SAE J2716 Interface
Communication Protocol Specification

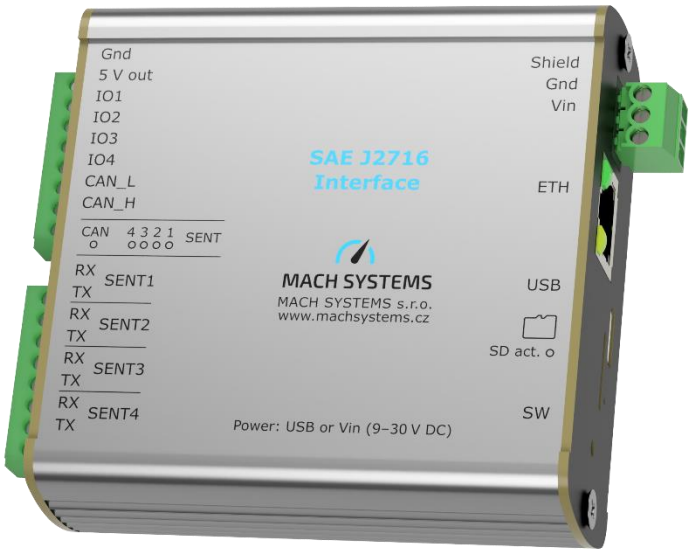
Changes

Date	Firmware Version	Change	Changed by
19. 6. 2024	1.7	Added DHCP Added sniffer functionality (see SENT Basic Configuration)	KH
11. 4. 2024	1.6	ETH parameters change clarification	KH
20. 11. 2023	1.5	Pause pulse clarification	KH
3. 11. 2023	1.3	Remove 0x94 SENT_STOP_SPC_RECEPTION	KH
31. 10. 2023	1.2	Rename ERRCODE to FRMERRCODE and value clarification. Added ERRCODE for invalid SYNC length	KH
15. 9. 2023	1.0	Fixes	MM
6. 9. 2023	1.0	Initial release	KH, MM

Contents

1. Introduction

The **SAE J2716 Interface** (p/n: **MACH-SENT-ETH)** is a flexible SAE J2716 (SENT bus) gateway, converter, and stand-alone data logger. The interface features four bi-directional SENT channels, Ethernet and USB ports, a CAN(/FD) channel, four analogue I/O channels, and a MicroSD card slot.



This document describes the communication protocol for controlling the device over Ethernet, USB, and CAN(/FD). This enables the possibility to integrate the device into other systems, such as HiL tester, test bench, simulator, PLC and others.

The communication protocol allows to:

- Configure the device (Ethernet, CAN(/FD))
- Configure the SENT channels
- Receive and transmit SENT Fast and Slow messages
- Configure the analogue I/Os and their mapping
- Configure MicroSD card logging and playback
- Use the device as an Ethernet-CAN(/FD) or USB-CAN(/FD) interface

Note: The most of the settings can also be configured over the device's web server.

The device can be used as a bi-directional gateway between SENT and Ethernet/USB/CAN(/FD), or as a stand-alone data logger of SENT and CAN(/FD) communication on a MicroSD card. The four analogue channels can be mapped on both received and transmitted SENT data, enabling a SENT-to-analogue output or analogue input-to-SENT conversion. The device can also act as a USB-CAN(/FD) or Ethernet-CAN(/FD) interface simultaneously to its SENT gateway/converter function.

The device features four independent SENT channels and allows the user to configure SENT parameters, receive and transmit SENT frame including the Short serial and Enhanced serial formats via the communication protocol. The device can also transmit multiple Slow messages with different Message Id over a SENT channel with the help of message buffers for slow message. Further, the device supports SENT/SPC mode and SENT physical layer polarity inversion, and can also be used to inject CRC fault errors into both Fast and slow messages.

The interface offers four analogue channels, each one configurable as input or output that can be mapped onto any of the device's SENT channel. The conversion settings are configurable by the user – start bit, bit length, linear transfer function: multiplier, offset, min/max voltage limits. The analogue input range is 0-5 V, and the analogue output range is 0 – 4.095 V (12-bit DAC).

When used as a SENT-CAN gateway, the device can be configured to transmit SENT Fast and Slow data frames over separated CAN identifiers, allowing a third-party system to easily process the data over CAN(/FD) network.

2. Communication Protocol

The communication between the **SAE J2716 Interface** and other system is based upon a binary protocol. The same message structure is used for both directions - to and from the device.
This protocol consists of Message Id and Data. For USB and Ethernet communication, the protocol is encapsulated by Start byte, Data length, Checksum and End byte. For CAN bus, the protocol is placed into the data bytes of a CAN frame.

2.1. USB and Ethernet

USB configuration: Virtual COM port (VCP), 115200 Baud, 8 data bits, no parity, 1 stop bit

Ethernet default configuration:

IP address	**192.168.1.100**
Subnet mask	255.255.255.0
Default gateway	192.168.1.1
Port	**8000**
Protocol	TCP or UDP

The device offers communication over both TCP/IP and UDP and its Ethernet parameters can be changed. This can be done directly from the protocol or, as there is a web server running, also via a web browser (Google Chrome is recommended).

Communication protocol structure:

STX (1B)	ID (1B)	DATALEN (2B)	DATA (X B)	CHECKSUM (1B)	ETX (1B)
0x02	MessageId	Number of data bytes (LSB first)	Data bytes Number of bytes = DATALEN	1-byte sum of ID, DATALEN and all DATA bytes	0x03

The rest of the document refers to DATA part only. The user is then responsible for encapsulating it with the rest of the protocol fields, namely STX, Id, DataLen, Checksum, and ETX.

2.2. CAN bus

If the device CAN port is not used as a USB/Ethernet-CAN interface, it is possible to use the CAN bus for diagnostic purposes. The device receives via CANID_RX and transmits over CANID_TX. Both CAN identifiers can be changed per device – see 3.5.1, 3.5.2, 3.5.3.

Default configuration:
CANID_RX = 0x123 Std Id.
CANID_TX = 0x321 Std Id.
CAN Baud = 500 KBaud, sample point: 80%
Frame format: Classical CAN (CAN FD support can be enabled)

CAN frame – data part:

CAN DATA 0 (1 B)	**CAN DATA (0 to 7 B for CAN) (0 to 63 B for CAN FD)**
MessageId	Protocol data bytes

Data Byte 0 is always used as MessageId, the rest of the data bytes carry the message content.

2.3. Message Overview

Note that all the channel indexes in the configuration (CAN, SENT, ADC) are zero indexed. Even though the first SENT channel is called SENT1, its index is 0. The CAN setting messages (Id 0x60 – 0x6A) are not available over CAN, usage of those IDs over CAN will result in error response. Also, SENT channel timestamp is not available over CAN (but is available over CAN FD).

Messa geId	Name	Request Data Length	Response Data Length	Description
0x01	BOOT_UP	No request needed	4	*A notification that the gateway was powered up. The data bytes of the response are CANID_BASE_RX. Sent via CAN and USB.*
Product information				
0x11	READ_SN	0	4	Read device serial number
0x12	READ_HW_INFO	0	6	Read device HW info
0x13	READ_SW_INFO	0	2	Read device SW info
Device configuration				
0x14	ETH_RESET_CONFIGURATION	0	0B ACK	Restore the default communication configuration
0x15	ETH_READ_CONFIGURATION	0	13	Read configuration
0x16	ETH_WRITE_CONFIGURATION	7	0B ACK	Write configuration
0x17	ETH_READ_IP_ADDRESS	0	5	Read IP address and mask
0x18	ETH_WRITE_IP_ADDRESS	5	0B ACK	Write IP address and mask
0x19	ETH_READ_PORT	0	2	Read communication port
0x1A	ETH_WRITE_PORT	2	0B ACK	Write communication port
0x1B	ETH_READ_MAC_ADDRESS	0	6	Read MAC address
0x1C	ETH_READ_DEFAULT_GW	0	4	Read default gateway
0x1D	ETH_WRITE_DEFAULT_GW	4	0B ACK	Write default gateway
0x1E	RTC_READ_TIMESTAMP	0	4	Read RTC second timestamp
0x1F	RTC_WRITE_TIMESTAMP	4	0B ACK	Write RTC second timestamp
0x20	ETH_DHCP	1	0B ACK or 1	Enable / disable the DHCP client
Device configuration – CAN protocol				
0x50	CAN_WRITE_LOCK_TOGGLE	1	0B ACK	Unlock / lock CAN parameters change
0x51	CAN_READ_RXID	0	4	Read CAN ID for protocol RX
0x52	CAN_WRITE_RXID	4	0B ACK	Write CAN ID for protocol RX
0x53	CAN_READ_TXID	0	4	Read CAN ID for protocol TX
0x54	CAN_WRITE_TXID	4	0B ACK	Write CAN ID for protocol TX
0x55	CAN_READ_SIMPLECONFIG	1	5	Read CAN basic settings (protocol, baud rate, sample point)
0x56	CAN_WRITE_SIMPLECONFIG	3 or 5	1B ACK	Write CAN basic settings (protocol, baud rate, sample point)

0x57	SENT_CAN_READ_ID	1	5	Read SENT alternative CAN ID configuration
0x58	SENT_CAN_WRITE_ID	5	1B ACK	Write SENT alternative CAN ID configuration
0x59	CAN_READ_LOGGING_INFO	1	2	Read CAN logging configuration
0x5A	CAN_WRITE_LOGGING_INFO	2	1B ACK	Write CAN logging configuration
0x5B	CAN_READ_STATUS	0	2	
CAN communication control				
0x60	CAN_WRITE_CONFIG	6	1B ACK	Configure CAN channel
0x61	CAN_WRITE_CONFIG_TIM	9	1B ACK	Configure CAN channel (with time quanta setting)
0x62	CAN_READ_CONFIG	1	12	Read CAN channel configuration
0x63	CAN_SAVE_CONFIG	1	1B ACK	Save CAN configuration to non-volatile memory
0x64	CAN_LOAD_CONFIG	1	1B ACK	Load CAN configuration from non-volatile memory
0x65	CAN_DEFAULT_CONFIG	1	1B ACK	Load CAN default configuration
0x66	CAN_ECHO_CONF	2	1B ACK	Enable / disable TX echo
0x67	CAN_START_CHANNEL	1	1B ACK	Start CAN channel
0x68	CAN_STOP_CHANNEL	1	1B ACK	Stop CAN channel
0x69	CAN_GET_TIMESTAMP	1	9	Get time in microseconds from startup of channel
0x6A	CAN_SEND_MESSAGE	5 to 71	1B ACK	Send CAN message / CAN message was sent
0x6B	CAN_RECEIVED_MESSAGE	N/A	13 to 79	Received CAN message
0x6C	CAN_ERROR_FRAME	N/A	10	Some error on CAN bus
SENT basic configuration				
0x70	SENT_READ_CFG	1	7	Read SENT channel configuration
0x71	SENT_WRITE_CFG	7	1B ACK	Write SENT channel configuration
0x72	SENT_READ_SPC_CFG	1	5	Read SENT channel SPC configuration
0x73	SENT_WRITE_SPC_CFG	5	1B ACK	Write SENT channel SPC configuration
0x74	SENT_START	1	1B ACK	Start SENT channel
0x75	SENT_STOP	1	1B ACK	Stop SENT channel
0x76	SENT_GET_TIMESTAMP	1	9	Get time in microseconds from startup of channel
0x77	SENT_LOAD_CONFIGURATION	0	0B ACK	Load SENT configuration from non-volatile memory
0x78	SENT_SAVE_CONFIGURATION	0	0B ACK	Write SENT configuration to non-volatile memory
0x79	SENT_DEFAULT_CONFIGURATION	0	0B ACK	Load SENT default settings

0x7A	SENT_READ_STATUS	0	4	Read SENT channel status (all channels at once)
0x7B	ADC_READ_VALUE	0	7	Read values of analogue inputs
0x7C	DAC_WRITE_VALUE	2	1B ACK	Write value to analogue output
SENT extended configuration				
0x80	SENT_DAC_READ_CONFIG	1	7	Read analogue output configuration
0x81	SENT_DAC_WRITE_CONFIG	7	1B ACK	Write analogue output configuration
0x82	SENT_DAC_READ_LIMIT	1	5	Read analogue output limits
0x83	SENT_DAC_WRITE_LIMIT	5	1B ACK	Write analogue output limits
0x84	SENT_ADC_READ_CONFIG	1	7	Read analogue input configuration
0x85	SENT_ADC_WRITE_CONFIG	7	1B ACK	Write analogue input configuration
0x86	SENT_READ_LOGGING_INFO	1	2	Read SENT logging configuration
0x87	SENT_WRITE_LOGGING_INFO	2	1B ACK	Write SENT logging configuration
0x88	SENT_RCNT_CONFIG	3	1B ACK	SENT RCNT configuration
0x89	SENT_START_PLAYBACK	2	1B ACK	Start SENT communication playback from log file
0x8A	SENT_STOP_PLAYBACK	1	1B ACK	Stop SENT communication playback from log file
0x8B	SENT_READ_FILE_COUNT	1	2	Read number of log files in the filesystem
0x8C	SENT_PLAYBACK_PROGRESS	0	2	Read percentage progress of playback
SENT message control				
0x90	SENT_SEND	4 to 7	1B ACK	Transmit SENT fast frame
0x91	SENT_SEND_SLOW	5	1B ACK	Write SENT slow frame buffer (no multiplexing)
0x92	SENT_WRITE_SLOW_BUFFER	5	1B ACK	Write SENT slow frame buffers (with multiplexing)
0x93	SENT_SPC_RECEIVE	2	1B ACK	Begin SENT/SPC reception
0x95	SENT_REC	N/A	4 to 7 or 12 to 15	SENT frame received
0x96	SENT_SLOW_REC	N/A	6 or 14	Slow SENT frame received (Short serial message or Enhanced serial message)
0x97	SENT_REC_ERR	N/A	2 or 10	SENT error received
0x98	SENT_SLOW_REC_ERR	N/A	2 or 10	SENT slow error received
0x99	SENT_TX_ECHO	N/A	4 to 7 or 12 to 15	SENT frame was transmitted

0x9A	SENT_SLOW_TX_ECHO	N/A	6 or 14	Slow SENT frame was transmitted
Miscellaneous				
0xFD	RESTART	0	0	Restart the device
0xFE	RESTART_BOOT	1	0	Restart device to USB / web bootloader
0xFF	GENERAL_ERROR	N/A	1, 2 or 3	An error occurred, see Error Codes for description

2.4. Error Codes

The following table describes error codes. General structure of error message is described below.

