Ethernet to CAN Converter Communication Protocol

Document version: 5
Date: December 14, 2022
Written by: Oleksandr Bogush, Axiomatic Technologies Corporation.

Contents

Introduction

The following document describes a proprietary communication protocol used by Axiomatic Ethernet to CAN and WiFi to CAN converters.

The protocol is used to transmit CAN messages over a TCP/IP network. In addition to CAN messages, the protocol also defines control and status messages necessary to communicate with the converter.

The document contains terminology and acronyms from CAN and TCP/IP protocols. Their meaning is explained in the appropriate CAN and TCP/IP documentation.

The document version contains a number and an optional letter. The number reflects changes in the actual protocol (all changes must be backward compatible), and the letter is used to change the protocol description.

Protocol Basics

The protocol is binary based. It runs on top of the standard UDP or TCP internet protocols. It uses a protocol independent message structure described below. This structure is also implemented in other Axiomatic proprietary communication protocols [1].

Protocol Message Structure

All protocol messages use the following protocol message structure, see Figure 1:

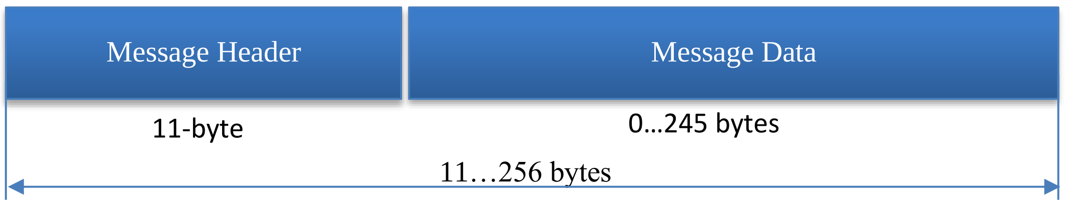


Figure 1. Protocol Message Structure

The protocol message contains:

- 11-byte *Message Header*;
- Variable size *Message Data* field, from 0 up to 245 bytes.

The overall size of the protocol message is limited to 256 bytes to protect the messages from fragmentation during transmission over the internet and to simplify handling in embedded systems with limited RAM resources.

The protocol messages are transmitted in the ascending octet order. All numerical values in all message fields, unless explicitly stated, are presented least significant byte (LSB) first.

Message Header

The protocol *Message Header* contains:

- 4-byte *Axiomatic Tag,* AXIO in capital letters;
- 2-byte *Protocol ID*;
- 2-byte *Message ID*;
- 1-byte *Message Version* (0 by default);
- 2-byte *Message Data Length.*

The protocol *Message Header* format is presented below:

Table 1. Protocol Message Header Format

Octet / Offset Octet	0	1	2	3
0	A	X	I	O
	0x41	0x58	0x49	0x4F
	Axiomatic Tag			
4	Protocol ID		Message ID	
8	Message Version (0 by default)	Message Data Length		Message Data (optional, if Message Data Length > 0)

The *Axiomatic Tag* is used for the message header identification.

The *Protocol ID* defines a proprietary protocol carried by this message. This field allows different protocols to use the same protocol independent message structure. The *Protocol ID* equal, for example, 0x36BA, is presented as: (0xBA, 0x36) – LSB first and the most significant byte (MSB) second:

Table 2. Protocol ID Presentation

Octet / Offset Octet	0	1	2	3
4	0xBA	0x36	Message ID	
	Protocol ID=14010=0x36BA			

The *Message ID* defines a type of the message within the specified protocol and the *Message Data Length* – its length. The *Message Data Length* should be zero for messages without the *Message Data* field.

The *Message Version* field is used to distinguish between different message data formats in messages with the same *Message ID*. Different versions of the same message should have backward compatible data formats.

This protocol message header format allows parsing of the protocol messages without any knowledge of the message data content. Each *Message Data* field is then parsed by individual protocol-specific parsers based on: *Protocol ID, Message ID and Message Version* fields.

Message Data

The *Message Data* field format depends on the protocol and the message type defined in the *Message Header*.

Protocol ID

The proprietary communication protocol described in this document uses the *Protocol ID* = 14010 – the project number of the converter first used this protocol.

Message IDs

The following *Message IDs* are defined in the current version of the protocol.

Table 3. Message IDs.

Message ID	Message Versions	Message Name
0	0	Undefined Message
1	0	CAN and Notification Stream. Deprecated
2	0	Status Request
3	0,1,2	Status Response
4	0,1,2	Heartbeat
5	0	CAN FD Stream

The *Undefined Message* has no parser associated with it. Messages with IDs not shown in Table 3 are not processed by the current version of the protocol. They are treated the same way, as *Undefined Messages*.

Connection Lifetime

The connection lifetime for a pair of nodes communicating using the proprietary communication protocol depends on the upper-level IP protocol and the message traffic between the nodes.

TCP Protocol

If the TCP protocol is used, the connection is maintained by the standard means of this protocol.

UDP Protocol

If the UDP protocol is used, the connection is considered to be lost after 10 seconds of inactivity on one of the nodes. Inactivity here means no protocol messages for a certain period of time.

To avoid disconnection on inactivity, it is recommended that the node communicating with the converter constantly send a *Heartbeat* or a *Status Request* message.

Heartbeat Message

The *Heartbeat* message is preferable, since it does not require a response message from the converter and it has a standard sending interval defined in the Heartbeat section of this document. The *Heartbeat* message with an undefined (all zeros) *Health Data* field can be potentially used for monitoring quality of the connection on the converter side in the future extensions of the protocol, if all other fields including the *Message Number* and the *Time Interval* are set.

