The jCOM1939 gateway is a high-performance, low-latency vehicle network adapter. It allows any host device with a serial COM port to monitor SAE J1939 data traffic and communicate with the SAE J1939 vehicle network. The gateway supports the full SAE J1939 protocol according to J1939/81 Network Management (Address Claiming) and J1939/21 Transport Protocol (TP).

The SAE J1939 Gateway Protocol is designed to establish a data exchange between a jCOM1939 gateway and a host system (e.g. a PC) per means of serial communication (RS232, USB). The protocol enables the host system to take advantage of the gateway’s capabilities by providing a set of commands from the host and responses from the gateway.

### jCOM1939 Gateway Operating Modes

The jCOM1939 gateway supports two operating modes to cover all possible scenarios. The operating mode can be set by the host system per command through the serial port as described in the next chapter.

The modes are:

1. **Listen-Only (Default):** This mode supports a mere J1939 data-monitoring mode, which does not require the use of a NAME and source address, i.e. there is no address claim process.
2. **Event:** Through this mode, the jCOM1939 gateway will report J1939 data as soon as they occur, which also requires that the receiving host system is capable of catching data in real-time.

The jCOM1939 gateway works per default as a mere monitoring device in *Listen-Only* mode, however, the gateway will, also per default, not report any data unless the host system (e.g. a PC under Windows/Linux) sets the corresponding filters (ADDFILTER command).

A full communication, i.e. bidirectional data transfer, between the host system and the J1939 vehicle network is only possible after the host system initiates the J1939 Address Claim process by sending the corresponding command (SETPARAM). The same command allows the host system to set the operating mode, either *Event* or *Polling*.

All operating modes allow the reception of up to 1785 bytes per message, while the *Event* operating mode also allows the transmission of up to 1785 bytes per message (BAM or RTS/CTS Session).

The *Event* operating mode engages the full Address Claim procedure, requiring a NAME, a preferred source address (SA), and an optional address range (in case the preferred SA is already taken).

The transmission of J1939 messages in the *Event* operating mode is always “real-time,” i.e. the message is being transmitted as soon as possible (as soon as the gateway proceeds during the bus arbitration process).

### Start-Up Sequence

The start-up sequences for each mode are:

**Listen-Only:**
1. Set all filters (ADDFILTER command).
2. Operation starts with setting the first filter.

**Event:**
1. Initiate Address Claim Process and set operating mode using the SETPARAM or SETPARAM1 command.
2. When using the SETPARAM1 command, the gateway responds with REPSTATUS message.
3. After successful address claim (check status per REPSTATUS message), data transmission is enabled.
4. Set all filters (ADDFILTER command).
5. Receiving messages is enabled with setting the first filter.

**jCOM1939 Gateway - Serial Communications Protocol**

This chapter describes the message framing protocol used to define the start and body of messages sent between host device and the jCOM1939 gateway.

**Message Format**

The jCOM1939 protocol defines two special tokens, START and ESC. START has the value of 192 decimal and ESC has the value of 219 decimal.

Data is sent to and from the jCOM1939 gateway using the industry standard method of byte stuffing. Byte stuffing is the process of inserting additional tokens when the START or ESC tokens are part of the data field.

All messages start with a START token, and due to byte stuffing, a START token can never appear in any other part of the message. If the value of 192 is contained in the message length, data field, or checksum, an ESC token is inserted into the data stream followed by a 220, this indicates a value of 192. If an ESC token appears in the message length, data field, or checksum, an ESC token is inserted into the stream followed by a 221, this indicates a value of 219.

All messages must be byte stuffed, following the START token, by the transmitter and unstuffed by the receiver.