Error Code	Data length	Comment
Communication protocol error Messages contain: Error Code and MessageId		
0xA0	2	Incorrect end byte on the Ethernet protocol
0xA1	2	Bad checksum on the protocol
0xA2	2	Unknown MessageId
0xA3	2	Too large or incorrect data length
0xA4	2	Invalid data
0xA5	2	Attempt to change CAN configuration from CAN without unlock
0xA6	2	Configuration could not be saved – EEPROM error
SENT bus errors Messages contain: Error Code, MessageId and Channel Number		
0xE0	3	Message could not be transmitted
0xE1	3	Channel mode not compatible with requested operation
0xE2	3	Other error – some wrong argument
General bus errors Messages contain: Error Code, MessageId and Channel Number		
0xF0	3	Configuration Error
0xF1	3	Channel running, channel should be stopped when it is being configured
0xF2	3	Invalid channel selected – index out of bounds
0xF3	3	Channel is not running
0xF4	3	Hardware FIFO is full (should not happen in normal operation)

3. Message Specification

3.1. General Response

Device responds with a message acknowledgment after receiving a valid message. The acknowledgement has the same Id as the original message. Acknowledgement is either without any data or it contains one data byte which signals which bus channel was relevant to the request. See Response Data Length in Message Overview for information about acknowledge length. If there is some problem, the response is Error Response.

Response:
No data when bus channel number is not relevant.

OR

DATA 0
Channel number

3.2. Error Response

MessageId = 0xFF

Device responds with an error if the command could not be processed correctly. MessageId contains ID of the requesting message. If Channel number is relevant, error response is three-byte and last byte is number of the relevant channel.
Response:

DATA 0	DATA 1
Error code	MessageId

When Channel number is not relevant.

OR

DATA 0	DATA 1	DATA 2
Error code	MessageId	Channel number

When both MessageId and Channel number are relevant.

See the table above to determine which error message contains what information.

3.3. Device Message Specification

3.3.1. Device serial number

MessageId = 0x11
This command is used for reading device serial number.
Request 0x11:
No data

Response:

DATA 0 – DATA 3
Device serial number

Example S/N: 02030106

DATA 0	DATA 1	DATA 2	DATA 3
06	01	03	02

3.3.2.Device hardware information

MessageId = 0x12

This command is used for reading device hardware number.

Request 0x12:

No data

Response:

DATA 0 – DATA 5
Device hardware number

Example HW Info: 000400030002

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5
02	00	03	00	04	00

3.3.3.Device software information

MessageId = 0x13

This command is used for reading software version number.

Request 0x13:

No data

Response:

DATA 0	DATA 1
VERSION MINOR	VERSION MAJOR

3.4. ETH Message Specification

3.4.1.Restore Default Configuration

MessageId = 0x14

Reset the default communication configuration – IP address, mask and port. After issuing this command, you must restart the device for changes to apply.

Request: Message without data.

Response: acknowledge.

3.4.2.Read and Write Configuration

MessageId = 0x15 for read, 0x16 for write

This command is used for changing IP address, mask and TCP port all in one step. After writing the configuration, you must restart the device for changes to apply.

The read variant is for reading IP address, mask, TCP port and MAC address. The difference is that you cannot write the MAC address. Note that when this MessageId is used over CAN, there is no MAC address. See the next subchapter for IP address and mask formats.

Request 0x15:

No data

Response:

DATA0 – DATA 3	DATA 4	DATA 5 – DATA 6	DATA 7 – DATA 12
IP address	Mask	Port	MAC Address

Request for 0x16:

DATA0 – DATA 3	DATA 4	DATA 5 – DATA 6
IP address	Mask	Port

Response: acknowledge.

3.4.3.Read and Write IP Address Configuration

MessageId = 0x17 for read, 0x18 for write

This command is for reading or writing device IP address and subnet mask. IP address octets are in the order first to last, e.g. 192.168.1.100 is encoded as C0 A8 01 64. Mask is in the one number format, which determines how many non-zero bits there is. For example, 24 corresponds with mask 255.255.255.0 (1111 1111.1111 1111.1111 1111.0000 0000). After issuing this command, you must restart the device for changes to apply.

Request 0x18:

DATA0 – DATA 3	DATA 4
IP address	Mask

Response: acknowledge.

Response to ID 0x17 has the same structure as request 0x18 above.

3.4.4.Read and Write TCP Port

MessageId = 0x19 for read, 0x1A for write

This command is for changing the application communication port. Default port is 8000. After issuing this command, you must restart the device for changes to apply.

Request 0x19:

DATA0 – DATA 1
Port

Response: acknowledge.

Response to 0x1A has the same structure as request for 0x19 above.

3.4.5.Read MAC Address

MessageId = 0x1B

This command is for reading device MAC address. Each device has unique MAC address which cannot be changed by the user. MAC address octets are in the order first to last, e.g. A0:19:6E:C2:A5:FC is encoded as A0 19 6E C2 A5 FC.

fRequest 0x1B:

No data

Response:

DATA 0 – DATA 5
MAC address

3.4.6. Read and Write Default Gateway

MessageId = 0x1C for read, 0x1D for write

Used for reading / changing the default gateway. Value in default configuration is 0.0.0.0 (in standard environment, default gateway is not needed as it is assumed that communication is running in single network segment).
IP address octets are in the order first to last, e.g. 192.168.1.100 is encoded as C0 A8 01 64.
After issuing this command, you must restart the device for changes to apply.

Request 0x1D:

DATA 0 – DATA 3
Default gateway IP address

Response:
No data

Response to 0x1C has the same structure as request for 0x1D above.

3.4.7. Read and Write RTC Timestamp

MessageId = 0x1E for read, 0x1F for write

RTC timestamp is timestamp of **current time** in seconds since 1970-01-01 00:00:00. It is added to the SENT log.

Request for read: No data

Response:

DATA 0	DATA 1	DATA 2	DATA 3
Timestamp LSB	Timestamp [1]	Timestamp [2]	Timestamp MSB

Write request has exactly the same structure as response to read (4-byte timestamp in little-endian format).

3.4.8. Read and Write DHCP Enable / Disable

MessageId = 0x20

Reading and writing of DHCP client enable. When DHCP is enabled, issuing IP address, mask or default gateway read means reading the currently active network value.

Request for read:

DATA 0
0x0

Response:

DATA 0

DHCP enable 0x00 – Disabled 0x01 – Enabled

Request for write:

DATA 0
0x01 – disable 0x02 – enable

Response: No data

3.5. CAN Communication Protocol Configuration

3.5.1. Unlock and Lock CAN configuration change

MessageId = 0x50

Before changing CAN protocol Rx / Tx Id and changing simple CAN configuration (Id 52, 54, 56) **via CAN**, you must issue this command to enable configuration change (other channels (TCP,…) can change those settings without issuing this command). If the data byte equals to one, configuration is enabled. If it is something else, it is disabled again.

Request 0x50:

DATA 0
DATA = 1 for unlock DATA ≠ 1 for lock

3.5.2. Read and Write CAN protocol Rx Id

MessageId = 0x51 for read, 0x52 for write

If issued via CAN, command for unlocking CAN settings must be issued prior to write variant of this command. Reading and changing the CAN Id which is used for configuration messages. Default setting is 0x123 (standard Id).

Request 0x52:

DATA 0	DATA 1	DATA 2	DATA 3
New CAN Id for Rx LSB	New CAN Id Byte 1	New CAN Id Byte 2	New CAN Id MSB + information

New CAN Id MSB + information:

bit 7							
EXTId	FDF	Reserved	ID28	ID27	ID26	ID25	ID24

- Bit 7: **EXTId** – determines if configuration messages are received with extended Id
  - 0: Protocol messages must have Standard Id
  - 1: Protocol messages must have Extended Id
- Bit 6: **FDF** – CAN or CAN FD frame
  - 0: Protocol messages must not have FDF flag set (CAN frame)

- 1: Protocol messages must have FDF flag set (CAN FD frame). Valid only when channel operates in CAN FD mode
- Bit 5: Reserved
- Bits [4:0]: Bits [28:24] of extended CAN Id of configuration messages (if applicable)

Response: No data when success, error message if CAN configuration change was not previously unlocked.

Response to 0x51 has the same structure as request for 0x52 above.

3.5.3.Read and Write CAN Protocol Tx Id

MessageId = 0x53 for read, 0x54 for write

If issued via CAN, command for unlocking CAN settings must be issued prior to write variant. Reading and changing the CAN Id which is used for configuration messages. Default setting is 0x321.

Request 0x54:

DATA 0	DATA 1	DATA 2	DATA 3
New CAN Id for Tx LSB	New CAN Id Byte 1	New CAN Id Byte 2	New CAN Id MSB + information

New CAN Id MSB + information:

bit 7							
EXTId	FDF	BRS	ID28	ID27	ID26	ID25	ID24

- Bit 7: **EXTId** – determines if configuration messages are sent with extended Id
  - 0: Protocol messages have Standard Id
  - 1: Protocol messages have Extended Id
- Bit 6: **FDF** – CAN or CAN FD frame
  - 0: Protocol messages do not have FDF flag set (CAN frame)
  - 1: Protocol messages have FDF flag set (CAN FD frame). Valid only when channel operates in CAN FD mode
- Bit 5: **BRS** – Bit Rate Switch
  - 0: Protocol messages do not have BRS flag set
  - 1: Protocol messages have BRS flag set (relevant when FDF = 1 and channel operates in CAN FD mode)
- Bits [4:0]: Bits [28:24] of extended CAN Id of configuration messages (if applicable)

Response: No data when success, error message if CAN configuration change was not previously unlocked.

Response to 0x53 has the same structure as request for 0x54 above.

3.5.4.Read and Write CAN Protocol Settings

MessageId = 0x55 for read, 0x56 for write

If issued via CAN, command for unlocking CAN settings must be issued prior to write variant of this command. When the CAN channel is configured with this command the Sync-Jump Width is always set to 2. **Important remark**: This configures the CAN bus baud rate used for manipulating SENT

gateway over CAN. You should not use this message if you want to use the device as a USB-CAN interface (CAN channel must be stopped).

Three-byte variant of this message is meant for CAN mode. If you want to use CAN FD and want to set different arbitration and data baud rate and/or sample point, you should use the five-byte variant. If you turn on CAN FD mode, all the protocol messages are sent as CAN FD frames with bit rate switching is enabled.

Request 0x56 (three-byte):

DATA 0	DATA 1	DATA 2
Channel number and SAVE bit	Simple configuration register 0	Simple configuration register 1

Channel number and SAVE bit:

bit 7							
SAVE	CHNL6	CHNL5	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0

- Bit [7]: **SAVE**: Flag to tell if configuration will be saved right away.
  - 0: Do not save the configuration
  - 1: Store device configuration to EEPROM immediately after reconfiguration.
- Bits [6:0]: **CHNL**: Always 0.

Simple configuration register 0:

bit 7							
PROTOCOL1	PROTOCOL0	Reserved	Reserved	ASPSEL3	ASPSEL2	ASPSEL1	APSSEL0

- Bits [7:6]: PROTOCOL – Protocol selection
  - 00: CAN
  - 01: CAN FD
  - 10…11: Reserved
- Bits [5:4]: Reserved
- Bits [3:0]: ASPSEL – Arbitration sample point selector
  - 0000: 60 %
  - 0001: 62.5 %
  - 0010: 65 %
  - 0011: 67.5 %
  - 0100: 70 %
  - 0101: 72.5 %
  - 0110: 75 %
  - 0111: 77.5 %
  - 1000: 80 %
  - 1001: 82.5 %
  - 1010: 85 %
  - 1011: 87.5 %
  - 1100: 90 %
  - 1101…1111: Reserved

Simple configuration register 1:

bit 7							
Reserved	Reserved	Reserved	Reserved	Reserved	ABAUDSEL2	ABAUDSEL1	ABAUDSEL0

- Bits [2:0]: ABAUDSEL – Arbitration baud rate selector
  - 000: 125 kBd
  - 001: 250 kBd
  - 010: 500 kBd
  - 011: 1 MBd
  - 100…111: Reserved

Request 0x56 (five-byte, relevant for CAN FD):

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
Channel number and SAVE bit	Simple configuration register 0	Simple configuration register 1	Simple configuration register 2	Simple configuration register 3

Channel number and SAVE bit:

bit 7							
SAVE	CHNL6	CHNL5	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0

- Bit [7]: **SAVE**: Flag to tell if configuration will be saved right away.
  - 0: Do not save the configuration
  - 1: Store device configuration to EEPROM immediately after reconfiguration.
- Bits [6:0]: **CHNL**: Always 0.