A blank *Heartbeat* message with all data fields set to zeros is also acceptable, but only for maintaining a connection with the converter. A converter itself cannot use such a message; it must define all the message data fields.

Status Request

If the *Status Request* message is used to maintain a connection between nodes, it must not be sent more often than the *Heartbeat* message.

CAN and Notification Stream. Deprecated[1]

The *CAN and Notification Stream* is the main old message type (before being deprecated) used by the Ethernet to CAN converter. It encodes CAN messages. In addition to CAN messages, it can be used to send short notification messages. The *CAN and Notification Stream* has *Message ID* = 1.

The notification messages are not defined in the current version of the protocol. This feature is left for the future use.

Message Data Structure

Each *CAN and Notification Stream* message consists of *CAN* and *Notification Frames* following each other in an arbitrary order. For example:

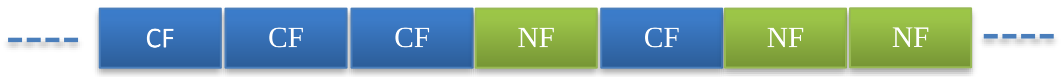


Figure 2. CAN and Notification Stream Example

CAN Frames (CF) are used to transmit various types of CAN messages. *Notification Frames* (NF) are intended to communicate status of the CAN interface. They also can be used to send run-time error information, etc.

CAN Frames

CAN Frames have the following format:

$$\mathbf{CF} = \{CB,[TSB_1,\ldots,TSB_4],IDB_1,IDB_2,[IDB_3,IDB_4],[DB_1,\ldots,DB_8]\} \quad (1)$$

Where:

CB – *Control Byte*;

$TSB_1,\ldots,TSB_4$ – Optional one to four bytes of the *Time Stamp*, LSB first;

$IDB_1,IDB_2,[IDB_3,IDB_4]$ – Two or four byte *CAN ID* with *Remote Frame* bit, LSB first;

[1] Use CAN FD Stream in the new design.

DB$_1$,…,DB$_8$ – Optional up to eight *CAN Data* bytes.

Due to a variety of CAN message types and different length of the timestamp, the length of the *CAN Frame* is variable. It is determined by the information in the first *Control Byte* (CB) of the frame.

Control Byte

Control Byte (CB) of the *CAN Frame* contains the following bits:

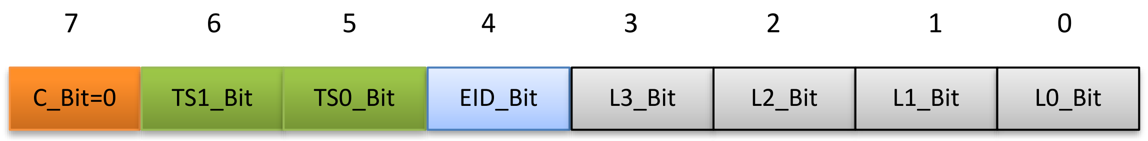


Figure 3. CAN Frame. Control Byte

Bit 7 = C_Bit: *Control Bit*
0: *CAN Frame*.
1: *Notification Frame*.
This bit defines a type of the frame. It should be always 0 for *CAN Frames*.

Bit 6:5 = TS_Bit[1:0]: *Time Stamp Length Bits*
Refer to Table 4 for the TS_Bit settings.

Table 4. CAN Frame. Time Stamp Length Bits

TS1_Bit	TS0_Bit	Time Stamp Length in bytes
0	0	0 – No Time Stamp
0	1	1
1	0	2
1	1	4

Bit 4 = EID_Bit: *Extended CAN ID Bit*
0: CAN ID is standard
1: CAN ID is extended

Bit 3:0 = L_Bit[3:0] : *CAN Data Length Bits*
Refer to Table 5 for L_Bit settings.

Table 5. CAN Frame. CAN Data Length Bits

L3_Bit	L2_Bit	L1_Bit	L0_Bit	Number of Data bytes
0	0	0	0	0
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7

L3_Bit	L2_Bit	L1_Bit	L0_Bit	Number of Data bytes
1	0	0	0	8
1	0	0	1	Undefined
1	-	-	-	
1	1	1	1	

Time Stamp

An optional *Time Stamp* (TS) carries a time interval between the current and the previous CAN messages received on the CAN bus. It is filled by the Ethernet to CAN converter for incoming CAN messages. This field is not used for encoding outcoming CAN messages sent to the Ethernet to CAN converter by external nodes.

The *Time Stamp* is measured in milliseconds and can be: 0, 1, 2 or 4 byte long depending on the TS_Bit value in the *Control Byte*.