<table>
<thead>
<tr>
<th>Code</th>
<th>Dec / Hex</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>START</td>
<td>192 / 0xC0</td>
<td>Start of Message</td>
</tr>
<tr>
<td>ESC</td>
<td>219 / 0xDB</td>
<td>Stuffing Byte</td>
</tr>
<tr>
<td>START_STUFF</td>
<td>220 / 0xDD</td>
<td>Stuffing Byte</td>
</tr>
<tr>
<td>ESC_STUFF</td>
<td>221 / 0xDD</td>
<td>Stuffing Byte</td>
</tr>
<tr>
<td>DATA_START</td>
<td>ESC+START_STUFF</td>
<td>Data = 219, 220 / 0xDB, 0xDC</td>
</tr>
<tr>
<td>DATA_ESC</td>
<td>ESC+ESC_STUFF</td>
<td>Data = 219, 221 / 0xDB, 0xDD</td>
</tr>
</tbody>
</table>

To send a packet, the transmitting device starts by sending a START character followed by the size of the data to follow, along with the data and the checksum of the packet.

The message length is a 2-byte field (MSB first, followed by LSB) equal to the size of the data field plus checksum, both before byte stuffing.

The checksum is a 2’s complement checksum over the length and data fields and is calculated before byte stuffing has been added.
What is a 2’s Complement Checksum?

A checksum is a technique to check data for transmission errors or tampering. If the last few bytes are the sum of all the preceding bytes, then any errors are likely to be detected.

In this case, the checksum is the sum of all bytes between the data length field (MSB) and the last byte of the actual data field, ignoring any carry, meaning the checksum is always 8 bits long (0 to 255).

A one’s complement is just a complement: \( \sim x \)

A two’s complement is a complement and increment, ignoring carry: \((\sim x)+1\)

Messages

This chapter describes the commands sent to and received from the gateway.

Note: All message definitions are documented in their unstuffed representation. All highlighted sections represent the data field.

Every message is part of the data field, and it starts with a message identifier. This is an eight-bit value that tells the receiver what type of message follows. This is followed by one or more additional data field(s), depending on the message. The message identifiers, data fields, responses, etc. are documented for each message in the following chapters.

The available messages and their message IDs are:

<table>
<thead>
<tr>
<th>Command Message</th>
<th>ID</th>
<th>Meaning</th>
<th>Source</th>
<th>Acknowledgment</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>0</td>
<td>Acknowledgement</td>
<td>Gateway</td>
<td>-</td>
</tr>
<tr>
<td>ADDFILTER</td>
<td>1</td>
<td>Add Filter</td>
<td>Host</td>
<td>ACK</td>
</tr>
<tr>
<td>DELFILTER</td>
<td>2</td>
<td>Delete Filter</td>
<td>Host</td>
<td>ACK</td>
</tr>
<tr>
<td>TXDATA</td>
<td>3</td>
<td>Transmit Data</td>
<td>Host</td>
<td>ACK</td>
</tr>
<tr>
<td>RXDATA</td>
<td>4</td>
<td>Receive Data</td>
<td>Gateway</td>
<td>-</td>
</tr>
<tr>
<td>RESET</td>
<td>5</td>
<td>Reset jCOM1939</td>
<td>Host</td>
<td>ACK</td>
</tr>
<tr>
<td>HEART</td>
<td>6</td>
<td>Heartbeat</td>
<td>Gateway</td>
<td>-</td>
</tr>
<tr>
<td>SETPARAM</td>
<td>7</td>
<td>Set Protocol Parameters</td>
<td>Host</td>
<td>ACK</td>
</tr>
<tr>
<td>REQINFO</td>
<td>8</td>
<td>Request Information from gateway</td>
<td>Host</td>
<td>REPSTATUS or RXDATA or VERSION</td>
</tr>
<tr>
<td>REPSTATUS</td>
<td>9</td>
<td>Report J1939 Protocol Status</td>
<td>Gateway</td>
<td>-</td>
</tr>
<tr>
<td>FLASH</td>
<td>10</td>
<td>Starts the in-circuit programmer</td>
<td>Host</td>
<td>ACK</td>
</tr>
<tr>
<td>SETACK</td>
<td>11</td>
<td>Set Acknowledgement</td>
<td>Host</td>
<td>ACK</td>
</tr>
</tbody>
</table>
ACK – Acknowledgment

This message is sent by the jCOM1939 to acknowledge the correct receipt of a message from the host system. ACK will not be transmitted in cases where the checksum is wrong or byte-stuffing errors were detected.