Simple configuration register 0: Same meaning as above
Simple configuration register 1: Same meaning as above
Simple configuration register 2:

bit 7							
Reserved	DBAUDSEL2	DBAUDSEL1	DBAUDSEL0	Reserved	Reserved	Reserved	Reserved

- Bit 7: Reserved
- Bits [6:4]: DBAUDSEL – Data baud rate selector
  - 000: 1 MBd
  - 001: 2 MBd
  - 010: 4 MBd
  - 011: 8 MBd
  - 100..111: Reserved
- Bits [3:0]: Reserved

Simple configuration register 3:

bit 7							
Reserved	Reserved	Reserved	Reserved	DSPSEL3	DSPSEL2	DSPSEL1	DSPSEL0

- Bits [3:0]: DSPSEL – Data sample point selector
  - 0000: 60 %
  - 0001: 62.5 %

- 0010: 65 %
- 0011: 67.5 %
- 0100: 70 %
- 0101: 72.5 %
- 0110: 75 %
- 0111: 77.5 %
- 1000: 80 %
- 1001: 82.5 %
- 1010: 85 %
- 1011: 87.5 %
- 1100: 90 %
- 1101…1111: Reserved

Response:

DATA 0
Channel number (always 0)

Possible errors: Channel number is not 0, incorrect register values.

Request 0x55:

DATA 0
Channel number (always 0)

Channel number must be always 0 as other channels are not implemented.

Response:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
Channel number	Simple configuration register 0	Simple configuration register 1	Simple configuration register 2 (for CAN FD)	Simple configuration register 3 (for CAN FD)

Channel number: always 0.
Simple configuration registers have the same meaning as above. In case of CAN mode, configuration registers 2 and 3 have no meaning (they are set to 0xFF).

3.5.5. Read and Write SENT Alternative CAN ID

MessageId = 0x57 for read, 0x58 for write

For some data acquisition systems, it is easier to receive SENT data messages over a CAN frame with a CAN identifier different from the one used for the Communication Protocol. This will help the measurement system to process incoming CAN frames without checking the *MessageId* at Data[0] of each CAN frame.
This feature is enabled per a SENT channel. When enabled, the gateway will transmit Fast RX SENT frames and Slow RX SENT frames for the particular SENT channel over CAN messages with separated CAN identifiers. *RxForwardMode* is still applied.

Request:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
Channel number	CANID0 (LSB)	CANID1	CANID2	CANID3 (MSB)

Channel number is SENT channel (0 to 3).

CANID0:

bit 7							bit 0
ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0

…

CANID3:

bit 7							bit 0
EXTId	FDF	BRS	ID28	ID27	ID26	ID25	ID24

- CANID3 bit 7: **EXTId** – determines if used Id is Standard Id or Extended Id
  - 0: Messages have Standard Id
  - 1: Messages have Extended Id
- CANID3 bit 6: **FDF** – CAN or CAN FD frame
  - 0: Protocol messages do not have FDF flag set (CAN frame)
  - 1: Protocol messages have FDF flag set (CAN FD frame). Valid when CAN communication protocol is set to CAN FD mode.
- CANID3 bit 5: **BRS** – Bit Rate Switch
  - 0: Protocol messages do not have BRS flag set.
  - 1: Protocol messages have BRS flag set (relevant when FDF = 1 and CAN communication protocol set to CAN FD mode).
- CANID3 bits [4:0]: - Bits [28:24] of extended CAN Id (if applicable)
- ID: CAN Id of Fast Data Frame.
  - CAN Id of SENT Fast Error Frame is configured implicitly to CANID + 1
  - CAN Id of SENT Slow Data Frame is configured implicitly to CANID + 2
  - CAN Id of SENT Slow Error Frame is configured implicitly to CANID + 3

Note: Even when this feature is enabled, the gateway will still transmit these messages over the protocol CAN identifiers (see 3.9.5, 3.9.6, 3.9.7, 3.9.8). This means the information will be doubled.

Response:

DATA 0
Channel number

Read request:

DATA 0
Channel number

SENT channel number (0 to 3).

Response:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
Channel number	CANID0 (LSB)	CANID1	CANID2	CANID3 (MSB)

It has the same structure as request 0x7B for write (described above).

3.5.6. Read and Write CAN Logging Information

MessageId = 0x59 for read, 0x5A for write

Read configuration of CAN file logging. Used when MicroSD card is connected to the device. Logging is not configurable, you can only enable / disable it

Read request:

DATA 0
Channel number = 0

Response:

DATA 0	DATA 1
Channel number= 0	Logging configuration register 1

Logging configuration register 1:

bit 7							bit 0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	ENABLE

- Bits [7:1]: Reserved
- Bit 0: **ENABLE**
  - 0: Logging disabled
  - 1: Logging to file enabled when MicroSD card is inserted. Note that logging is really started after device restart or after channel is started. In other words, this bit does not tell if logging is really running, this is determined by CAN Channel Status Channel Status.

Write request:

DATA 0	DATA 1
Channel number = 0	Logging configuration register 1

Write command has exactly the same structure as response to read described above.

Response:

DATA 0
Channel number = 0

3.5.7. Read CAN Channel Status

MessageId = 0x5B

Read status of the CAN channel.
There are two status flags: Running and Logging.

- Running: Channel is started. Frames can be received/transmit, forwarding to PC is enabled.
- Logging: Channel logging is running. Frames are logged to an MicroSD Card if it is inserted.

Request: No data

Response:

DATA 0	DATA 1
CAN1 status	Reserved = 0xFF

CAN1 status:

bit 7	bit 2	bit 1	bit 0
Reserved		CH1LOGGING	CH1RUNNING

- Bits [7:2]: Reserved
- Bit 1: CH1LOGGING
- Bit 0: CH1RUNNING

3.6. CAN and CAN FD Messages

Messages in this section cannot be used over CAN. Attempt to use them over CAN will result in error response.

3.6.1.Channel Configuration

MessageId = 0x60

This message configures a CAN(/FD) channel. The time quanta for CAN FD controller are chosen by given sample point and baud rate. Sample point can be set exactly for baud rates up to 2 MBd. For 4 MBd, sample point is rounded to nearest lower multiple of 5 %; for 8 MBd, sample point is rounded to nearest lower multiple of 10 %. Note that for Data baud rate of 8 MBaud, Arbitration baud rate 1 MBaud should be used. The actual time quanta setting can be obtained by **Read Configuration** command. The CAN FD controller clock is 80 MHz.

Request:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5
Channel number and SAVE bit	Configuration Register 1	Configuration Register 2	Configuration Register 3	Configuration Register 4 **(CAN FD mode)**	Configuration Register 5 **(CAN FD mode)**

Channel number and SAVE bit:

bit 7							
SAVE	CHNL6	CHNL5	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0

- Bit [7]: **SAVE**: Flag to tell if configuration will be saved right away.
  - 0: Do not save the configuration
  - 1: Store device configuration to EEPROM immediately after reconfiguration.
- Bits [6:0]: **CHNL**: Always 0.

Configuration register 1:

bit 7							bit 0
PROTOCOL1	PROTOCOL0	AUTOSTART	ACK	ASP3	ASP2	ASP1	ASP0

- Bits [7:6]: **Protocol selection**
  - 00 – CAN 2.0B
  - 01 – ISO CAN FD
  - 10…11 – Reserved
- Bit 5: **AutoStart**
  - 0 – CAN channel is NOT automatically started on power-up
  - 1 – CAN channel is automatically started on power-up
- Bit 4: **Acknowledge mode**
  - 0 – Normal mode
  - 1 – Silent mode
- Bits [3:0]: **Arbitration Sample Point**
  - 0000 – 60%
  - 0001 – 62.5%
  - 0010 – 65%
  - 0011 – 67.50%
  - 0100 – 70%
  - 0101 – 72.50%

- 0110 – 75%
- 0111 – 77.50%
- 1000 – 80%
- 1001 – 82.50%
- 1010 – 85%
- 1011 – 87.50%
- 1100 – 90%
- 1101…1111 – Reserved

Configuration register 2:

bit 7							bit 0
Reserved	Reserved	Reserved	Reserved	Reserved	ABAUD2	ABAUD1	ABAUD0

- Bits [2:0]: **Arbitration baud rate**
  - 000 – 125 kBd
  - 001 – 250 kBd
  - 010 – 500 kBd
  - 011 – 1 MBd
  - 100…111 – Reserved

Configuration register 3:

bit 7							bit 0
Reserved	ASJW6	ASJW5	ASJW4	ASJW3	ASJW2	ASJW1	ASJW0

- Bits [6:0]: **Arbitration jump width**
  - 0000000 – 1
  - 0000001 – 2
  - 0000010 – 3
  - 0000011 – 4
  - …
  - 1111111 – 128

Configuration register 4 (**relevant for CAN FD mode only**):

bit 7							bit 0
Reserved	DBAUD2	DBAUD1	DBAUD0	DSJW3	DSJW2	DSJW1	DSJW0

- Bits [6:4]: **Data baud rate**
  - 000 – 1 MBd
  - 001 – 2 MBd
  - 010 – 4 MBd
  - 011 – 8 MBd
  - 100..111 – Reserved
- Bits [3:0]: **Data Synchronization jump width**
  - 0000 – 1
  - 0001 – 2
  - 0010 – 3
  - 0011 – 4
  - …
  - 1111 – 16

Configuration register 5 (**relevant for CAN FD mode only**):

bit 7							bit 0
Reserved	Reserved	Reserved	Reserved	DSP3	DSP2	DSP1	DSP0

- Bits [3:0]: **Data Sample Point**. Note that for baud rate 4 MBd, actual value will be rounded to nearest lower multiple of 5 %. For 8 MBd, it is nearest lower 10 %. Furthermore, 87.5 % will always be rounded to 85 % due to hardware constraints.
  - 0000 – 60%
  - 0001 – 62,5%
  - 0010 – 65%
  - 0011 – 67,50%
  - 0100 – 70%
  - 0101 – 72,50%
  - 0110 – 75%
  - 0111 – 77,50%
  - 1000 – 80%
  - 1001 – 82,50%
  - 1010 – 85%
  - 1011 – 87,50%
  - 1100 – 90%
  - 1101 – Reserved
  - 1110 – Reserved
  - 1111 – Reserved

Response:

DATA 0
Channel number (always 0)

Possibilities for error: CAN channel cannot be reconfigured - wrong arbitration or data jump width.

Default configuration

- ISO CAN FD
- Normal mode
- Arbitration speed 500 kBd
- Arbitration SJW 8
- Arbitration Sample Point 80%
- Data speed 2 MBd
- Data SJW 4
- Data Sample Point 80 %
- Autostart disabled

3.6.2.Channel Configuration with Time Quanta Timing

MessageId = 0x61

Same as channel configuration, except in this case also exact time quanta sizes are set.

Request:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4

Channel and SAVE bit	Configuration register 1	Configuration register 6	Configuration register 7	Configuration register 8
DATA 5	**DATA 6**	**DATA 7**	**DATA 8**	
Configuration register 9	Configuration register 10 **(CAN FD mode)**	Configuration register 11 **(CAN FD mode)**	Configuration register 12 **(CAN FD mode)**	

Channel number and SAVE bit:

bit 7							
SAVE	CHNL6	CHNL5	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0

- Bit [7]: **SAVE**: Flag to tell if configuration will be saved right away.
  - 0: Do not save the configuration
  - 1: Store device configuration to EEPROM immediately after reconfiguration.
- Bits [6:0]: **CHNL**: Always 0.

Configuration register 1: Has the same structure as in Channel Configuration except there is no ASP (arbitration selection) field. Those bits have no meaning here.

bit 7							bit 0
PROTOCOL1	PROTOCOL0	AUTOSTART	ACK	Reserved	Reserved	Reserved	Reserved

- Bits [7:6]: **Protocol**
  - 00 – CAN 2.0B
  - 01 – ISO CAN FD
  - 10…11 – Reserved
- Bit 5: **AutoStart**
  - 0 – CAN channel is NOT automatically started on power-up
  - 1 – CAN channel is automatically started on power-up
- Bit 4: **Acknowledge mode**
  - 0 – Normal mode
  - 1 – Silent mode
- Bits [3:0]: Reserved

Configuration register 6:

bit 7							bit 0
ATSEG1_7	ATSEG1_6	ATSEG1_5	ATSEG1_4	ATSEG1_3	ATSEG1_2	ATSEG1_1	ATSEG1_0

- Bits [7:0]: **Arbitration time segment 1**
  - 0000 0000 – 1
  - 0000 0001 – 2
  - 0000 0010 – 2
  - 0000 0011 – 3
  - …
  - 1111 1111 – 256

Configuration register 7:

bit 7							bit 0
Reserved	ATSEG2_6	ATSEG2_5	ATSEG2_4	ATSEG2_3	ATSEG2_2	ATSEG2_1	ATSEG2_0

- Bits [6:0]: **Arbitration time segment 2**

- 000 0000 – 1
- 000 0001 – 2
- 000 0010 – 3
- 000 0011 – 4
- …
- 111 1111 – 128

Configuration register 8:

bit 7							bit 0
APRESC_7	APRESC_6	APRESC_5	APRESC_4	APRESC_3	APRESC_2	APRESC_1	APRESC_0

- Bits [7:0]: **Arbitration prescaler**
  - 0000 0000 – 1
  - 0000 0001 – 2
  - 0000 0010 – 3
  - 0000 0011 – 4
  - …
  - 1111 1111 – 256

Configuration register 9:

bit 7							bit 0
Reserved	ASJW6	ASJW5	ASJW4	ASJW3	ASJW2	ASJW1	ASJW0

- Bits [6:0]: **Arbitration jump width**
  - 000 0000 – 1
  - 000 0001 – 2
  - 000 0010 – 3
  - 000 0011 – 4
  - …
  - 111 1111 – 128

Configuration register 10 (**relevant for CAN FD mode only**):

bit 7							bit 0
Reserved	Reserved	Reserved	DTSEG1_4	DTSEG1_3	DTSEG1_2	DTSEG1_1	DTSEG1_0

- Bits [4:0]: **Data time segment 1**
  - 0 0000 – 1
  - 0 0001 – 2
  - 0 0010 – 2
  - 0 0011 – 3
  - …
  - 1 1111 – 32

Configuration register 11 (**relevant for CAN FD mode only**):

bit 7							bit 0
DSJW3	DSJW2	DSJW1	DSJW0	DTSEG2_3	DTSEG2_2	DTSEG2_1	DTSEG2_0

- Bits [7:4]: **Data Synchronization jump width**
  - 0000 – 1
  - 0001 – 2

- 0010 – 3
- 0011 – 4
- …
- 1111 – 16

- Bit [3:0]: **Data time segment 2**
  - 0000 – 1
  - 0001 – 2
  - 0010 – 3
  - 0011 – 4
  - …
  - 1111 – 16

Configuration register 12 (**relevant for CAN FD mode only**):

bit 7							bit 0
Reserved	Reserved	Reserved	DPRESC4	DPRESC3	DPRESC2	DPRESC1	DPRESC0

- Bits [4:0]: **Data prescaler**
  - 0 0000 – 1
  - 0 0001 – 2
  - 0 0010 – 3
  - 0 0011 – 4
  - …
  - 1 1111 – 32

Response:

DATA 0
Channel number (always 0)

Possible errors: CAN channel cannot be reconfigured - wrong arbitration or data jump width.