For 1-byte *Time Stamp*:
Bit 0:7 in TSB$_1$ = TS_Bit[7:0] : *1-byte Time Stamp*

7	6	5	4	3	2	1	0	
TS7_Bit	TS6_Bit	TS5_Bit	TS4_Bit	TS3_Bit	TS2_Bit	TS1_Bit	TS0_Bit	TSB$_1$

Figure 4. CAN Frame. One-byte Time Stamp

For 2-byte *Time Stamp*:
Bit 0:7 in TSB$_1$, Bit 0:7 in TSB$_2$ = TS_Bit[15:0] : *2-byte Time Stamp*

7	6	5	4	3	2	1	0	
TS7_Bit	TS6_Bit	TS5_Bit	TS4_Bit	TS3_Bit	TS2_Bit	TS1_Bit	TS0_Bit	TSB$_1$

7	6	5	4	3	2	1	0	
TS15_Bit	TS14_Bit	TS13_Bit	TS12_Bit	TS11_Bit	TS10_Bit	TS9_Bit	TS8_Bit	TSB$_2$

Figure 5. CAN Frame. Two-byte Time Stamp

For 4-byte *Time Stamp*:
Bit 0:7 in TSB$_1$, Bit 0:7 in TSB$_2$, Bit 0:7 in TSB$_3$, Bit 0:7 in TSB$_4$ = TS_Bit[31:0] : *4-byte Time Stamp*

7	6	5	4	3	2	1	0	
TS7_Bit	TS6_Bit	TS5_Bit	TS4_Bit	TS3_Bit	TS2_Bit	TS1_Bit	TS0_Bit	TSB$_1$
TS15_Bit	TS14_Bit	TS13_Bit	TS12_Bit	TS11_Bit	TS10_Bit	TS9_Bit	TS8_Bit	TSB$_2$
TS23_Bit	TS22_Bit	TS21_Bit	TS20_Bit	TS19_Bit	TS18_Bit	TS17_Bit	TS16_Bit	TSB$_3$
TS31_Bit	TS30_Bit	TS29_Bit	TS28_Bit	TS27_Bit	TS26_Bit	TS25_Bit	TS24_Bit	TSB$_4$

Figure 6. CAN Frame. Four-byte Time Stamp

CAN Identifier

CAN Identifier (*CAN ID*) structure is different for *Standard* and *Extended CAN ID*.

Standard CAN ID

7	6	5	4	3	2	1	0	
ID7_Bit	ID6_Bit	ID5_Bit	ID4_Bit	ID3_Bit	ID2_Bit	ID1_Bit	ID0_Bit	IDB$_1$
RF_Bit	Reserved	Reserved	Reserved	Reserved	ID10_Bit	ID9_Bit	ID8_Bit	IDB$_2$

Figure 7. CAN Frame. Standard CAN ID.

Bit 0:7 in IDB$_1$, Bit 0:2 in IDB$_2$ = ID_Bit[10:0] : *CAN Standard Identifier*

Bit 3:6 in IDB$_2$ = Reserved.

Bit 7 in IDB$_2$ = RF_Bit : *Remote Frame Bit*
0: *CAN Frame* is a regular data frame
1: *CAN Frame* is a remote request for a data frame

Extended CAN ID

7	6	5	4	3	2	1	0	
ID7_Bit	ID6_Bit	ID5_Bit	ID4_Bit	ID3_Bit	ID2_Bit	ID1_Bit	ID0_Bit	IDB_1
7	6	5	4	3	2	1	0	
ID15_Bit	ID14_Bit	ID13_Bit	ID12_Bit	ID11_Bit	ID10_Bit	ID9_Bit	ID8_Bit	IDB_2
7	6	5	4	3	2	1	0	
ID23_Bit	ID22_Bit	ID21_Bit	ID20_Bit	ID19_Bit	ID18_Bit	ID17_Bit	ID16_Bit	IDB_3
7	6	5	4	3	2	1	0	
RF_Bit	Reserved	Reserved	ID28_Bit	ID27_Bit	ID26_Bit	ID25_Bit	ID24_Bit	IDB_4

Figure 8. CAN Frame. Extended CAN ID

Bit 0:7 in IDB_1, IDB_2, IDB_3 and Bit 0:4 in IDB_4 = ID_Bit[28:0] : *CAN Extended Identifier*

Bit 5:6 in IDB_4 = Reserved

Bit 7 in IDB_4 = RF_Bit : *Remote Frame Bit*
0: *CAN Frame* is a regular data frame
1: *CAN Frame* is a remote request for a data frame

CAN Data Bytes

Optional *CAN Data* bytes are placed after *CAN ID* in the same order they appear in the CAN message. The number of *CAN Data* bytes is specified by the L_Bit field in the *Control Byte* (CB).

Notification Frames

The format of the *Notification Frames* is presented below:

$$\mathbf{NF} = \{NIDB, NDB_1, \ldots, NDB_4\} \quad (2)$$

Where:

NIDB – *Notification Identifier* byte;

$NDB_1, \ldots, NDB_4$ – *Notification Data* bytes.

In opposite to the variable size *CAN Frames*, *Notification Frames* have a 5-byte fixed size format. *Notification Identifier* byte (NIDB) carries the *Notification Identifier*, which defines information sent by the *Notification Frame* and the meaning of the four *Notification Data* bytes (NDB) associated with the frame.

Notification Identifier

The *Notification Identifier* byte (NIDB) has the following format:

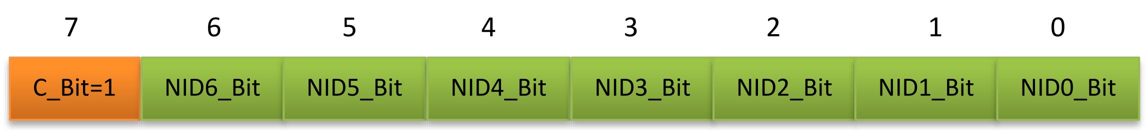


Figure 9. Notification Frame. Notification ID Byte

Bit 7 = C_Bit: *Control Bit*
0: *CAN Frame*.
1: *Notification Frame*.
This bit defines a type of the frame. It should be always 1 for *Notification Frames*.

Bit 0:6 = NID_Bit[6:0] : *Notification Identifier*
Seven bits of the *Notification Identifier* can determine up to 128 different notification messages.