Note:
- The ACK message can be suppressed through the SETACK message.

**Message Identifier:** 0  
**Total Length:** 6 bytes  
**Response:** -

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_ACK</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Message Id of acknowledged message</td>
<td>xx</td>
</tr>
<tr>
<td>5</td>
<td>Checksum</td>
<td>xx</td>
</tr>
</tbody>
</table>

ADDFILTER – Add Filter PGN

This message adds a PGN to the acceptance filter. The jCOM1939 gateway allows up to 80 message filters. Set the PGN to 0x100000 to allow all data without filtering; disable the feature by deleting the filter (DELFILTER command) using the same PGN.

Note:
- The jCOM1939 gateway will, per default, not pass on any data messages unless their associated PGN is set (ADDFILTER command).
- Setting filters referring to PGNs used for Address Claim or Network Management will be ignored, i.e. the gateway reports only data and request PGNs.
- To enable SAE J1939 request messages (PGN = 0xEAxx), it is sufficient to use the base PGN 0xEA00 (59904) or any PGN in the range from 0xEA00 to 0xEAFF, which will enable the entire range.

**Message Identifier:** 1  
**Total Length:** 8 bytes  
**Response:** ACK

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_FA</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>PGN MSB</td>
<td>xx</td>
</tr>
</tbody>
</table>
DELFILTER – Delete Filter PGN

This message deletes a PGN from the acceptance filter. Set the PGN to 0x100000 to delete all filters.

**Note:**
- The gateway will, per default, not pass on any data messages unless their associated PGN is set (ADDFILTER command).

**Message Identifier:** 2  
**Total Length:** 8 bytes  
**Response:** ACK

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_FA</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>PGN MSB</td>
<td>xx</td>
</tr>
<tr>
<td>5</td>
<td>PGN 2ND</td>
<td>xx</td>
</tr>
<tr>
<td>6</td>
<td>PGN LSB</td>
<td>xx</td>
</tr>
<tr>
<td>7</td>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

TXDATA – Transmit Data

This message schedules a J1939 message for transmission. The message will be sent as soon as the J1939 network allows.

**Note:**
- The gateway will not transmit any data during the Address Claim process (applies only to the Event operating mode), which can last up to several hundreds of milliseconds.
- A J1939 message can be between 0 and 1785 bytes long. If a message is longer than 8 bytes, the jCOM1939 gateway will, depending on the destination address, manage the transmission per broadcast (BAM) or peer-to-peer (RTS/CTS) session.
- The jCOM1939 gateway will allow you the use of any source address, provided that the address claim process was finished successfully. It is highly recommended using the negotiated source address (obtained through the REPSTATUS message), but this feature is provided for maximum flexibility.

**Message Identifier:** 3  
**Total Length:** Variable (19 with 8 data bytes)  
**Response:** ACK

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>xx</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>xx</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_TX</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>PGN MSB</td>
<td>xx</td>
</tr>
</tbody>
</table>
TXDATAL – Transmit Data With Loopback

This message is designed and handles data transmission exactly as the regular TXDATA message. However, when the gateway receives the TXDATAL message, the data will be reflected back to the host system (which is a mandatory feature for gateway applications).

Message Identifier: 16
Total Length: Variable (19 with 8 data bytes)
Response: ACK

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>xx</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>xx</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_TX</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>PGN MSB</td>
<td>xx</td>
</tr>
<tr>
<td>5</td>
<td>PGN 2ND</td>
<td>xx</td>
</tr>
<tr>
<td>6</td>
<td>PGN LSB</td>
<td>xx</td>
</tr>
<tr>
<td>7</td>
<td>Destination Address</td>
<td>xx</td>
</tr>
<tr>
<td>8</td>
<td>Source Address</td>
<td>xx</td>
</tr>
<tr>
<td>9</td>
<td>Priority</td>
<td>0...7</td>
</tr>
<tr>
<td>10</td>
<td>Data Field Start</td>
<td>xx</td>
</tr>
</tbody>
</table>

TXDATAL – Transmit Data With Loopback

<table>
<thead>
<tr>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

RXDATA – Receive Data

This message is transmitted to the host device as soon as the jCOM1939 gateway receives a J1939 message (Event operating mode) or when requested per REQINFO message. The message can be between 0 and 1785 bytes long.