3.6.3.Read Configuration

MessageId = 0x62

This command reads CAN interface settings. If configuration is set by precise timing message, 0xF values are set instead of Sample point and Baud rate values.

Request:

DATA 0
Channel number

Channel number must always be set to 0.

Response:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5
Channel number (always 0)	Configuration register 1	Configuration register 2	Configuration register 3	Configuration register 6	Configuration register 7
DATA 6	**DATA 7**	**DATA 8**	**DATA 9**	**DATA 10**	**DATA 11**

Configuration register 8	Configuration register 4 **(CAN FD mode)**	Configuration register 5 **(CAN FD mode)**	Configuration register 10 **(CAN FD mode)**	Configuration register 11 **(CAN FD mode)**	Configuration register 12 **(CAN FD mode)**
DATA 12					
Echo configuration					

Configuration register 1: see Channel Configuration for this register's field description.

bit 7							bit 0
PROTOCOL1	PROTOCOL0	AUTOSTART	ACK	ASP3	ASP2	ASP1	ASP0

Configuration register 2: see Channel Configuration for this register's field description.

bit 7							bit 0
Reserved	Reserved	Reserved	Reserved	Reserved	ABAUD2	ABAUD1	ABAUD0

Configuration register 3: see Channel Configuration for this register's field description. Same as register 10 in Channel Configuration with Time Quanta Timing.

bit 7							bit 0
Reserved	ASJW6	ASJW5	ASJW4	ASJW3	ASJW2	ASJW1	ASJW0

Configuration register 6: see Channel Configuration with Time Quanta Timing for this register's field description.

bit 7							bit 0
ATSEG1_7	ATSEG1_6	ATSEG1_5	ATSEG1_4	ATSEG1_3	ATSEG1_2	ATSEG1_1	ATSEG1_0

Configuration register 7: see Channel Configuration with Time Quanta Timing for this register's field description.

bit 7							bit 0
Reserved	ATSEG2_6	ATSEG2_5	ATSEG2_4	ATSEG2_3	ATSEG2_2	ATSEG2_1	ATSEG2_0

Configuration register 8: see Channel Configuration with Time Quanta Timing for this register's field description.

bit 7							bit 0
APRESC_7	APRESC_6	APRESC_5	APRESC_4	APRESC_3	APRESC_2	APRESC_1	APRESC_0

Configuration register 4 (**relevant for CAN FD mode only**): see Channel Configuration for this register's field description.

bit 7							bit 0
Reserved	DBAUD2	DBAUD1	DBAUD0	DSJW3	DSJW2	DSJW1	DSJW0

Configuration register 5 (**relevant for CAN FD mode only**): see Channel Configuration for this register's field description.

bit 7							bit 0
Reserved	Reserved	Reserved	Reserved	DSP3	DSP2	DSP1	DSP0

Configuration register 10 (**relevant for CAN FD mode only)**: see Channel Configuration with Time Quanta Timing for this register's field description.

bit 7							bit 0
Reserved	Reserved	Reserved	DTSEG1_4	DTSEG1_3	DTSEG1_2	DTSEG1_1	DTSEG1_0

Configuration register 11 (**relevant for CAN FD mode only)**: see Channel Configuration with Time Quanta Timing for this register's field description. Only difference is that there is not the DSJW field (it can be seen in register 4).

bit 7							bit 0
Reserved	Reserved	Reserved	Reserved	DTSEG2_3	DTSEG2_2	DTSEG2_1	DTSEG2_0

Configuration CHANNEL N Register 12 (**relevant for CAN FD mode only)**: see Channel Configuration with Time Quanta Timing for this register's field description.

bit 7							bit 0
Reserved	Reserved	Reserved	DPRESC4	DPRESC3	DPRESC2	DPRESC1	DPRESC0

Echo configuration:

bit 7							bit 0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	TXECHO	RXECHO

- Bit 1: **TX Echo on / off**
  - 0 – Echo Off
  - 1 – Echo On (default)
- Bit 0: **RX Echo on / off**
  - 0 – Echo Off
  - 1 – Echo On (default)

3.6.4.Save configuration

MessageId = 0x63

Saves CAN configuration to the non-volatile memory. Note that configuration can be also saved in the moment of reconfiguration if SAVE bit is used (see above).

Request:

DATA 0
Channel number

Channel number must always be set to 0.

Response:

DATA 0
Channel number

Acknowledgment when configuration was saved, general error message in case of error.

3.6.5.Load Configuration

MessageId = 0x64

Load the CAN configuration from non-volatile memory.

Request:

DATA 0
Channel number

Channel number must always be set to 0.

Response:

DATA 0
Channel number

Acknowledgment when configuration was loaded, general error message in case of error.

3.6.6.Default Configuration

MessageId = 0x65

Apply CAN default configuration. For default configuration values see 3.6.1 Channel Configuration.

Request:

DATA 0
Channel number and SAVE bit

Channel number and SAVE bit:

bit 7							
SAVE	CHNL6	CHNL5	CHNL4	CHNL3	CHNL2	CHNL1	CHNL0

- Bit [7]: **SAVE**: Flag to tell if configuration will be saved right away.
  - 0: Do not save the configuration
  - 1: Store device configuration to EEPROM immediately after reconfiguration.
- Bits [6:0]: **CHNL**: Always 0.

Response:

DATA 0
Channel number

Acknowledgment when configuration was loaded, general error message in case of error.
Reasons for error: wrong channel selected, CAN channel is already running.

3.6.7.Frame Echo Configuration

MessageId = 0x66

Request:

DATA 0	DATA 1
Channel number	Echo configuration

Channel number must always be set to 0.
Echo configuration:

bit 7							bit 0
Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	TXECHO	RXECHO

- Bit 1: **TX Echo on/off**
  - 0 – Echo off (default)
  - 1 – Echo on
- Bit 0: **RX Echo on/off**
  - 0 – Echo off
  - 1 – Echo on (default)

Response:

DATA 0
Channel number

Acknowledgment when echo configuration was changed, general error message in case of error.
Reasons for error: wrong channel selected, CAN channel is already running.

3.6.8.Start Channel

MessageId = 0x67
Start CAN channel.

Request:

DATA 0
Channel number

Channel number must always be set to 0.

Response:

DATA 0
Channel number

Acknowledgment when channel was started, general error message in case of error.
Reasons for error: wrong channel selected, CAN channel is already running

3.6.9.Stop Channel

MessageId = 0x68
Stop CAN channel.

Request:

DATA 0
Channel number

Channel number must always be set to 0.

Response:

DATA 0
Channel number

Acknowledgment when channel was stopped, general error message in case of error.
Reasons for error: wrong channel selected, CAN channel is not running

3.6.10. Get Channel Timestamp

MessageId = 0x69

Get CAN channel timestamp. Timestamp is 64-bit number that represents the time from startup of CAN(/FD) channel in microseconds.

Request:

DATA 0
Channel number

Channel number must always be set to 0.

Response:

DATA 0	DATA 1 - 8
Channel number	Timestamp byte 0 – 7

Channel number: always 0.
Timestamp: microsecond timestamp LSB first.

3.6.11. Transmit Frame / Frame Transmitted

MessageId = 0x6A

This message transmits CAN frame. The structure of frame is different when Extended ID is set. Without extended ID the header (protocol frame data before CAN data) is 5 bytes long. With extended ID it is 7 bytes long. The format of ID is LSB.
When the frame is sent on the bus, there is another message with this frame Id.

Request if EXTId bit (see below) is 0:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5 - n
Channel	MESSAGE_INFO	ID0	ID1	DLC	DATA

Request if EXTId bit (see below) is 1:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7 - n
Channel	MESSAGE_INFO	ID0	ID1	ID2	ID3	DLC	DATA

Channel must be set to 0.
MESSAGE_INFO:

bit 7							bit 0
Reserved	Reserved	Reserved	FDF	ESI	BRS	RTR	EXTId

- Bit 4: **FDF**
  - 0 – Frame to be transmitted in Classic CAN format
  - 1 – Frame to be transmitted in FDCAN format
- Bit 3: **ESI**
  - 0 – Transmitting node is error active
  - 1 – Transmitting node is error passive
- Bit 2: **BRS**
  - 0 – FDCAN frames transmitted / received without bit rate switching

    - 1 – FDCAN frames transmitted / received with bit rate switching
- Bit 1: **RTR**
    - 0 – Data frame
    - 1 – Remote frame
- Bit 0: **EXTId**
    - 0 – Standard ID. Request without data is 5 bytes.
    - 1 – Extended ID. Request without data is 7 bytes.

DLC: Number of data bytes.

Response:

DATA 0
Channel number

Acknowledgment if the frame was successfully passed to the controller for transmission, general error message in case of error.
Possible reasons for error: wrong bit configuration.

If Tx echo is enabled, device sends another message with this Id after the CAN frame is really sent on the bus. Its format again depends on state of the EXTId bit (standard or extended CAN ID).
Response with EXTId equal to zero:

DATA 0	DATA 1	DATA 2 - 9	DATA 10	DATA 11	DATA 12	DATA 13 - n
Channel	MESSAGE_INFO	Timestamp byte 0 - 7	ID0	ID1	DLC	DATA

Response with EXTId equal to one:

DATA 0	DATA 1	DATA 2 -9	DATA 10	DATA 11	DATA 12	DATA 13
Channel	MESSAGE_INFO	Timestamp bytes 0 - 7	ID0	ID1	ID2	ID3

DATA 14	DATA 15 - n
DLC	CAN DATA

Timestamp is 64-bit number that represents the time from startup of CAN(/FD) channel in microseconds. The bit order in message is LSB. Other parts of this message are the same as in the send request.

3.6.12. Frame Received

MessageId = 0x6B
Message response has similar structure as Transmit Frame. The only difference is in the added timestamp bytes (data bytes 2 to 9). Timestamp represents the time from startup of CAN(/FD) channel to reception of the frame in microseconds. For this message no request is needed, device sends it when it receives some CAN frame. Upon reception of CAN error frame, CAN Error Frame described in the next section is sent.

Response if frame with Standard Id was received (EXTId bit is 0):

DATA 0	DATA 1	DATA 2...9	DATA 10	DATA 11	DATA 12	DATA n
Channel number	MESSAGE_INFO	Timestamp byte 0 - 7	ID0	ID1	DLC	DATA

Response if frame with Extended Id was received (EXTId bit is 1):

DATA 0	DATA 1	DATA 2...9	DATA 10	DATA 11
Channel number	MESSAGE_INFO	Timestamp byte 0 - 7	ID0	ID1
DATA 12	**DATA 13**	**DATA 14**	**DATA n**	
ID2	ID3	DLC	DATA	

Meaning of the fields is exactly the same as in transmission request message.

3.6.13. CAN Error Frame

MessageId = 0x6C

This message is sent asynchronously when there is some error on CAN.

Response:

DATA 0	DATA 1	DATA 2...9
Channel number	Error type	Timestamp byte 0 – 7

Channel number: Always 0

Error type:

- 0: Bit Stuff Error
- 1: Form Error
- 2: Acknowledge Error
- 3: Bit Error
- 4: CRC Error

Timestamp: 64-bit number representing duration in microseconds from channel start.

3.7. SENT Basic Configuration

This part describes configuration of SENT bus. The configuration is split into structures that are stored in RAM. There are four SENT channels with indexes 0 to 3.
If the user wants to save the configuration from RAM into a non-volatile memory, he transmits SENT_WRITE_CFG (see Read and Write SENT Configuration) message to the device. Similarly, SENT_READ_CFG message is used to load a configuration into registers in RAM. If a valid configuration is present in a non-volatile memory, it is automatically loaded on power-up.
In some cases (depending on the connected SENT device), you will have to disable HW CRC as some sensor types count the CRC with the status field (against the SENT standard) – otherwise you will receive an error frame. Furthermore, sometimes you have to swap the data nibbles to receive correct data. Both can be done in the channel configuration.
Sniffing – channel can be configured as sniffer (by selecting sniffer source), which means that SENT traffic from source channel is forwarded to the sniffing channel. For more information about this feature, see User Manual.