Notification Data

The four *Notification Data* bytes carry information defined by the *Notification Identifier.*

There are no notification messages supported by the current version of the protocol.

CAN FD Stream

The *CAN FD Stream* is the new main message type used by the Ethernet to CAN converter. It encodes CAN and CAN FD messages. In addition to CAN messages, it can be used to send error messages or any type of notification messages. The *CAN FD Stream* has *Message ID* = 5.

A node reports the support of *CAN FD Stream* by setting the *CAN FD Stream Flag* in the *Supported Features* field of the Status Response and Heartbeat messages.

Message Data Structure

Each *CAN FD Stream* message consists of one or several *CAN FD Frames*:

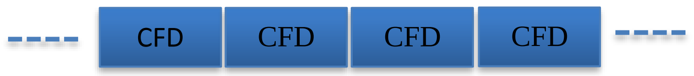


Figure 10. CAN FD Stream

CAN FD Frames (CFD) are used to transmit CAN FD or Classical CAN frames together with the frame routing data and an absolute timestamp to a receiving node. Adding frame routing data, presented by *CAN Frame Address*, allows sending the CAN frame to up to 32 CAN ports simultaneously on the receiving node.

When a special *Error Message Flag* is set, *CAN FD Frames* convey Error or Notification Messages.

It is possible to have only one *CAN FD Frame* in the *CAN FD Stream* message when a receiving node requests this feature.

CAN Frame Routing

This protocol supports CAN frame routing to different CAN ports on the same IP device. This is achieved by using a special routing address that uniquely identifies CAN ports on the device. To allow simultaneously sending CAN data to different CAN ports, this address is designed the way that it can include a set of individual CAN port IDs.

The CAN routing address is presented by two parameters: *Channel Group (CG)* and *Channel ID Set (CIDS)*: (CG, CIDS). The *Channel Group* defines a set of channels in the *Channel ID Set.* The *Channel ID Set* selects channels, or CAN ports, that will be addressed from this *Channel Group*.

Each Channel ID is presented as a bit (flag) at the channel number position in the *Channel ID Set* field. The *Channel ID Set* can contain up to 32 Channel IDs that are ORed together in the 4-byte *Channel ID Set* field. This allows sending a CAN frame simultaneously to up to 32 CAN ports.

Each CAN port has its own routing address, *CAN Port Address,* on the IP device*.* This address is included as *CAN Frame Address* in *CAN FD Frames* sent by the converter through the Ethernet port.

When the *CAN Frame Address* matches the address of a CAN port on another device, the port receives the transmitted CAN frame. This way, CAN ports on different IP devices can be virtually connected together.

Two addresses are considered to match if their *Channel Groups* are equal and there is at least one Channel ID in the *Channel ID Sets* that is selected (set to 1) in both addresses.

The converter CAN ports transmit CAN data with only one CAN port Channel ID in the *CAN Frame Address*. However, it is possible to transmit CAN data with multiple Channel IDs in the *Channel ID Set* field of the *CAN Frame Address* if it is necessary to broadcast CAN data to multiple CAN ports.

It is assumed that Axiomatic CAN converters and PC software supporting only *CAN and Notification Stream* messages (that do not have CAN data routing) are using (CG=0, CIDS=1) combination for their CAN routing address.

The all zero CAN routing address (CG=0, CIDS=0) has a special meaning of undefined address. It should not be used in *CAN FD Frames* but can be used for frame filtering in device nodes, see *CAN Port Input Filter Address* in *Status Response* and *Heartbeat* messages.

CAN routing addresses with non-zero *Channel Group* field and zero *Channel ID Set* field (CG≠0, CIDS=0) do not select any CAN ports and are considered invalid.

CAN FD Frame

CAN FD Frame has the following format:

CFDF = {PCNFB$_1$, PCNFB$_2$, CGB, CIDSB$_1$,…,CIDSB$_4$, ATB$_1$,…, ATB$_4$, CANFB, CANLB, CANID$_1$,…, CANID$_4$, [DB$_1$,…,DB$_{64}$]}

(3)

Where:

PCNFB$_1$, PCNFB$_2$ – 2-byte *Physical Channel Number & Flags*, LSB first;
CGB – 1-byte *Channel Group.* First part of the *CAN Frame Address*;
CIDSB$_1$,…,CIDSB$_4$ – 4-byte *Channel ID Set*, LSB first. Second part of the *CAN Frame Address*;
ATB$_1$,…, ATB$_4$ – 4-byte *Absolute Timestamp* in milliseconds, LSB first;
CANFB – 1-byte *CAN Flags*;
CANLB – 1-byte *CAN Length*;
CANIDB$_1$,…, CANIDB$_4$ – 4-byte CAN Identifier, LSB first;
DB$_1$,…,DB$_{64}$ – Optional up to 64 *Data Bytes* (up to 8 bytes for Classical CAN and up to 64 bytes for CAN FD).

The position of all data fields in the frame is fixed. This allows building a simple parser that extracts *CAN FD Frame* data from communication protocol messages based on the absolute position of the *CAN FD Frame* data fields in the *CAN FD Stream* message. The *One Frame per Message in CAN FD Stream Flag* should be set in this case in the *Supported Features* field of the *Status Response* or *Heartbeat* messages to avoid multiple *CAN FD Frames* being sent to the node in one *CAN FD Stream* message.