Note:
- The gateway will not forward any data as long as the filters are not set.
- The gateway will also not forward any data during the Address Claim process (applies only to the Event operating mode).

Message Identifier: 4
Total Length: Variable (19 with 8 data bytes)
Response: -
**RESET – Gateway Reset**

This message requests a gateway reset, causing a re-initialization of all SAE J1939 protocol parameters, which includes setting the operating mode to *Listen-Only*.

The purpose of the RESET command is for synchronizing the gateway with the host system. The gateway’s settings will not be affected by a host system power-down, meaning all filter settings plus negotiated address will still be active. To prevent any synchronization issues, it is therefore recommended to issue the RESET command as soon as the host system powers down or starts up.

**Note:**
- The gateway-reset command does not affect the heartbeat frequency or ACK message settings.

**Message Identifier:** 5  
**Total Length:** 8 bytes  
**Response:** ACK

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>xx</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>xx</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_RESET</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Reset Key 1</td>
<td>0xA5</td>
</tr>
<tr>
<td>5</td>
<td>Reset Key 2</td>
<td>0x69</td>
</tr>
<tr>
<td>6</td>
<td>Reset Key 3</td>
<td>0x5A</td>
</tr>
<tr>
<td>7</td>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

**HEART - Heartbeat**

This message is per default being transmitted with a one second frequency; it is being used as a so-called “watchdog” function to monitor the serial connection functionality.

The message contains the hardware and software versions in a form of A.BB.CC (for instance, 1.02.03):

- A - Major Update
Following the hardware and software versions are two error counters that allow an insight into the quality of the serial connection:

- Checksum Error
- Byte Stuffing Error

Please be aware that the range of these counters is from 0 to 255, where, at a count greater than 255, the counter will start at zero again. However, action should be taken as soon as a certain (project-specific) count has been reached.

The heartbeat message frequency can be adjusted through the SETHEART message, which also allows suppressing the heartbeat.

**Message Identifier:** 6  
**Total Length:** 13 bytes  
**Response:** -

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>08</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_HEART</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>HW_VERSIONMAJOR</td>
<td>xx</td>
</tr>
<tr>
<td>5</td>
<td>HW_VERSIONMINOR</td>
<td>xx</td>
</tr>
<tr>
<td>6</td>
<td>HW_VERSIONBUGFIX</td>
<td>xx</td>
</tr>
<tr>
<td>7</td>
<td>SW_VERSIONMAJOR</td>
<td>xx</td>
</tr>
<tr>
<td>8</td>
<td>SW_VERSIONMINOR</td>
<td>xx</td>
</tr>
<tr>
<td>9</td>
<td>SW_VERSIONBUGFIX</td>
<td>xx</td>
</tr>
<tr>
<td>10</td>
<td>ERRCNT_CHECKSUM</td>
<td>xx</td>
</tr>
<tr>
<td>11</td>
<td>ERRCNT_STUFFING</td>
<td>xx</td>
</tr>
<tr>
<td>12</td>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

**SETPARAM, SETPARAM1 – Set Protocol Parameters**

This message sets the NAME of the control application (CA) according to SAE J1939/81 (Address Claim Process), sets a preferred source address and address range for the address claim process and starts the Address Claim process.

With using the SETPARAM1 command, as soon as the address claim process is finished (successful or not), the gateway will send a REPSTATUS message, reporting the address claim status and the negotiated address.

The parameters in this message are:

- NAME according to SAE Standard
- Source Address (Preferred Address)
- Address Range Bottom (optional; if not used set as NULL address = 254)
- Address Range Top (optional; if not used set as NULL address = 254)
- Operating Mode
Note:
• The Address Claim process is initiated by overwriting the Listen-Only mode (gateway default) to Event mode.