3.7.1. Read and Write SENT Configuration

MessageId = 0x70 for read, 0x71 for write

Request 0x70:

DATA 0
Channel number

Channel number is SENT channel.

Response:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6
SENT channel + SENT configuration register 1	SENT configuration register 2	SENT configuration register 3	Unit time LSB	Unit time MSB	Pause pulse LSB	Pause pulse MSB

SENT channel + SENT configuration register 1:

bit 7							bit 0
SNIFFER2	SNIFFER1	SNIFFER0	INVERT	SWAPFASTDATANIBBLES	CHNL2	CHNL1	CHNL0

- Bits [7:5]: **SNIFFER** – Channel is used for sniffing (source channel selection). Channel cannot select neither itself nor another sniffing channel. Configuration is taken from the source channel, moreover, channel direction is set to TX, Auto Start bit is enabled and SENT/SPC is disabled
  - 0: Invalid – sniffer is not used
  - 1: SENT1 is used for sniffer
  - 2: SENT2 is used for sniffer
  - 3: SENT3 is used for sniffer
  - 4: SENT4 is used for sniffer
  - 5…7: Reserved
- Bit 4: **INVERT**
  - 0 – SENT normal operation (default)
  - 1 – SENT bus is inverted (idle in 0, non-standard). Note that when bus is inverted, SPC mode cannot be used.
- Bit 3: **SWAPFASTDATANIBBLES**
  - 0 – Data nibble order is unchanged
  - 1 – Data nibbles are swapped within a byte
- Bits [2:0]: **CHNL** – Channel which configuration was requested
  - 0: SENT1
  - 1: SENT2
  - 2: SENT3
  - 3: SENT4
  - 4…7: Reserved

SENT configuration register 2:

bit 7							bit 0
NIBBLECNT3	NIBBLECNT2	NIBBLECNT1	NIBBLECNT0	CRCMODE1	CRCMODE0	DIR	AUTOSTART

- Bits [7:4]: **NIBBLECNT** - Number of data nibbles. Values 1 to 8 are valid.
- Bits [3:2]: **CRCMODE**
  - 0 – HW CRC off
    - Frame CRC for transmission is taken from the sending request. For reception, CRC mismatch does not generate an error! CRC fields in the forwarded frame are still valid and CRCCALC field is calculated by the standard
  - 1 – HW CRC on
    - Frame CRC is calculated by the standard
  - 2 – SW CRC
    - Frame CRC is calculated from data bytes and status nibble (not by the standard!)
  - 3 – Faulty – CRC Error injection
    - Purposely incorrect CRC is used for transmission. Note that for reception, CRC mismatch does not generate an error.
- Bit 1: **DIR** - Channel direction
  - 0 – SENT TX
  - 1 – SENT RX
- Bit 0: **AUTOSTART**
  - 0 – Channel is not started on power-up
  - 1 – Channel is started on power-up

SENT configuration register 3:

bit 7			bit 4
SPCENABLE	SLOWTXCRCFAULT	SLOWTXECHO	SLOWCHANNELMODE1

bit 3			bit 0
SLOWCHANNELMODE0	FWDMODE1 ECHOMODE1	FWDMODE0 ECHOMODE0	PULSEPAUSEENABLE

- Bit 7: **SPCENABLE** – enable usage of SENT/SPC on channel. When INVERT bit is on, SPC mode cannot be used.
- Bit 6: **SLOWTXCRCFAULT** – enable Tx slow message CRC fault injection
  - 0 – Disabled
  - 1 – Enabled. Relevant when SLOWCHANNELMODE is 1 or 2
- Bit 5: **SLOWTXECHO** – enable of Tx echo for slow frames. This is relevant when SLOWCHANNELMODE is not zero and channel is configured as Tx.
  - 0 – Slow frames are not echoed when transmitted
  - 1 – Slow frames are echoed when transmitted
- Bits [4:3]: **SLOWCHANNELMODE**
  - 0 – Fast channel only
  - 1 – Short serial
  - 2 – Enhanced serial
- Bits [2:1]: **FWDMODE** or **ECHOMODE** – fast channel forward mode for Rx or echo mode for Tx
  - 0 – As fast as possible (Rx) or No echo (for Tx)
  - 1 – Fixed 10 ms
  - 2 – Fixed 100 ms

    - 3 – On change + 1 s
- Bit 0: **PULSEPAUSEENABLE**
    - 0 – Pause pulse not used
    - 1 – Transmit/receive frames with pause pulse. This can be enabled if correct length is filled in the last two registers. See below for more info.

Unit time: Tick time to be used. Stored in tens of nanoseconds (3 us => 300).

- Valid range is 50 to 9000 (500 ns to 90 us)
- Note that in order to use Tick Time of 500 us, some hardware changes must be done. See device user manual for more details.

Pause pulse: Relevant for both TX and RX.

- TX: SENT frame length in ticks (FrameTime). See the table below for possible values for different numbers of data nibbles.
- RX: When bus frames use Pause pulse, it must be enabled so that communication can be correctly decoded. Furthermore, when Pause pulse is enabled, framing errors that may originate from detecting two successive Sync Pulses are ignored. This is needed because Pause pulse could be interpreted as a valid Sync pulse and the following actual Sync pulse could be interpreted as a framing error.

Pause pulse range:

Number of data nibbles	Minimum Frame Time [ticks]	Maximum Frame Time [ticks]
1	147	860
2	174	872
3	201	884
4	228	896
5	255	908
6	282	920
7	309	932
8	336	944

Those equations apply:

$$T_{frame}[us] = FrameTime * T_{tick}$$

$$120 + 27 * N \leq FrameTime \leq 848 + 12 * N$$

where N is number of data nibbles.

Request 0x71 has same structure as response for 0x70:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6
SENT channel + SENT configuration register 1	SENT configuration register 2	SENT configuration register 3	Unit time LSB	Unit time MSB	Pause pulse LSB	Pause pulse MSB

See above for the meaning of the configuration structures. SENT channel has to be stopped before writing its configuration.

Response:

DATA 0
Channel number

Acknowledgment when channel was successfully reconfigured, error message when there was some problem (wrong channel number, wrong configuration, channel was running).

3.7.2.Read and Write SENT/SPC Configuration

MessageId = 0x72 for read, 0x73 for write

Channels can operate in the SENT/SPC mode. This standard adds a synchronous slave functionality – when slave detects a master pulse of the set length, it sends a SENT frame on the bus. This is used mainly for communication with some sensors.
If channel direction in the configuration is set to TX, device is slave. When it detects a pulse longer than `SpcPulseLengthMin` and shorter than `SpcPulseLengthMax`, it sends a frame. **Note** that when the SPC mode is enabled, **all the fast frames** are sent this way – only after a master pulse. If the direction is RX, device is master and it initiates the reception by sending the master request pulse with the length of `SpcPulseLength`. This is done using the SPC start reception message. If the request is for the periodical master pulse, it is sent with the set period (`SpcPeriod`). Some hardware changes must be done in order to use SENT/SPC mode. See device user manual for more details.

Request 0x72 – reading the configuration:

DATA 0
Channel number

Channel number is SENT channel (0 to 3).

Response:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
CHANNEL	SPCPULSELENGTHMIN	SPCPULSELENGTHMAX	SPCPULSELENGTH	SPCPERIOD

- **CHANNEL**: SENT channel which configuration was requested.
- **SPCPULSELENGTHMIN**: Minimal pulse length for sending slave response. Value is in Unit (Tick) Time. This is used for TX direction.
- **SPCPULSELENGTHMAX**: Maximal pulse length for sending slave response. Value is in Unit (Tick) Time. This is used for TX direction.
- **SPCPULSELENGTH**: Master pulse length. Used for RX direction.
- **SPCPERIOD**: Period in milliseconds of sending the master pulse. Used for RX direction

Request 0x73 for write has the same structure as response above.

3.7.3.Starting and Stopping SENT Channels

MessageId = 0x74 for start, 0x75 for stop

Start / stop SENT channel. Started channel means that it can transmit and receive frames to PC. Note that channel can be started or stopped even if logging is enabled (see SENT Status and Extended Configuration for more information). However, no channel can be started/stopped when replaying of SENT frames is active.

SENT fast/slow frames and errors are echoed to the **communication channel from which SENT channel was started** (except for when channel is started from website – then all available communication channels are used). Furthermore, see section about channel configuration in Read and Write SENT Configuration for info about frame echo.

Request:

DATA 0
Channel number

Channel number is SENT channel (0 to 3). Value 0xFF means that all channels are selected. Note: if all channels are selected for starting and one of the channels is running, no error is returned. Otherwise, if start of a running channel is requested, it is considered an error.

Response:

DATA 0
Channel number

Acknowledgment when channel was successfully started / stopped, error message when error occurred (for example wrong channel number).

3.7.4.Read SENT channel timestamp

MessageId = 0x76

Channel timestamp is 64-bit number representing time in microseconds since the channel was started. Timestamp of a stopped channel reads as 0. Note that this MessageId is not available over CAN.

Request:

DATA 0
Channel number

Channel number is SENT channel number (0 to 3).

Response:

DATA 0	DATA 1 – DATA 8
Channel number	Timestamp

Channel number is the same as in request.
Timestamp: 64-bit unsigned integer sent LSB first.

3.7.5.Loading and Writing SENT configuration to/from non-volatile memory

MessageId = 0x77 for loading, 0x78 for saving

Saves / loads SENT configuration to / from the non-volatile memory. Loading can be done only when none of the channels is running.

Request: No data (it applies to all channels).

Response: No data when configuration was saved / loaded, general error message in case of error.

3.7.6.Load SENT Default Settings

MessageId = 0x79

Apply SENT default configuration. This can be done only if the channel is not running.

Request: No data (it applies default settings to all the channels)

Response: No data when configuration was loaded, general error message in case of error.
Reasons for error: some SENT channel is currently running.

3.7.7.Read SENT Status

MessageId = 0x7A

Read status of all the SENT channels.
Each channel has three status flags: Running, Logging and Replay.

- Running: Channel is started. Frames can be received/transmit, forwarding to PC is enabled.
- Logging: Channel logging is running. Frames are logged to an MicroSD Card if it is inserted. Note that Logging and Running can be combined freely.
- Replay: Channel playback from log file is running. When this is active, neither Running nor Logging can be turned on.

Request: No data

Response:

DATA 0	DATA 2	DATA 3	DATA 4
SENT channel 1 status	SENT channel 2 status	SENT channel 3 status	SENT channel 4 status

SENT channel 1 status:

bit 7	bit 3	bit 2		bit 0
Reserved		CH1REPLAY	CH1LOGGING	CH1RUNNING

- Bits [7:3]: Reserved
- Bit 2: CH1REPLAY
- Bit 1: CH1LOGGING
- Bit 0: CH1RUNNING

SENT channel 2 status:

bit 7	bit 3	bit 2		bit 0
Reserved		CH2REPLAY	CH2LOGGING	CH2RUNNING

- Bits [7:3]: Reserved
- Bit 2: CH2REPLAY

- Bit 1: CH2LOGGING
- Bit 0: CH2RUNNING

SENT channel 3 status:

bit 7	bit 3	bit 2	bit 0
Reserved		CH3REPLAY	CH3LOGGING	CH3RUNNING

- Bits [7:3]: Reserved
- Bit 2: CH3REPLAY
- Bit 1: CH3LOGGING
- Bit 0: CH3RUNNING

SENT channel 4 status:

bit 7	bit 3	bit 2	bit 0
Reserved		CH4REPLAY	CH4LOGGING	CH4RUNNING

- Bits [7:3]: Reserved
- Bit 2: CH4REPLAY
- Bit 1: CH4LOGGING
- Bit 0: CH4RUNNING

3.7.8.Read ADC Inputs

MessageId = 0x7B

Read voltage on all the analogue inputs. Values are 14-bit millivolt values that are stored back-to-back in the response. Refer to following table.

Request: No data

Response:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6
AN0 8 bits	AN0 6 bits + AN1 2 bits	AN1 8 bits	AN1 4 bits + AN2 4 bits	AN2 8 bits	AN2 2 bits + AN3 6 bits	AN3 8 bits

DATA0:

bit 7							bit 0
AN0_7	AN0_6	AN0_5	AN0_4	AN0_3	AN0_2	AN0_1	AN0_0

DATA1:

bit 7							bit 0
AN1_1	AN1_0	AN0_13	AN0_12	AN0_11	AN0_10	AN0_9	AN0_8

DATA2:

bit 7							bit 0
AN1_9	AN1_8	AN1_7	AN1_6	AN1_5	AN1_4	AN1_3	AN1_2

DATA3:

bit 7	bit 0

AN2_3	AN2_2	AN2_1	AN2_0	AN1_13	AN1_12	AN1_11	AN1_10

DATA4:

bit 7							bit 0
AN2_11	AN2_10	AN2_9	AN2_8	AN2_7	AN2_6	AN2_5	AN2_4

DATA5:

bit 7							bit 0
AN3_5	AN3_4	AN3_3	AN3_2	AN3_1	AN3_0	AN2_13	AN2_12

DATA6:

bit 7							bit 0
AN3_13	AN3_12	AN3_11	AN3_10	AN3_9	AN3_8	AN3_7	AN3_6

3.7.9. Write DAC Outputs

MessageId = 0x7C

Write value to a DAC. This cannot be done if this some **running** SENT channel is mapped to this DAC output. In order to keep the output forced, this message must be issued periodically (there is a timeout of 5 seconds).