Physical Channel Number & Flags

The *Physical Channel Number & Flags* field is used to transmit a physical channel number (PCN) of the CAN port on the transmitting node. It also contains a three-bit *Flags* (F) field, see Figure 11. The *Flags* are reserved for the future use.

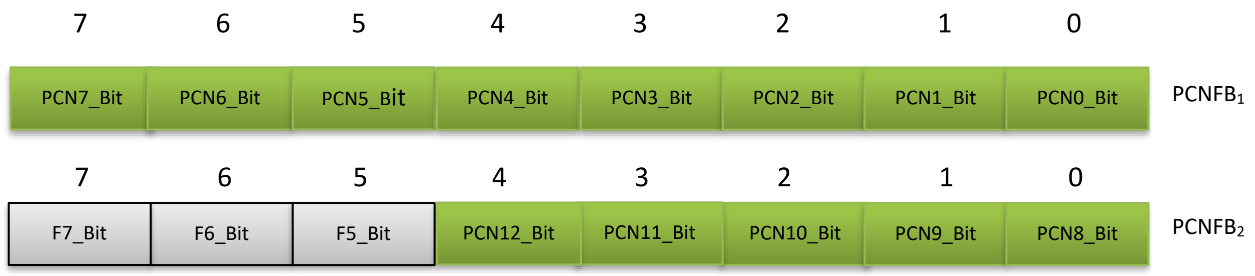

7 6 5 4 3 2 1 0
PCN7_Bit PCN6_Bit PCN5_Bit PCN4_Bit PCN3_Bit PCN2_Bit PCN1_Bit PCN0_Bit
PCNFB1
7 6 5 4 3 2 1 0
F7_Bit F6_Bit F5_Bit PCN12_Bit PCN11_Bit PCN10_Bit PCN9_Bit PCN8_Bit
PCNFB2

Figure 11. CAN FD Frame. Physical Channel Number & Flags

Bit 0:7 in PCNFB$_1$, Bit 0:5 in PCNFB$_2$ = PCN_Bit[12:0] : *Physical Channel Number;*
Bit 5:7 in PCNFB$_2$ = F_Bit[7:5]: *Flags. Flags = 0.* Reserved.

Physical Channel Number range is: [0…8191], 0 – if not used or channel is undefined.

In comparison with the *Channel Group* and *Channel IDs*, the *Physical Channel Number* contains the actual device CAN port number, not a virtual *CAN Frame Address*. It can be used, for example, for data logging purposes.

CAN Frame Address

The *CAN Frame Address* is a CAN routing address presented by two parameters: *Channel Group* (CG) and *Channel ID Set* (CIDS): (CG, CIDS). This is a virtual parameter that associates the *CAN FD Frame* with CAN ports on a receiving device.

Channel Group

The *Channel Group* (CG) is the first part of the *CAN Frame Address* (CG, CIDS). It can take any value in the [0…255] range. All values, including 0, can be used to define a valid *Channel Group*.

Channel ID Set

The *Channel ID Set* (CIDS) is the second part of the *CAN Frame Address* (CG, CIDS). It defines channels in the given *Channel Group*.

Absolute Timestamp

The *Absolute Timestamp* field contains a free running counter value with 1 millisecond resolution. The value is captured when the CAN frame is received from the CAN port. The *Absolute Timestamp* is set to 0, if not used.

CAN Flags

The *CAN Flags* field contains a set of flags for encoding CAN FD or Classical CAN frames, see Figure 12.

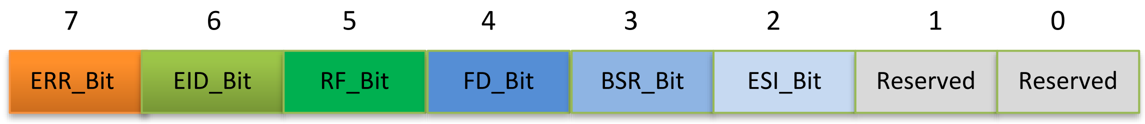


Figure 12. CAN FD Frame. CAN Flags

Bit 7 in CANFB = ERR_Bit: *Error Message Flag*
Bit 6 in CANFB = EID_Bit: *Extended ID Flag*
Bit 5 in CANFB = RF_Bit: *Remote Frame Flag*
Bit 4 in CANFB = FD_Bit: *CAN FD Frame Flag*
Bit 3 in CANFB = BSR_Bit: *CAN FD Bit Rate Switch (BRS) Flag*
Bit 2 in CANFB = ESI_Bit: *CAN FD Error Status Indicator (ESI) Flag*
Bit 1:0 in CANFB = 0 : Reserved.

The values of *CAN Flags* are presented in Table 6.

Table 6. CAN Flags

CAN Flag Name	Value
Error Message Flag	0x80
Extended ID Flag	0x40
Remote Frame Flag	0x20
CAN FD Frame Flag	0x10
CAN FD Bit Rate Switch (BRS) Flag	0x08
CAN FD Error Status Indicator (ESI) Flag	0x04

If the *Error Message Flag* is set, the CAN FD frame carries an Error or Notification message. In this case, the message data length is set in the *CAN Length* field, and the message data is conveyed in up to 64 *Data Bytes*.

The following Error Messages are defined in the current version of the protocol, see Table 7. No Notification Messages are currently defined.

Table 7. CAN Error Messages

CAN Error Message Name	CAN Data Length	Data Bytes, DB_1
CAN Error Undefined	1	0
CAN Warning	1	1
CAN Passive	1	2
CAN Bus-Off	1	3

CAN Length

The *CAN Length* (CANL) field contains the length of the CAN frame or the number of data bytes in the Error or Notification message. The *CAN Length* range is:

- [0,…,8] for Classical CAN Data Frames;
- [0,…,8, 12, 16, 20, 24, 32, 48, 64] – for CAN FD frames and Classical CAN Remote Frame Requests;
- [1,…,64] – for error and notification messages.