Setting the address range or not will impact the Address Claim process:

1. Full SAE J1939/81 compliant address claim with source address and alternative address range:

The user can set a preferred source address, but, in case the address is already used in the vehicle network, an alternative address range is available for the address claim process.

2. Full SAE J1939/81 compliant address claim with only one source address:

Setting both address range parameters to 254 but submitting a source address (i.e. setting the control application as non-arbitrary-address-capable) will reset the Arbitrary Address Capable flag in the NAME.

Note:
• SETPARAM, SETPARAM1 will be rejected (i.e. no ACK will be sent) in case the operating mode is set other than the above-defined values.
• The SETPARAM1 command (ID = 14) function is identical to the original SETPARAM command, but in addition it initiates the REPSTATUS message, which is being transmitted as soon as the Address Claim process is finished. This allows the host’s program to wait for the REPSTATUS message to obtain the negotiated source address.

Message Identifier: 7/14
Total Length: 17 bytes
Response: ACK

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>14</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_SET</td>
<td>7/14</td>
</tr>
<tr>
<td>4</td>
<td>NAME LSB</td>
<td>xx</td>
</tr>
<tr>
<td>5</td>
<td>NAME</td>
<td>xx</td>
</tr>
<tr>
<td>6</td>
<td>NAME</td>
<td>xx</td>
</tr>
<tr>
<td>7</td>
<td>NAME</td>
<td>xx</td>
</tr>
<tr>
<td>8</td>
<td>NAME</td>
<td>xx</td>
</tr>
<tr>
<td>9</td>
<td>NAME</td>
<td>xx</td>
</tr>
<tr>
<td>10</td>
<td>NAME</td>
<td>xx</td>
</tr>
<tr>
<td>11</td>
<td>NAME MSB</td>
<td>xx</td>
</tr>
<tr>
<td>12</td>
<td>Source Address (SA)</td>
<td>xx</td>
</tr>
<tr>
<td>13</td>
<td>SA Range Bottom</td>
<td>xx</td>
</tr>
<tr>
<td>14</td>
<td>SA Range Top</td>
<td>xx</td>
</tr>
<tr>
<td>15</td>
<td>Operating Mode</td>
<td>xx</td>
</tr>
<tr>
<td>16</td>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

REQINFO – Request Information from J1939 Gateway

This message, sent from the host system to the jCOM1939 gateway, requests the transmission of the report message REPSTATUS or the version message VERSION according to the Request ID.
**Request ID:**
REPSTATUS (ID = 9)
VERSION (ID = 13)

**Note:**
- The gateway will return no message when the Request ID is not the REPSTATUS-ID.
- Further request IDs may be added in future firmware versions.

**Message Identifier:** 8
**Total Length:** 6 bytes
**Response:** Depending on Request ID

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_RQ</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>REQUEST_ID</td>
<td>xx</td>
</tr>
<tr>
<td>5</td>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

**REPSTATUS - Report J1939 Protocol Status and Source Address**

This message reports the address claim status and the negotiated source address.

The reported STATUS will be:

- 01 – Address Claim in Progress
- 02 – Address Claim Successful – Normal Data Traffic
- 03 – Address Claim Failed
- 04 – Listen-Only Mode

The reported SA will be NULLADDRESS (254) when the address claim process is in progress, the address claim failed, or the operating mode is Listen-Only. In case the source address is 254, check the J1939 Protocol Status.

**Message Identifier:** 9
**Total Length:** 7 bytes
**Response:** -

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_RS</td>
<td>9</td>
</tr>
<tr>
<td>4</td>
<td>Status</td>
<td>xx</td>
</tr>
<tr>
<td>5</td>
<td>Source Address</td>
<td>xx</td>
</tr>
<tr>
<td>6</td>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

**FLASH – Starts the In-Circuit Programmer**
This message starts the in-circuit programmer. This is usually invoked by pulling a pin low during power-up, but with this function it can be started by the host system at any time. A flash programming tool can be used after calling this function.