Request:

DATA 0	DATA 1 – DATA 2
Channel number	Value

Channel number is the same as in request.
Value: 12-bit unsigned millivolt voltage, which is sent LSB first. Maximum value is 4095 mV.
Note that if Value = 0xFFFF, DAC is powered down.

Response:

DATA 0
Channel number

Acknowledgment when success, error response when something went wrong – wrong channel selected or some running SENT channel is mapped to it.

3.8. SENT Extended Configuration

Using those messages, you can configure SENT I/O settings, alternative CAN IDs, SENT logging and RCNT configuration.
An analogue output channel (12-bit, 0 - 4.095 V) can be mapped on any RX SENT channel. Bit position and bit length within the SENT Data Nibbles is configurable by the user. So is the linear transfer function and voltage Min and Max limits. There are four such analogue outputs. Furthermore, it is possible to map analogue inputs to SENT channel. Configuration parameters are similar to output configuration except there are no Min and Max limits.

3.8.1. Read and Write DAC Configuration

MessageId = 0x80 for read, 0x81 for write

Request 0x80:

DATA 0
DAC channel number (0 to 3)

Response:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6
DAC + SENT channel	DAC configuration register 1	DAC configuration register 2	DAC offset LSB	DAC offset MSB	DAC multiplier LSB	DAC multiplier MSB

DAC and SENT channel number:

bit 7							bit 0
Reserved	Reserved	SENTCH2	SENTCH1	SENTCH0	DACCH2	DACCH1	DACCH0

- Bits [7:6]: Reserved
- Bits [5:3]: **SENTCH** – SENT channel used for output mapping.
  - 0 – Disabled (High Z)
  - 1 – SENT 1 (IO mapped to SENT1)
  - 2 – SENT 2 (IO mapped to SENT2)
  - 3 – SENT 3 (IO mapped to SENT3)
  - 4 – SENT 4 (IO mapped to SENT4)
  - 5…7 – Reserved
- Bits [2:0]: **DACCH** – DAC channel which configuration read was requested
  - 0 – IO1
  - 1 – IO2
  - 2 – IO3
  - 3 – IO4
  - 4…7 - Reserved

DAC configuration register 1:

bit 7							bit 0
Reserved	Reserved	NIBBLEORDER	STARTBIT4	STARTBIT3	STARTBIT2	STARTBIT1	STARTBIT0

- Bits [7:6]: Reserved
- Bit 5: **NIBBLEORDER** (endianness)

- 0 – Big-endian (MSN first)
- 1 – Little-endian (LSN first)

- Bits [4:0]: **STARTBIT** – start bit in the Data Field of a SENT frame

DAC configuration register 2:

bit 7							bit 0
Reserved	Reserved	LENGTH5	LENGTH4	LENGTH3	LENGTH2	LENGTH1	LENGTH0

- Bits [7:6]: Reserved
- Bits [5:0]: **LENGTH** – Bit length of used data in the Data Field

DAC offset: Offset value that is sent LSB first. For more information see equation below.
DAC multiplier: Multiplier value (sent LSB first). For more information see equation below.

This equation applies to the voltage on analogue output:

$$U_{out} = \frac{RawValue * Multiplier}{1024} + Offset\ [mV]$$

Apart from the physical range of the DAC (which is 0 to 4.095 V), the voltage range can further be limited by software. See below for voltage limits.

Request 0x81 has the same structure as response to 0x80:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6
DAC and SENT channel number	DAC configuration register 1	DAC configuration register 2	DAC offset LSB	DAC offset MSB	DAC multiplier LSB	DAC multiplier MSB

Response:

DATA 0
Channel number

Acknowledgment when success, error response when something went wrong (wrong channel selected).

3.8.2.Read and Write DAC Limits

MessageId = 0x82 for read, 0x83 for write

This limits the range of the analogue channel.

Request 0x82:

DATA 0
DAC channel number (0 to 3)

Response:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4

DAC channel number	Minimum voltage LSB	Minimum voltage MSB	Maximum voltage LSB	Maximum voltage MSB

DAC channel number: Channel number which configuration read was requested.
Minimum voltage: 16bit value in millivolts determining the minimum voltage on the output.
Maximum voltage: 16bit value in millivolts determining the maximum voltage on the output.
Those parameters are independent from the parameters configured in the message writing the DAC configuration.

Request 0x83 for writing the limits has exactly the same structure as response to 0x82:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
DAC channel number (0 to 3)	Minimum voltage LSB	Minimum voltage MSB	Maximum voltage LSB	Maximum voltage MSB

Response:

DATA 0
Channel number

Acknowledgment when success, error response when something went wrong (wrong channel selected).

3.8.3.Read and Write ADC Configuration

MessageId = 0x84 for read, 0x85 for write

Request 0x84:

DATA 0
ADC channel number (0 to 3) and operation

ADC channel number and operation:

bit 7							bit 0
Reserved	OPERATION	Reserved	Reserved	Reserved	ADCCH2	ADCCH1	ADCCH0

- Bit [7]: Reserved
- Bit [6]: **OPERATION** – Request configuration registers and offset or multiplier
  - 0 – Configuration registers and offset
  - 1 – Multiplier
- Bits [5:3]: Reserved
- Bits [2:0]: **ADCCH** – ADC channel which configuration read was requested
  - 0 – IO1
  - 1 – IO2
  - 2 – IO3
  - 3 – IO4
  - 4…7 - Reserved

Response is either configuration registers and offset or multiplier.
Response when OPERATION == 0:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
ADC and SENT channel number; operation	ADC configuration register 1	ADC configuration register 2	Offset LSB	Offset MSB

ADC and SENT channel number; operation:

bit 7							bit 0
Reserved	OPERATION	SENTCH2	SENTCH1	SENTCH0	ADCCH2	ADCCH1	ADCCH0

- Bit [7]: Reserved
- Bit [6]: **OPERATION** – Response data bytes meaning; 0 in this case
  - 0 – Configuration registers and offset
  - 1 – Multiplier
- Bits [5:3]: **SENTCH** – SENT channel used for output mapping.
  - 0 – Input is disabled
  - 1 – SENT 1 (IO mapped to SENT1)
  - 2 – SENT 2 (IO mapped to SENT2)
  - 3 – SENT 3 (IO mapped to SENT3)
  - 4 – SENT 4 (IO mapped to SENT4)
  - 5…7 – Reserved
- Bits [2:0]: **ADCCH** – ADC channel which configuration read was requested
  - 0 – IO1
  - 1 – IO2
  - 2 – IO3
  - 3 – IO4
  - 4…7 - Reserved

ADC configuration register 1:

bit 7							bit 0
Reserved	Reserved	NIBBLEORDER	STARTBIT4	STARTBIT3	STARTBIT2	STARTBIT1	STARTBIT0

- Bits [7:6]: Reserved
- Bit 5: **NIBBLEORDER** (endianness)
  - 0 – Big-endian (MSN first)
  - 1 – Little-endian (LSN first)
- Bits [4:0]: **STARTBIT** – start bit in the Data Field of a SENT frame

ADC configuration register 2:

bit 7							bit 0
Reserved	Reserved	LENGTH5	LENGTH4	LENGTH3	LENGTH2	LENGTH1	LENGTH0

- Bits [7:6]: Reserved
- Bits [5:0]: **LENGTH** – Bit length of used data in the Data Field

Response when OPERATION == 1:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4

ADC and SENT channel number; operation	Multiplier LSB	Multiplier [1]	Multiplier [2]	Multiplier MSB

ADC and SENT channel number; operation:

bit 7							bit 0
Reserved	OPERATION	Reserved	Reserved	Reserved	ADCCH2	ADCCH1	ADCCH0

- Bit [7]: Reserved
- Bit [6]: **OPERATION** – Response data bytes meaning; 1 in this case
  - 0 – Configuration registers and offset
  - 1 – Multiplier
- Bits [5:3]: Reserved
- Bits [2:0]: **ADCCH** – ADC channel which configuration read was requested
  - 0 – IO1
  - 1 – IO2
  - 2 – IO3
  - 3 – IO4
  - 4…7 - Reserved

ADC multiplier: Multiplier value (sent LSB first). This value is an IEEE 754 floating-point value. For more information see equation below.

Request for write: write is separated to two operations similarly to read command. First command is for configuration registers and offset and second command is for multiplier.

Write request 1:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
ADC and SENT channel number; operation	ADC configuration register 1	ADC configuration register 2	Offset LSB	Offset MSB

ADC and SENT channel number; operation:

bit 7							bit 0
Reserved	OPERATION	SENTCH2	SENTCH1	SENTCH0	ADCCH2	ADCCH1	ADCCH0

- Bit [7]: Reserved
- Bit [6]: **OPERATION** – Response data bytes meaning; 0 in this case
  - 0 – Configuration registers and offset
  - 1 – Multiplier
- Bits [5:3]: **SENTCH** – SENT channel used for output mapping.
  - 0 – Input is disabled
  - 1 – SENT 1 (IO mapped to SENT1)
  - 2 – SENT 2 (IO mapped to SENT2)
  - 3 – SENT 3 (IO mapped to SENT3)
  - 4 – SENT 4 (IO mapped to SENT4)
  - 5…7 – Reserved
- Bits [2:0]: **ADCCH** – ADC channel which configuration read was requested

- 0 – IO1
- 1 – IO2
- 2 – IO3
- 3 – IO4
- 4…7 - Reserved

Meaning of other data bytes is the same as in read command.

Response:

DATA 0
ADC channel number and operation (same as in read request)

Write request 2:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
ADC and SENT channel number; operation	Multiplier LSB	Multiplier [1]	Multiplier [2]	Multiplier MSB

ADC and SENT channel number; operation:

bit 7							bit 0
Reserved	OPERATION	SENTCH2	SENTCH1	SENTCH0	ADCCH2	ADCCH1	ADCCH0

- Bit [7]: Reserved
- Bit [6]: **OPERATION** – Data bytes meaning; 1 in this case
  - 0 – Configuration registers and offset
  - 1 – Multiplier
- Bits [5:3]: **SENTCH** – SENT channel used for output mapping.
  - 0 – Input is disabled
  - 1 – SENT 1 (IO mapped to SENT1)
  - 2 – SENT 2 (IO mapped to SENT2)
  - 3 – SENT 3 (IO mapped to SENT3)
  - 4 – SENT 4 (IO mapped to SENT4)
  - 5…7 – Reserved
- Bits [2:0]: **ADCCH** – ADC channel which configuration read was requested
  - 0 – IO1
  - 1 – IO2
  - 2 – IO3
  - 3 – IO4
  - 4…7 - Reserved

Response:

DATA 0
ADC channel number and operation (same as in read request)

Following equation applies to conversion:

$$RawValue = \frac{U_{in} - Offset}{Multiplier}$$

Where RawValue is the resulting value that is sent on SENT.
Note that when you use this mapping and RCNT simultaneously on the same frame bits, RCNT has always priority.

3.8.4.Read and Write SENT Logging Configuration

MessageId = 0x86 for read, 0x87 for write

Read configuration of SENT file logging. Used when MicroSD card is connected to the device. Logging is configurable, you can select frequency of logging and which frames you want to save.
Read request:

DATA 0
Channel number

SENT channel number (0 to 3).

Response:

DATA 0	DATA 1
Channel number	Logging configuration register 1

Channel number is SENT channel number.

Logging configuration register 1:

bit 7	bit 4				bit 0
Reserved	SLOWCHANNELLOG1	SLOWCHANNELLOG0	LOGMODE1	LOGMODE0	ENABLE

- Bits [7:5]: Reserved
- Bits [4:3]: **SLOWCHANNELLOG**
  - 0: Log fast channel frames only
  - 1: Log slow channel frames only
  - 2: Log both fast and slow channel frames
- Bits [2:1]: **LOGMODE** – logging filtering setting
  - 0: As fast as possible
  - 1: Fixed 10 ms
  - 2: Fixed 100 ms
  - 3: On change + 1 s
- Bit 0: **ENABLE**
  - 0: Logging disabled
  - 1: Logging to file enabled when MicroSD card is inserted. Note that logging is really started after device restart or after channel is started. In other words, this bit does not tell if logging is really running, this is determined by Channel Status.

Write request:

DATA 0	DATA 1

Channel number	Logging configuration register 1

Write command has exactly the same structure as response to read described above.

Response:

DATA 0
Channel number

SENT channel number (0 to 3) on success, Error Response otherwise.

3.8.5.SENT Rolling Counter Configuration

MessageId = 0x88

Using this messageId, you can enable a configurable rolling counter (RCNT) in the SENT fast frames. Note that the RCNT configuration has the priority of its data field over a standard Transmit Fast Frame. Hence Transmit Fast Frame will not overwrite the RCNT data field when the RCNT is active. Once the channel is stopped, RCNT is disabled.

