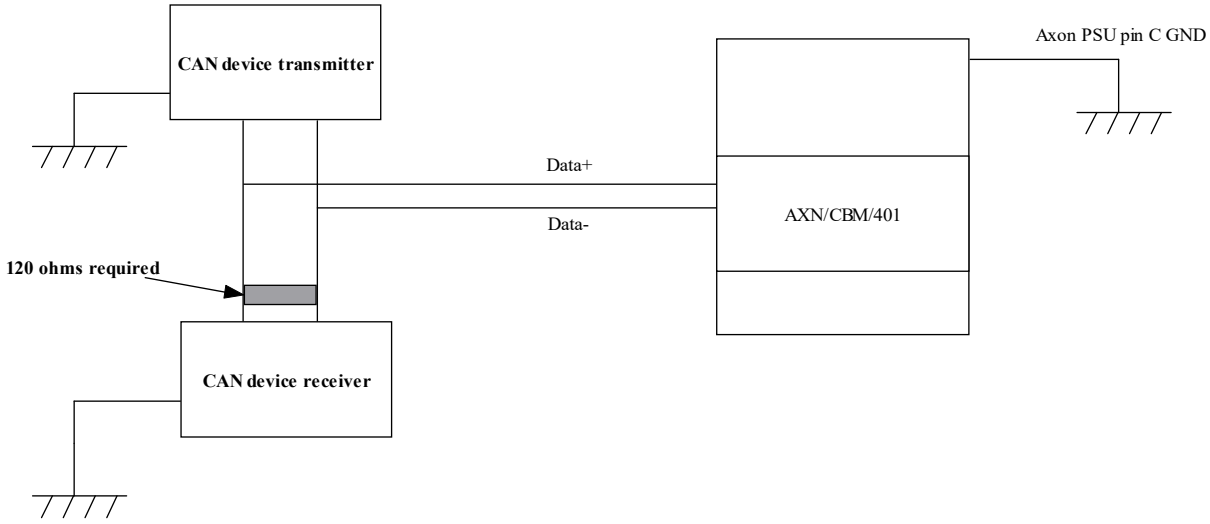


Figure 64-16: Example of no termination required on the AXN/CBM/401

The AXN/CBM/401 has an optional CAN pin termination. Refer to the *AXN/CBM/401* data sheet for further details.

64.8.2 Grounding

Grounding schemes is a topic of discussion beyond the scope of this technical note. The previous two figures show a system configuration presented without a separate ground wire. The basic rules for electronic circuits still require a clean ground connection to ensure error-free communication between drivers (Tx) and receivers (Rx). For further information refer to *TEC/NOT/063 — Grounding and shielding of the Axon and Acra KAM-500*.

64.8.3 ACK: acknowledgment

The AXN/CBM/401 is silent and only monitors however it can detect the post-message acknowledgment bit is present. If an acknowledgment bit is required to be transmitted, the AXN/CBM/402 is capable of transmitting the acknowledgment bit when it receives a valid message.

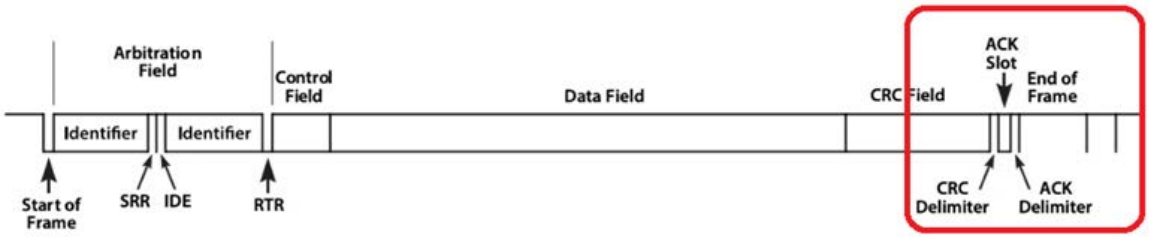
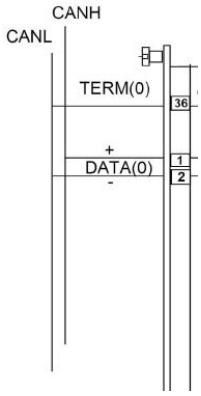


Figure 64-17: ACK slot shown where the bit is located

64.8.4 Troubleshooting

1. Ensure CANH is correctly wired to DATA+ and CANL is correctly wired to DATA- as shown in the following figure.



2. Ensure the baud rate is set appropriately.
3. Determine if termination is required or not (see “64.8.1 Termination” on page 12).
4. Ensure the CAN bus ground is connected to the Axon ground (GND). See *TEC/NOT/063 - Grounding and shielding of the Axon and Acra KAM-500*.
5. If some messages are not parsed as expected, it might be due to ack bit not being present for these nodes. Confirm if adding another node or using the AXN/CBM/402 to transmit the ACK bit resolves this issue.

64.8.5 Recommended reading

To better understand this paper, read the following documents.

Table 64-3: Data sheets

Document	Description
AXN/CBM/401	CAN bus monitor parser/packetizer - 8ch
AXN/CBM/402	CAN bus monitor parser/packetizer with acknowledgement support - 8ch

Table 64-4: Technical notes

Document	Description
TEC/NOT/063	Grounding and shielding of the Axon and Acra KAM-500
TEC/NOT/067	IENA and iNET-X packet payload formats

Table 64-5: User manual

Document	Description
DOC/MAN/030	DAS Studio 3 User Manual