After sending the FLASH command, all LEDs on the jCOM1939 will be on. In order to switch back to normal operation, you need to go through a power-off-on cycle.

**Message Identifier:** 10  
**Total Length:** 5 bytes  
**Response:** -

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_FLASH</td>
<td>10</td>
</tr>
<tr>
<td>4</td>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

**SETACK – Set Acknowledgment**

This message allows the activation/deactivation of the acknowledgment message ACK.

**Note:**
- The ACK message will be active per default, unless overwritten by the host system.

**Message Identifier:** 11  
**Total Length:** 6 bytes  
**Response:** ACK (only when activated)

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_SETACK</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>ACTIVE</td>
<td>xx (0-inactive, 1-active)</td>
</tr>
<tr>
<td>5</td>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

**SETHearth – Set Heartbeat Frequency**

This message allows the setting of the heartbeat frequency in units of milliseconds. The message frequency is assigned by two bytes, allowing a range of 100 – 5000 milliseconds, i.e. time of less than 100 milliseconds will be corrected to 100, and a time of more than 5000 will be corrected to 5000.

**Note:**
- Passing a heartbeat frequency of 0 will suppress the message, and it will not be reset to the minimum time of 100 milliseconds.

**Message Identifier:** 12  
**Total Length:** 7 bytes
Response: ACK

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_SETHEART</td>
<td>12</td>
</tr>
<tr>
<td>4</td>
<td>FREQ MSB</td>
<td>xx</td>
</tr>
<tr>
<td>5</td>
<td>FREQ LSB</td>
<td>xx</td>
</tr>
<tr>
<td>6</td>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

**VERSION – Report Hardware & Software Version**

This message is a response to the REQINFO message in which the VERSION ID was used. The actual data is identical to the HEARTBEAT message. For that reason, requesting the VERSION message should only be used after the HEARTBEAT message was suppressed.

Message Identifier: 13
Total Length: 11 bytes
Response: -

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>08</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_VERSION</td>
<td>13</td>
</tr>
<tr>
<td>4</td>
<td>HW_VERSIONMAJOR</td>
<td>xx</td>
</tr>
<tr>
<td>5</td>
<td>HW_VERSIONMINOR</td>
<td>xx</td>
</tr>
<tr>
<td>6</td>
<td>HW_VERSIONBUGFIX</td>
<td>xx</td>
</tr>
<tr>
<td>7</td>
<td>SW_VERSIONMAJOR</td>
<td>xx</td>
</tr>
<tr>
<td>8</td>
<td>SW_VERSIONMINOR</td>
<td>xx</td>
</tr>
<tr>
<td>9</td>
<td>SW_VERSIONBUGFIX</td>
<td>xx</td>
</tr>
<tr>
<td>10</td>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

**SETMSGMODE – Set Message Mode**

The message mode defines the type of data (PGNs) that is being passed from the gateway to the host system. This command is only necessary for gateway applications where the user needs to see all PGNs including those for protocol control.

The settings are:

- **00** – ECU Message Mode (default after RESET) – The gateway reports only PGNs addressed to the ECU’s source address or the global address.
- **01** – Gateway Message Mode 1 – The gateway reports PGNs addressed to all destination addresses including the global address (255).
- **02** – Gateway Message Mode 2 – The gateway reports all PGNs as described in mode 1 but also includes protocol PGNs according to SAE J1939/21 and SAE J1939/81.

Message Identifier: 15
Total Length: 6 bytes
Response: ACK

<table>
<thead>
<tr>
<th>Index</th>
<th>Code</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>MSG_TOKEN_START</td>
<td>192</td>
</tr>
<tr>
<td>1</td>
<td>Length MSB</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Length LSB</td>
<td>03</td>
</tr>
<tr>
<td>3</td>
<td>MSG_ID_MSGMODE</td>
<td>15</td>
</tr>
<tr>
<td>4</td>
<td>MSGMODE</td>
<td>xx</td>
</tr>
<tr>
<td>5</td>
<td>CHKSUM</td>
<td>xx</td>
</tr>
</tbody>
</table>

jCOM1939 Gateway – Windows Programming Interface

This chapter describes the programming interface under the Windows operating system. All code samples are based on Visual Studio 2012 using C#.