CAN Identifier

The *CAN Identifier* (CANID) field contains Standard or Extended CAN ID, see Figure 13.

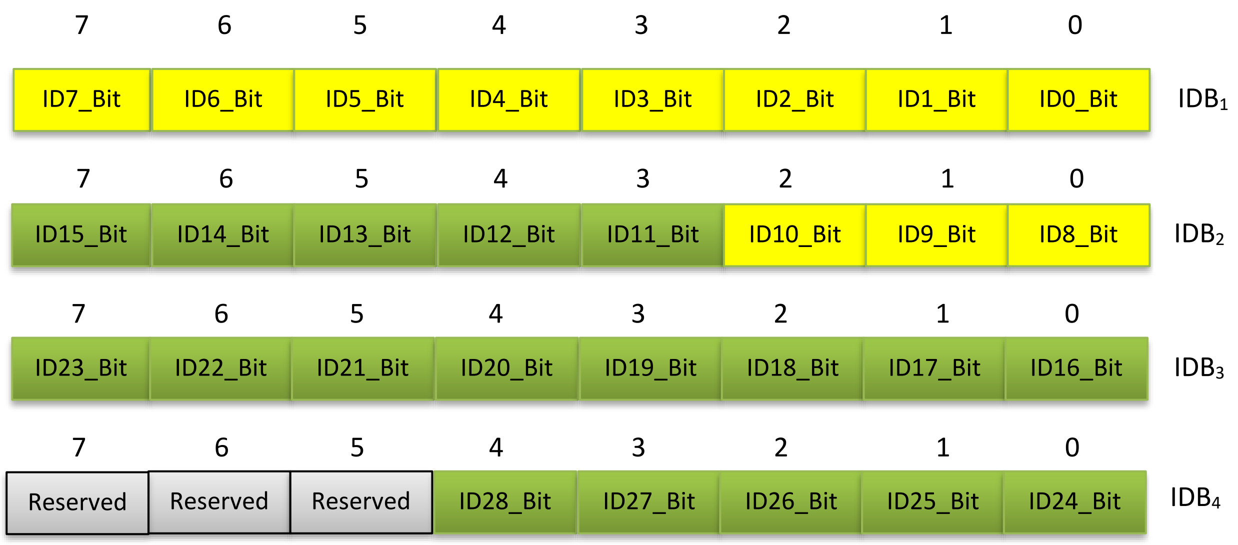

7 6 5 4 3 2 1 0
ID7_Bit ID6_Bit ID5_Bit ID4_Bit ID3_Bit ID2_Bit ID1_Bit ID0_Bit
IDB1
ID15_Bit ID14_Bit ID13_Bit ID12_Bit ID11_Bit ID10_Bit ID9_Bit ID8_Bit
IDB2
ID23_Bit ID22_Bit ID21_Bit ID20_Bit ID19_Bit ID18_Bit ID17_Bit ID16_Bit
IDB3
Reserved Reserved Reserved ID28_Bit ID27_Bit ID26_Bit ID25_Bit ID24_Bit
IDB4

Figure 13. CAN FD Frame. CAN Identifier

In case of Standard CAN ID:

- Bit 0:7 in IDB_1, Bit 0:2 in IDB_2 = ID_Bit[10:0] : *Standard CAN Identifier;*
- Bit 3:7 in IDB_2, Bit 0:7 in IDB_3, Bit 0:4 in IDB_4 = 0.

In case of Extended CAN ID:
Bit 0:7 in IDB_1, IDB_2, IDB_3 and Bit 0:4 in IDB_4 = ID_Bit[28:0] : *Extended CAN Identifier*

Bit 5:7 in IDB_4 = 0 : Reserved.

Data Bytes

The optional CAN *Data Bytes* are placed after *CAN Identifier* in the same order they appear in the Classical CAN Data or CAN FD message. The number of CAN *Data Bytes* is specified in the *CAN Length* field. No data is placed for Classical CAN Remote Frame Requests.

The number of *Data Bytes* and their content is defined individually for Error and Notification Messages.

Status Response

The *Status Response* message with *Message ID* = 3 is sent by the Ethernet to CAN converter in response to the *Stratus Request* message. The most recent *Status Response* Message Versions is 2.

Status Response Data Fields

The *Status Response* message sends the following data:

$$\mathbf{SRM} = \{HDB_1,\ldots, HDB_4, CANRxDEB_1,\ldots,CANRxDEB_4, CANTxDEB_1,\ldots,CANRxDEB_4, CANBusOffB_1,\ldots,CANBusOffB_4, CTB, SFB_1,\ldots, SFB_4, CGB, CIDSB_1,\ldots,CIDSB_4\} \quad (4)$$

Where:

$HDB_1,\ldots, HDB_4$ – 4-byte *Health Data*, LSB first;
$CANRxDEB_1,\ldots, CANRxDEB_4$ – 4-byte *CAN Receive Error Counter*, LSB first;
$CANTxDEB_1,\ldots, CANTxDEB_4$ – 4-byte *CAN Transmit Error Counter*, LSB first;
$CANBusOffB_1,\ldots, CANBusOffB_4$ – 4-byte *CAN Bus Off Counter*, LSB first;
CTB – 1-byte *Converter Type* (Defined only in *Message Version ≥ 1*);
$SFB_1,\ldots, SFB_4$ – 4-byte *Supported Features*, LSB first , (Defined only in *Message Version =2*) ;
CGB – 1-byte *Channel Group.* First part of the *CAN Port Input Filter Address.* (Defined only in *Message Version = 2*);
$CIDSB_1,\ldots,CIDSB_4$ – 4-byte *Channel ID Set*, LSB first. Second part of the *CAN Port Input Filter Address.* (Defined only in *Message Version = 2*).