Request:

DATA 0	DATA 1	DATA 2
SENT channel number	RCNT configuration register 1	RCNT configuration register 2

SENT channel number:

bit 7							bit 0
Reserved	Reserved	Reserved	Reserved	Reserved	SENTCH2	SENTCH1	SENTCH0

- Bits [3:0]: **SENTCH** – SENT channel used for output mapping.
  - 0 – SENT1
  - 1 – SENT2
  - 2 – SENT3
  - 3 – SENT4
  - 4…7 - Reserved

RCNT configuration register 1:

bit 7							bit 0
Reserved	ENABLE	NIBBLEORDER	STARTBIT4	STARTBIT3	STARTBIT2	STARTBIT1	STARTBIT0

- Bits [7:6]: Reserved
- Bit 6: **ENABLE** – Enabling / disabling of the RCNT
  - 0 - RCNT on this channel is disabled
  - 1 – RCNT on this channel is enabled
- Bit 5: **NIBBLEORDER** (endianness)
  - 0 – Big-endian (MSN first)
  - 1 – Little-endian (LSN first)
- Bits [4:0]: **STARTBIT** – start bit in the Data Field of a SENT frame

RCNT configuration register 2:

bit 7	bit 0

Reserved	Reserved	LENGTH5	LENGTH4	LENGTH3	LENGTH2	LENGTH1	LENGTH0

- Bits [7:6]: Reserved
- Bits [5:0]: Bit length of used data in the Data Field
    - Max. value: 32

Response:

DATA 0
Channel number

Acknowledgment if configuration written successfully.

3.8.6.Start / Stop SENT log playback

MessageId = 0x89 for start, 0x8A for stop

SENT log files can be replayed – logged frames are transmitted on the bus. Files can be selected by their number (numbered from 0). File list can be obtained from the device server's website, file count can be read using appropriate messageId from the protocol. Note that when playback is running, frame echo is turned off.

Request for start:

DATA 0	DATA 1	DATA 2
SENT Channel bit field	File index LSB	File index MSB

SENT channel bit field:

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Reserved	Reserved	Reserved	Reserved	SENT channel 3 enable	SENT channel 2 enable	SENT channel 1 enable	SENT channel 0 enable

- Bits [7:4]: Reserved
- Bit 3: SENT channel 3 playback enable
- Bit 2: SENT channel 2 playback enable
- Bit 1: SENT channel 1 playback enable
- Bit 0: SENT channel 0 playback enable

Note that channels, where you want to enable the playback, must be configured as TX in channel configuration.

Response:

DATA 0
Channel number

Acknowledgment if file playback was started successfully. Possible reasons for error: channel is running, channel is logging, no file with that number exists.

Request for stop:

DATA 0
Channel number

Response:

DATA 0
Channel number

Acknowledgment if file playback was stopped successfully. Possible reasons for error: playback was not started.

3.8.7. Read Number of Log Files

MessageId = 0x8B

Read how many log files is saved in the filesystem (used for playback).

Request: No data

Response:

DATA 0	DATA 1
File count LSB	File count MSB

Acknowledgment if file playback was stopped successfully. Possible reasons for error: playback was not started.

3.8.8.Read Playback Progress

MessageId = 0x8C

Read percentage progress of running playback.

Request: No data

Response:

DATA 0	DATA 1
Percentage	Status

Percentage: progress of playback. This is relevant only when playback was already started.
Status: status of the playback. 0 when no error, 1 when **last** playback was terminated because of some error. This is reset on next playback run

3.9. SENT Message Control

Those messages are used for manipulating the SENT frames – request their sending or acknowledge their reception.

3.9.1.Transmit SENT Fast Frame

MessageId = 0x90

Request transmission of SENT fast frame.
Depending on SENT channel configuration field *SwapFastDataNibbles*, the data nibbles can be swapped within a byte. Swapping is useful when the user needs to parse 12-bit values based on a DBC file.
It should be noted that *SwapFastDataNibbles* is considered for both directions. E.g., not only when a SENT frame is forwarded, but also when the user writes TX buffers for SENT transmission.

When sending a frame in the SPC mode (it is enabled by the bit *SPCEnabled* in the channel configuration), data are only updated in the buffer and are sent to bus only when correct master pulse is detected.
You do not have to always send the full message, but you must send all the data nibbles (in accordance with the parameter NibbleCount that is set in the channel configuration (number of data nibbles)).

Request when NibbleCount < 3:

DATA 0	DATA 1	DATA 2	DATA 3
SENT channel	Status nibble and data count	Data nibble 0 and 1	CRC

Request when 5 > NibbleCount >= 3:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
SENT channel	Status nibble and data count	Data nibble 0 and 1	Data nibble 2 and 3	CRC

Request when 7 > NibbleCount >= 5:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5
SENT channel	Status nibble and data count	Data nibble 0 and 1	Data nibble 2 and 3	Data nibble 4 and 5	CRC

Request when NibbleCount >= 7:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6
SENT channel	Status nibble and data count	Data nibble 0 and 1	Data nibble 2 and 3	Data nibble 4 and 5	Data nibble 6 and 7	CRC

SENT channel: SENT channel number where to transmit the frame (0 to 3).
Status nibble and data count:

bit 7							bit 0
NIBBLECNT3	NIBBLECNT2	NIBBLECNT1	NIBBLECNT0	STATUS3	STATUS2	STATUS1	STATUS0

- Bits [7:4]: **NIBBLECNT** – Number of data nibbles that follow
- Bits [3:0]: **STATUS** – SENT status nibble

Data nibble 0 and 1 if SwapFastDataNibbles is 0:

bit 7							bit 0
NIBBLE1_3	NIBBLE1_2	NIBBLE1_1	NIBBLE1_0	NIBBLE0_3	NIBBLE0_2	NIBBLE0_1	NIBBLE0_0

Data nibble 0 and 1 if SwapFastDataNibbles is 1:

bit 7							bit 0
NIBBLE0_3	NIBBLE0_2	NIBBLE0_1	NIBBLE0_0	NIBBLE1_3	NIBBLE1_2	NIBBLE1_1	NIBBLE1_0

- Bits [7:4]: **NIBBLE1 or NIBBLE0** – Nibble 1 or nibble 0 data
- Bits [3:0]: **NIBBLE0 or NIBBLE1** – Nibble 0 or nibble 1 data

…, data nibble 2 and 3, data nibble 4 and 5, …

Data nibble 6 and 7 if SwapFastDataNibbles is 0:

bit 7							bit 0
NIBBLE7_3	NIBBLE7_2	NIBBLE7_1	NIBBLE7_0	NIBBLE6_3	NIBBLE6_2	NIBBLE6_1	NIBBLE6_0

Data nibble 6 and 7 if SwapFastDataNibbles is 1:

bit 7							bit 0
NIBBLE6_3	NIBBLE6_2	NIBBLE6_1	NIBBLE6_0	NIBBLE7_3	NIBBLE7_2	NIBBLE7_1	NIBBLE7_0

- Bits [7:4]: **NIBBLE7 or NIBBLE6** – Nibble 7 or nibble 6 data
- Bits [3:0]: **NIBBLE6 or NIBBLE7** – Nibble 6 or nibble 7 data

CRC:

bit 7							bit 0
CRCCALC3	CRCCALC2	CRCCALC1	CRCCALC0	CRC3	CRC2	CRC1	CRC0

- Bits [7:4]: **CRCCALC:** Reserved here, no meaning
- Bits [3:0]: **CRC** – Frame CRC. If CRCMODE in SENT channel configuration is set to "Software CRC", frame is transmitted with this CRC. Otherwise not relevant.

Response:

DATA 0
Channel number

Acknowledgment when frame transmission successfully requested, Error Response otherwise.

3.9.2. Write SENT Slow Buffer without Multiplexing

MessageId = 0x91

This messageId starts transmission of SENT Slow frame in the fast frames. It is used for transmitting only one slow message. If you want a cyclical transmission of several slow messages, see 3.9.3 Write SENT Slow Buffer with Multiplexing.

Request:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
SENT channel	Message Id	Data LSB	Data MSB	Frame info

SENT channel: SENT channel number where to transmit the frame (0 to 3).
Message Id: SENT slow message Id.
Data: SENT slow frame data.
Frame info:

bit 7							bit 0
CONFIG	RESERVED	CRC4	CRC3	CRC2	CRC1	CRC1	CRC0

- Bit 7: **CONFIG** – Configuration Bit for Enhanced Serial Format. Relevant when Enhanced serial mode turned on in SENT Basic Configuration.
  - 0 – 8-bit Message Id, 12-bit data
  - 1 – 4-bit Message Id, 16-bit data
- Bit 6: Reserved
- Bits [5:0]: **CRC** – Slow Frame CRC

Response:

DATA 0
Channel number

Acknowledgment when frame transmission successfully requested, Error Response otherwise.

3.9.3. Write SENT Slow Buffer with Multiplexing

MessageId = 0x92

The gateway allows a cyclic transmission of Slow messages with different Message Ids. This is useful for sensor simulations when the user needs to multiplex between several Slow messages.
By enabling several message buffers, the particular SENT TX channel will automatically transmit the Slow messages in a cyclic order. The *SlowChannelMode* in SENT channel configuration has to be set to either Short serial or Enhanced serial mode.

Request:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
SENT channel number (0 to 3)	Buffer settings	Slow frame Message Id	Data LSB	Data MSB

Buffer settings:

bit 7							bit 0
Reserved	CONFIG	ENABLE	INDEX4	INDEX3	INDEX2	INDEX1	INDEX0

- Bit 6: **CONFIG** – Configuration bit for Enhanced serial format (not relevant for Short serial format – see channel settings)
  - 0 – 8-bit Message Id, 12-bit data (or Short serial frame)
  - 1 – 4-bit Message Id, 16-bit data
- Bit 5: **ENABLED –** Buffer enable bit
- Bits [4:0]: **INDEX** – Buffer index

Slow frame Message Id: Slow message identifier. Not relevant when buffer is not enabled.
Slow frame data. It is in the Intel coding (LSB first). Not relevant when buffer is not enabled.

A message buffer can be enabled or disabled even whilst a SENT channel is running. Transmitting a single slow message (Write SENT Slow Buffer without Multiplexing) will switch back to single slow message buffer mode even when at least one message buffer above is enabled.
All message buffers are cleared when a SENT channel is stopped. The user can configure the buffers before a SENT channel is started so that a cyclic transmission begins once the first Fast message is sent (Transmit SENT Fast Frame).

Response:

DATA 0
Channel number

Acknowledgment when frame buffer successfully written, Error Response otherwise.

3.9.4.Begin SENT/SPC Reception

MessageId = 0x93

Start the SPC reception (channel acts as SENT/SPC master).

Request:

DATA 0	DATA 1
SENT channel number (0 to 3)	SPC reception mode

SPC reception mode:
Buffer settings:

bit 7	bit 1	bit 0
Reserved		SPCRXMODE

- Bit 0: **SPCRXMODE** – SPC master reception mode (periodical or one-shot)
  - 0 – One-shot sending. There is generated one master pulse.
  - 1 – Periodical sending. Master pulse is generated with the configured period. Period must be set in the SENT/SPC channel configuration.

Response:

DATA 0
Channel number

Acknowledgment when pulse generation was successfully requested, Error Response otherwise.

Stopping SENT/SPC reception – if periodical pulse generation was started (SPCRXMODE set to one), it can be stopped by issuing command for requesting one-shot sending (SPCRXMODE set to zero). Device will generate one additional pulse and then stop.

3.9.5.SENT Fast Frame Reception

MessageId = 0x95

Device sends message with this Id asynchronously after it received a SENT fast frame. For CAN, it has the same structure as Transmit SENT Fast Frame. For Ethernet, USB, and CAN FD, a timestamp of message reception is added. Length of the response depends on the NIBBLECNT field - number of received data nibbles. Note that *SwapFastDataNibbles* channel parameter still applies here (if it is set to one, nibbles are swapped within a byte). See section Starting and Stopping SENT Channels to see which channels are used for frame echo. Note that for CAN FD, frames are padded with 0xFF to a valid CAN FD length.

CAN response when NIBBLECNT < 3:

DATA 0	DATA 1	DATA 2	DATA 3
SENT reception channel	Status nibble and data count	Data nibble 0 and 1	CRC

Other channels' response when NIBBLECNT < 3:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4 – DATA 11
SENT reception channel	Status nibble and data count	Data nibble 0 and 1	CRC	Timestamp bytes 0 – 7

CAN response when 5 > NIBBLECNT >= 4:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4
SENT reception channel	Status nibble and data count	Data nibble 0 and 1	Data nibble 2 and 3	CRC

Other channels' response when 5 > NIBBLECNT >= 4:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5 – DATA 12
SENT reception channel	Status nibble and data count	Data nibble 0 and 1	Data nibble 2 and 3	CRC	Timestamp bytes 0 – 7

CAN response when 7 > NIBBLECNT <= 5:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5
SENT reception channel	Status nibble and data count	Data nibble 0 and 1	Data nibble 2 and 3	Data nibble 4 and 5	CRC

Other channels' response when 7 > NIBBLECNT <= 5:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6 – DATA 13
SENT reception channel	Status nibble and data count	Data nibble 0 and 1	Data nibble 2 and 3	Data nibble 4 and 5	CRC	Timestamp bytes 0 – 7

CAN response when NIBBLECNT >= 7:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6
SENT reception channel	Status nibble and data count	Data nibble 0 and 1	Data nibble 2 and 3	Data nibble 4 and 5	Data nibble 6 and 7	CRC

Other channels' response when NIBBLECNT >= 7:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6	DATA 7 – DATA 14
SENT reception channel	Status nibble and data count	Data nibble 0 and 1	Data nibble 2 and 3	Data nibble 4 and 5	Data nibble 6 and 7	CRC	Timestamp bytes 0 – 7

Data fields are discussed in detail in section 3.9.1 Transmit SENT Fast Frame.
Only difference is the CRC field and timestamp:
CRC:

bit 7							bit 0
CRCCALC3	CRCCALC2	CRCCALC1	CRCCALC0	CRC3	CRC2	CRC1	CRC0

- Bits [7:4]: **CRCCALC –** CRC value that should have been received. It is computed by the device.
- Bits [3:0]: **CRC** – Frame CRC value received from the bus.