The modules (classes) used are:

- COMport.cs – Handles the COM port communication.
- jCOM1939.cs – Handles the communication protocol between the host system and the J1939 gateway.

COMport.cs

While this class contains a number of functions, the only two function calls needed for an application are:

- **COMport.Initialize** – Determines the available COM ports and initializes the selected port.
- **COMport.Terminate** – Closes the currently open COM port. This function should always be called upon termination of the Windows application.

The selection of the COM port connected to the jCOM1939 gateway is a first mandatory step to establish communication. The COM port number as assigned by the operating system may differ from computer to computer.

**COMport.Initialize**

**Description:** Initializes COM port; determines available ports

**Parameters:** int nComPort;

**Return Code:** OK / ERROR

For that reason, use nComPort = -1 to accomplish the COM port scanning. The program will fill all COM port information into an array (sPorts[]) and also recommend the appropriate port number (nSelPort). The variable nPorts contains the number of ports found. In order to initialize a COM port, the code must pass nComPort to the function, where nComPort is the index pointing to the entry in sPorts.

**Usage:**

```csharp
if (COMPort.Initialize(3) == ERROR) MessageBox.Show("Unable to initialize COMK port 3.");
```

**Note:** In the above sample the COM port number is 3, but it used as a mere example.
The following code sample demonstrates the scanning of COM ports and filling the retrieved information into a combo-box control (cboCOMPort):

```csharp
if (COMPort.Initialize(-1) != 0)
    MessageBox.Show("No COM port available!");
else
    // Fill the COM combobox
    for (int nIndex = 0; nIndex < COMPort.nPorts; nIndex++)
        cboCOMPort.Items.Add(COMPort.sPorts[nIndex]);

    // Select the combobox index
    cboCOMPort.SelectedIndex = COMPort.nSelPort;

    // Close the serial port
    COMPort._serialport.Close();

// end else
// end if
```

The following code initializes the COM port:

```csharp
if (COMPort.Initialize(cboCOMPort.SelectedIndex) != 0)
    MessageBox.Show("Sorry! There is a problem with the COM port.", "Attention!");
```

**COMPort.Terminate**

**Description:** Closes the currently opened COM port

**Parameters:** None

**Return Code:** None

The function uses the current settings to determine the open port and closes it.

**Usage:**

```csharp
COMPort.Terminate();
```

**jCOM1939.cs**

The jCOM1939 class handles the communication protocol between the host system and the J1939 gateway, i.e. all messages as described in chapter *jCOM1939 Gateway - Serial Communications Protocol* are represented by corresponding function calls. They are:

<table>
<thead>
<tr>
<th>Command Message</th>
<th>ID</th>
<th>Meaning</th>
<th>Source</th>
<th>Function Call</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACK</td>
<td>0</td>
<td>Acknowledgement</td>
<td>Gateway</td>
<td>RX_J1939</td>
</tr>
<tr>
<td>ADDFILTER</td>
<td>1</td>
<td>Add Filter</td>
<td>Host</td>
<td>FA_J1939</td>
</tr>
<tr>
<td>DELFILTER</td>
<td>2</td>
<td>Delete Filter</td>
<td>Host</td>
<td>FD_J1939</td>
</tr>
<tr>
<td>TXDATA</td>
<td>3</td>
<td>Transmit Data</td>
<td>Host</td>
<td>TX_J1939</td>
</tr>
<tr>
<td>RXDATA</td>
<td>4</td>
<td>Receive Data</td>
<td>Gateway</td>
<td>RX_J1939</td>
</tr>
<tr>
<td>RESET</td>
<td>5</td>
<td>Reset jCOM1939</td>
<td>Host</td>
<td>RESET_J1939</td>
</tr>
<tr>
<td>HEART</td>
<td>6</td>
<td>Heartbeat</td>
<td>Gateway</td>
<td>RX_J1939</td>
</tr>
<tr>
<td>SETPARAM</td>
<td>7</td>
<td>Set Protocol Parameters</td>
<td>Host</td>
<td>SET_J1939</td>
</tr>
<tr>
<td>REQINFO</td>
<td>8</td>
<td>Request Information from gateway</td>
<td>Host</td>
<td>RQ_J1939</td>
</tr>
<tr>
<td>FLASH</td>
<td>10</td>
<td>Starts the in-circuit programmer</td>
<td>Host</td>
<td>FLASH</td>
</tr>
<tr>
<td>SETACK</td>
<td>11</td>
<td>Set Acknowledgement</td>
<td>Host</td>
<td>SA_J1939</td>
</tr>
</tbody>
</table>
### Set Heartbeat Frequency