Health Data

The 4-byte *Health Data* field contains the health status information of the Ethernet to CAN converter. It is described in [2].

Converter Type

The following *Converter Types* are defined in *Message Version ≥ 1*:

Table 8. Converter Types

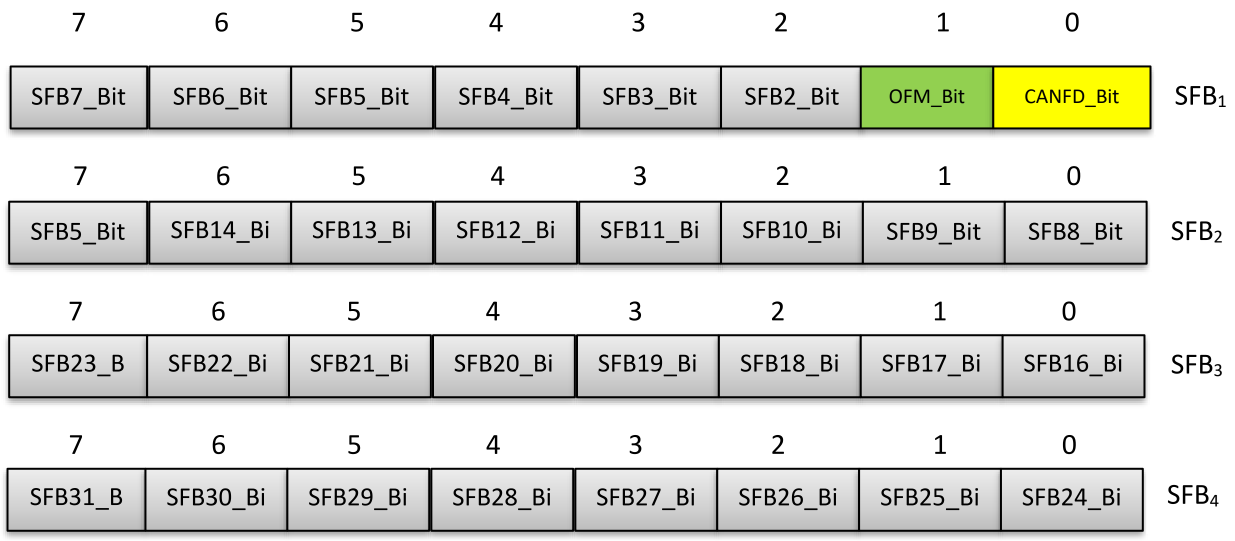
Converter Type	Name
0[1]	Ethernet to CAN converter with CAN Voltage Output

Converter Type	Name
1	WiFi to CAN converter
2	WiFi to CAN converter with CAN datalogging capability

[1]Default value if the *Converter Type* is not defined (e.g., in the *Message Version* = 0)

Supported Features

The *Supported Features* field, defined in Message Version = 2, contains Communication Protocol features that the node is supporting. The field contains flags individually marking each supported Communication Protocol feature, see Figure 14.



7 6 5 4 3 2 1 0
SFB7_Bit SFB6_Bit SFB5_Bit SFB4_Bit SFB3_Bit SFB2_Bit OFM_Bit CANFD_Bit
SFB1
SFB5_Bit SFB14_Bi SFB13_Bi SFB12_Bi SFB11_Bi SFB10_Bi SFB9_Bit SFB8_Bit
SFB2
SFB23_B SFB22_Bi SFB21_Bi SFB20_Bi SFB19_Bi SFB18_Bi SFB17_Bi SFB16_Bi
SFB3
SFB31_B SFB30_Bi SFB29_Bi SFB28_Bi SFB27_Bi SFB26_Bi SFB25_Bi SFB24_Bi
SFB4

Figure 14. Status Response. Supported Features

Bit 0 in SFB_1 =CANFD_Bit : *CAN FD Stream Flag*
Bit 1 in SFB_1 = OFM_Bit : *One Frame per Message in CAN FD Stream Flag*
Bit 2:7 in SFB_1, Bit 0:7 in SFB_2,…,SFB_4 = 0. Reserved.

The currently defined flag values are presented in Table 9.

Table 9. Supported Features

Name	Value
CAN FD Stream Flag	0x0001
One Frame per Message in CAN FD Stream Flag	0x0002

The *CAN FD Stream Flag* is set when the node supports *CAN FD Stream.*

The *One Frame per Message in CAN FD Stream Flag* requests all nodes to send only one *CAN FD Frame* in the *CAN FD Stream* message to the node. The *CAN FD Stream Flag* should be set when this flag is used.

CAN Port Input Filter Address

The *CAN Port Input Filter Address* defines a set of CAN ports on the node, which can process input *CAN FD Frames*. This information can reduce the network traffic to the node and overall system load since there is no need to send *CAN FD Frames* on the node that the node will not process.

The *CAN Port Input Filter Address* is a CAN routing address presented by two parameters: Channel Group (CG) and Channel ID Set (CIDS): (CG, CIDS).