Timestamp: 64-bit number representing time in microseconds from channel start. It is sent LSB first. Timestamp is not present on CAN.

3.9.6.SENT Slow Frame Reception

MessageId = 0x96

Device sends message with this Id asynchronously after it received a SENT slow frame. For CAN, it has the same structure as request for Write SENT Slow Buffer without Multiplexing, except there is one data byte more – CRC calculated. This is true for CAN; for CAN FD and other channels, there is added a timestamp.

CAN response:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5
SENT reception channel	Message Id	Data LSB	Data MSB	Frame info	CRC calculated

Other channels' response:

DATA 0	DATA 1	DATA 2	DATA 3	DATA 4	DATA 5	DATA 6 – DATA 13
SENT reception channel	Message Id	Data LSB	Data MSB	Frame info	CRC calculated	Timestamp (LSB first)

First four bytes are the same as request Write SENT Slow Buffer without Multiplexing.
Frame info differs in Format bit:

bit 7							bit 0
FORMAT	FRAMETYPE	CRC4	CRC3	CRC2	CRC1	CRC1	CRC0

- Bit 7: **FORMAT** – Enhanced serial format configuration bit
  - 0 – 8-bit Message Id, 12-bit data (or Short serial frame, see the next bit)
  - 1 – 4-bit Message Id, 16-bit data
- Bit 6: **FRAMETYPE** – Slow frame type
  - 0 – Short serial
  - 1 – Enhanced serial
- Bits [5:0]: **CRC** – Slow Frame CRC

CRC calculated:

bit 7 bit 6	bit 5					bit 0
Reserved	CRCCALC5	CRCCALC4	CRCCALC3	CRCCALC2	CRCCALC1	CRCCALC0

- Bits [7:6]: Reserved
- Bits [5:0]: **CRCCALC** – CRC value that should have been received. It is computed by the device.

Timestamp: 64-bit number representing time in microseconds from channel start. It is sent LSB first. Timestamp is not present on CAN, it is present on CAN FD and other channels.

3.9.7.SENT Fast Frame Error

MessageId = 0x97

Device transmits message with this Id asynchronously if there is some error in the fast SENT frame. On CAN, it is two-byte, on CAN FD and other channels, it is ten-byte.

CAN response:

DATA 0	DATA 1
SENT reception channel	Error code and type

Other channels' response:

DATA 0	DATA 1	DATA 2 – DATA 9
SENT reception channel	Error code and type	Timestamp

SENT reception channel: SENT channel number where the error was detected
Error code and type:

bit 7							bit 0
Reserved	Reserved	ERR TYPE1	ERR TYPE0	FRMERR CODE3	FRMERR CODE2	FRMERR CODE1	FRMERR CODE0

- Bits [5:4]: **ERRTYPE** – Error type
  - 0 – CRC value mismatch
  - 1 – Framing error – nibble has incorrect length (it is shorter than 12 UT or longer than 27 UT)
  - 2 – Adjacent sync error – calibration pulse has a valid length of 56 UT ± 20%, but differs too much from the previous one (successive calibration pulses must not differ for more than 1.5625%)
  - 3 – SENT bus error – wrong sync value – calibration pulse was expected but device detected pulse of incorrect length (different from 56 UT ± 20%). These errors are filtered out when connecting the device to already running SENT bus
- Bits [3:0]: **FRMERRCODE** – Framing error code. Relevant when ERRTYPE is Framing error. Localizes the error within frame
  - 1 – Status nibble
  - 2 – Data nibble 0 error
  - 3 – Data nibble 1 error
  - …
  - 9 – Data nibble 7 error
  - 10 – CRC nibble error

Timestamp: 64-bit number representing time in microseconds from channel start. It is sent LSB first. Timestamp is not present on CAN.

NOTE that when SENT channel has configuration field CRCMODE in configuration register 2 set to HW CRC off (value 0), incorrect CRC in received Fast Frame does not generate an error.

3.9.8.SENT Slow Frame Error

MessageId = 0x98

Device transmits message with this Id asynchronously if there is some error in the slow SENT frame. On CAN it is two-byte, on CAN FD and other channels, it is ten-byte (there is a timestamp added).

CAN Response:

DATA 0	DATA 1
SENT reception channel	Error type

Other channels' response:

DATA 0	DATA 1	DATA 2 – DATA 9
SENT reception channel	Error type	Timestamp

SENT reception channel: SENT channel number where the error was detected
Error type:

bit 7							bit 0
Reserved	Reserved	ERRTYPE1	ERRTYPE0	Reserved	Reserved	Reserved	Reserved

- Bits [7:6]: Reserved
- Bits [5:4]: **ERRTYPE** – Error type
  - 0 – CRC
  - 1 – Framing
  - 2 – Sync
- Bits [3:0]: Reserved

Timestamp is 64-bit number representing time in microseconds since channel start. It is sent LSB first. Timestamp is not present on CAN, it is present on CAN FD and other channels.

3.9.9.SENT Frame Transmission Acknowledgement

MessageId = 0x99

This message is sent asynchronously by the device (if appropriate echo is set) when there is a frame transmitted on the SENT bus. It has exactly the same format as 3.9.5 SENT Fast Frame Reception.

3.9.10. SENT Slow Frame Transmission Acknowledgement

MessageId = 0x9A

This message is sent asynchronously by the device (if appropriate echo is set) when there is a slow frame transmitted on the SENT bus. It has exactly the same format as 3.9.6 SENT Slow Frame Reception. *SwapFastDataNibbles* channel parameter applies here too.

3.10. Miscellaneous Messages

3.10.1. Restart Device

MessageID = 0xFD

Issuing this command makes the device restart. Restart is needed after changing IP address or port.

Request, response: No data

3.10.2. Restart Device to Bootloader

MessageId = 0xFE

This command restarts device to System Bootloader, so that new firmware can be loaded. It can be chosen which bootloader will be started: System Bootloader for connection via USB and STM32CubeProgrammer or HTTP bootloader for upload from web browser.

System Bootloader: Only USB cable and STM32CubeProgrammer is needed for flashing the device.
HTTP bootloader: Recommended web browser for firmware upload is Google Chrome. Uploaded file must be in the binary format (.bin).
Request:

DATA 0
Bootloader selection

- Bootloader selection: 0 = System Bootloader, 1 = Web bootloader

Response:
No data

4. Communication Examples

Examples in this chapter are for Ethernet and USB that use a two-byte data length field.

4.1. Device Settings

Command	Bytes [hex]
Read SN	02 11 00 00 11 03 Example response: 02 11 04 00 00 01 02 03 1B 03 Serial number is 03020100 Bytes explanation - request: 02 – STX 11 – ID 00 00 – DATALEN 11 – Checksum 03 – ETX Response: 02 – STX 11 – ID 04 00 – DATALEN 00 01 02 03 – DATA 1B – Checksum 03 – ETX
Read MAC address	02 1B 00 00 1B 03 Example response: 02 1B 06 00 A7 19 6E C2 A5 FC B2 03 MAC address is A7:19:6E:C2:A5:FC
Write Ethernet settings IP address: 192.168.1.101 Mask: 24 (255.255.255.0) Port: 8001	02 16 07 00 C0 A8 01 65 18 41 1F 63 03 Response: 02 16 00 00 16 03
Write default gateway IP address: 192.168.1.100	02 1D 04 00 C0 A8 01 64 EE 03 Response: 02 1D 00 00 1D 03
Restart device	02 FD 00 00 FD 03

4.2. CAN communication (device acting as CAN interface)

Command	Bytes [hex]
Write CAN configuration Channel index: 0 (0x00) Protocol CAN 2.0, auto start false, normal ack, arbitration sample point: 80 % (0x08) Arbitration baud rate: 1 MBaud (0x03) Arbitration SJW: 1 (0x00)	02 60 06 00 00 08 03 00 FF FF 6F 03 Response: 02 60 01 00 00 61 03
Configure CAN echo Rx echo: ON Tx echo: ON	02 66 02 00 00 03 6B 03 Response: 02 66 01 00 00 67 03
Start CAN channel	02 67 01 00 00 68 03 Response: 02 67 01 00 00 68 03
Transmit CAN frame Channel index: 0 (0x00) FDF = ESI = BRS = RTR = EXTId = 0 (0x00) ID: 0x222	02 6A 0D 00 00 00 22 02 08 01 02 03 04 05 06 07 08 C7 03 Response: 02 6A 01 00 00 6B 03

DLC: 8 bytes Data: 0x01 0x02 0x03 0x04 0x05 0x06	
Asynchronous response when frame transmitted Channel index: 0 (0x00) FDF = ESI = BRS = RTR = EXTId = 0 (0x00) Timestamp: 2 115 042 µs (0x00000000002045E2) ID: 0x222 DLC: 8 bytes Data: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 Note: in order to receive this message, you must have other CAN device connected to the bus (or you will receive Acknowledge error)	No request Response: 02 6A 15 00 00 00 E2 45 20 00 00 00 00 00 22 02 08 01 02 03 04 05 06 07 08 16 03

4.3. CAN FD configuration (device acting as CAN FD interface)

Command	Bytes [hex]
Write CAN configuration Channel index: 0 (0x00) Protocol ISO CAN FD, auto start false, normal ack, arbitration sample point: 80 % (0x48) Arbitration baud rate: 500 kBaud (0x02) Arbitration SJW: 1 (0x00) Data baud rate: 2 MBaud, data SJW: 1 (0x10) Data sample point: 80 % (0x08)	02 60 06 00 00 48 02 00 10 08 C8 03 Response: 02 60 01 00 00 61 03
Configure CAN echo Rx echo: OFF Tx echo: ON	02 66 02 00 00 02 6A 03 Response: 02 66 01 00 00 67 03
Start CAN channel	02 67 01 00 00 68 03 Response: 02 67 01 00 00 68 03
Transmit CAN FD frame Channel index: 0 (0x00) FDF = BRS = 1, ESI = RTR = EXTId = 0 (0x14) ID = 0x333 DLC: 16 bytes Data: 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x00 0x00 0x00 0x00 0x00	02 6A 15 00 00 14 33 03 10 01 02 03 04 05 06 07 08 09 0A 0B 00 00 00 00 00 1B 03 Response: 02 6A 01 00 00 6B 03
Asynchronous response when frame transmitted Channel index: 0 (0x00) FDF = BRS = 1, ESI = RTR = EXTId = 0 (0x14) ID = 0x333 DLC = 16 bytes Data: same as in the preceding request	No request Response: 02 6A 1D 00 00 14 6E 9B 65 0A 00 00 00 00 33 03 10 01 02 03 04 05 06 07 08 09 0A 0B 00 00 00 00 00 00 9B 03

4.4. SENT communication (device acting as SENT gateway)

This example expects you have connected SENT1 RX <-> SENT2 TX (loopback). If you do everything correctly, you should be able to measure 767 mV on the IO1 output.

Command	Bytes [hex]
Write SENT1 configuration Channel index 0, swap fast frames disabled (0x00) Autostart on, direction Rx, HW CRC on, nibble count 6 (0x67) Pause pulse disabled, echo mode 100 ms, slow frame: short serial format, slow frame CRC fault off, SPC disabled (0x0A) Unit time 300 µs (0x012C) Pause pulse: 0 (not used)	02 71 07 00 00 67 0A 2C 01 00 00 16 03 Response: 02 71 01 00 00 72 03
Write SENT2 configuration Channel index 1, swap fast frames disabled (0x01) Configuration register 1: same as in previous message, only change is direction – Tx Configuration register 2, unit time and pause pulse same as in previous message	02 71 07 00 01 65 0A 2C 01 00 00 15 03 Response: 02 71 01 00 01 73 03
Save SENT configuration (all channels)	02 78 00 00 78 03 Response: 02 78 00 00 78 03
Write DAC configuration Channel: DAC1, SENT1 (0x08) Start bit: 4, nibble order big-endian (0x04) Bit length: 12 (0x0C) Offset: 256 (0x0100) Multiplier: 128 (0x0080)	02 81 07 00 08 04 0C 00 01 80 00 21 03 Response: 02 81 01 00 00 82 03
Start SENT channel 2 Valid if channel is not already running (autostart is on in default configuration), else error	02 74 01 00 00 75 03 Response: 02 74 01 00 00 75 03 (or 03 02 FF 02 00 F1 00 F2 03 if channel already running)
Transmit SENT frame (channel 2) SENT channel: SENT2 (0x01) Status nibble: 15, nibble count: 6 (0x6F) Data nibbles: 0x0 0xF 0xF 0xF 0x0 0x0 CRC not specified (it is computed by the board)	02 90 07 00 01 6F 00 FF 0F 00 00 15 03 Response: 02 90 01 00 01 92 03
Asynchronous response when frame transmitted (channel SENT2) Computed CRC nibble is 0xA	02 99 06 00 01 6F 00 FF 0F AA C7 03
Asynchronous response when frame received (channel SENT1) Computed and received CRC nibbles are both 0xA	02 95 06 00 00 6F 00 FF 0F AA C2 03
Send slow frame (without multiplexing) Channel SENT2, ID 5, Data 0x98	02 91 05 00 01 05 98 00 00 34 03 Response: 02 91 01 00 01 93 03
Asynchronous response when slow frame received Channel SENT1, ID 5, Data 0x98, CRC = CRC calculated = 0x01	02 96 06 00 00 05 98 00 01 01 3B 03

5. Contact

MACH SYSTEMS s.r.o.
www.machsystems.cz
info@machsystems.cz
Czech Republic

Company Registration: 29413893
VAT no.: CZ29413893