If the *CAN Port Input Filter Address* is undefined: (CG, CIDS) = (0, 0), the CAN port input filter is disabled, and the node can process *CAN FD Frames* from all CAN ports. It is also the default *CAN Port Input Filter Address* when the *Channel Group* and *Channel ID Set* are not defined in *Message Version* < 2.

Channel Group

The *Channel Group* (CG) is the first part of the *CAN Port Input Filter Address*: (CG, CIDS). The *Channel Group* format is the same as in the *CAN FD Frames*. It is defined in Message Version = 2.

Channel ID Set

The *Channel ID Set* (CIDS) field is the second part of the *CAN Port Input Filter Address*: (CG, CIDS). The *Channel ID Set* format is the same as in the *CAN FD Frames*. This field is defined in Message Version = 2.

Heartbeat

The *Heartbeat* message with *Message ID* = 4 is sent by the Ethernet to CAN converter every 1s to maintain a link with the connected node and to inform the node about the converter status. The most recent *Heartbeat* Message Versions is 2.

Heartbeat Data Fields

The *Heartbeat* message sends the following data:

HM = {MNB_1,…, MNB_4, TIB_1,…, TIB_4, HDB_1,…, HDB_4, CTB, CGB, $CIDSB_1$,…,$CIDSB_4$}

(5)

Where:

MNB_1,…, MNB_4 – 4-byte *Message Number*, LSB first;

TIB_1,…, TIB_4 – 4-byte *Time Interval* in milliseconds elapsed from the last *Heartbeat* message, LSB first;

HDB_1,…, HDB_4 – 4-byte *Health Data*, LSB first;

CTB – 1-byte *Converter Type* (Defined only in *Message Version ≥ 1*);

SFB_1,…, SFB_4 – 4-byte *Supported Features*, LSB first , (Defined only in *Message Version =2*) ;

CGB – 1-byte *Channel Group.* First part of the *CAN Port Input Filter Address.* (Defined only in *Message Version = 2*);

$CIDSB_1$,…,$CIDSB_4$ – 4-byte *Channel ID Set*, LSB first. Second part of the *CAN Port Input Filter Address*. (Defined only in *Message Version = 2*).

If the *Heartbeat* message is used only to maintain a connection between the node and the converter, the node can use a blank *Heartbeat* message with all data fields from *Message Version* < 2 set to zero. The *Supported Features, Channel Group,* and *Channel ID Set* fields from Message Version = 2 should contain regular values in the blank Heartbeat message.

Message Number
The *Message Number* is a value of a free-running global counter, which is incremented every time the *Heartbeat* message is generated. One counter is used for all connected nodes.

The nodes can examine the *Message Number* consistency to check the state of the data link between the node and the converter in case a connectionless UDP communication protocol is used to carry protocol messages.

The *Message Number* can be set to zero in a blank *Heartbeat* message.

Time Interval
The *Heartbeat Message* sends the *Time Interval* in milliseconds elapsed from the last *Heartbeat* message. This value is close to 1000 for 1s heartbeat interval and can be used by nodes to estimate delays in communication between the nodes and the Ethernet to CAN converter.

The *Time Interval* can be set to zero in a blank *Heartbeat* message.

Health Data
The *Health Data* field is the same as in the *Status Response* message [2]. In can be set to zero in a blank *Heartbeat* message.

Converter Type
The *Converter Type* field is the same as in the *Status Response* message. In can be set to zero in a blank *Heartbeat* message.

Supported Features
The *Supported Features* field is the same as in the *Status Response* message.

CAN Port Input Filter Address
The *CAN Port Input Filter Address* is the same as in the *Status Response* message.

Channel Group
The *Channel Group* field is the same as in the *Status Response* message.

Channel ID Set
The *Channel ID Set* field is the same as in the *Status Response* message.

References

[1] O. Bogush, "Ethernet to CAN Converter Discovery Protocol. Document version: 1A," Axiomatic Technologies Corporation, April 5, 2021.

[2] O. Bogush, "Ethernet to CAN Converter Health Status. Document version: 3," Axiomatic Technologies Corporation, April 5, 2021.

Document Version History

Document Version	Date	Author	Changes
5	December 14, 2022	Olek Bogush	Added support for CAN FD Stream. Deprecated support for CAN and Notification Stream. Added Supported Features, Channel Group, and Channel IDs to Status Response and Heartbeat Messages.
4	April 5, 2021	Olek Bogush	Added WiFi to CAN converters in Introduction section. Removed Health Data field format. It is now a part of the "Ethernet to CAN Converter Health Status. Document Version 3". Added *Converter Type* and different versions of the *Status Response* and *Heartbeat* messages. Updated *References* section.
3	October 26, 2016	Olek Bogush	Added the document version description in the *Introduction* section. Changed the *Protocol Basics* section. Defined the protocol independent message structure. Generalized the protocol message header format to support different *Protocol IDs*. Added *Connection Lifetime* subsection. Added a blank *Heartbeat* message. Added *References* section.
2	June 27, 2016	Olek Bogush	Added *Flash Memory Driver Initialization Operational Status* field. Updated reference to the *Ethernet to CAN Converter Health Status* document.
1A	February 5, 2016	Olek Bogush	Made the document generic. Removed references to the project 14010, except for the protocol ID. Replaced the *Ethernet to CAN Gateway* term with the term: *Ethernet to CAN Converter*. Corrected Table 10. Heath Status aggregation rules.
1	October 28, 2015	Olek Bogush	Initial version.