**Host:** SH_J1939

**Description:**
Checks for received messages from the J1939 gateway

**Parameters:**
- ref long lPGN, ref int nDest, ref int nSrc, ref int nPriority, ref byte[] nData, ref int nDataLen
  - lPGN = Received PGN
  - nDest = Destination Address
  - nSrc = Source Address
  - nPriority = PGN Priority
  - nData = Received data
  - nDataLen = Number of received data bytes

**Return Code:**
- RX_Message = A regular PGN was received
- RX_NoMessage = No message was received
- RX_FaultyMessage = Message was received but contained errors
- RX_HEART = Heartbeat message received
- RX_ACK_FA = Acknowledgment for Filter Add received
- RX_ACK_FD = Acknowledgment for Filter Delete received
- RX_ACK_TX = Acknowledgment for Transmit Data received
- RX_ACK_RESET = Acknowledgment for Reset received
- RX_ACK_SETPARAM = Acknowledgment for Set Protocol Parameters received
- RX_ACK_SETPARAM1 = Acknowledgment for Set Protocol Parameters w/ Response received
- RX_RS = J1939 Protocol Status received
- RX_ACK_MSGMODE = Acknowledgment for Message Mode received
- RX_ACK_LOOPBACK = Acknowledgment for Loopback received
- RX_VERSION = HW/SW Version Report received

**Usage:**
```c
long lPGN = 0;
int nDest = 0;
int nSrc = 0;
int nPriority = 0;
byte[] nData = new byte[8]; // Will be resized in the RX_J1939 function call
int nDataLen = 0;

int nReceiveMode = jCOM1939.RX_J1939(ref lPGN, ref nDest, ref nSrc, ref nPriority, ref nData, ref nDataLen);
```

### Set Protocol Parameters with REPSTATUS response

**Host:** SET_J1939

**Description:**
Checks for received messages from the J1939 gateway

**Parameters:**
- ref long lPGN, ref int nDest, ref int nSrc, ref int nPriority, ref byte[] nData, ref int nDataLen
  - lPGN = Received PGN
  - nDest = Destination Address
  - nSrc = Source Address
  - nPriority = PGN Priority
  - nData = Received data
  - nDataLen = Number of received data bytes

**Return Code:**
- RX_Message = A regular PGN was received
- RX_NoMessage = No message was received
- RX_FaultyMessage = Message was received but contained errors
- RX_HEART = Heartbeat message received
- RX_ACK_FA = Acknowledgment for Filter Add received
- RX_ACK_FD = Acknowledgment for Filter Delete received
- RX_ACK_TX = Acknowledgment for Transmit Data received
- RX_ACK_RESET = Acknowledgment for Reset received
- RX_ACK_SETPARAM = Acknowledgment for Set Protocol Parameters received
- RX_ACK_SETPARAM1 = Acknowledgment for Set Protocol Parameters w/ Response received
- RX_RS = J1939 Protocol Status received
- RX_ACK_MSGMODE = Acknowledgment for Message Mode received
- RX_ACK_LOOPBACK = Acknowledgment for Loopback received
- RX_VERSION = HW/SW Version Report received

**Usage:**
```c
long lPGN = 0;
int nDest = 0;
int nSrc = 0;
int nPriority = 0;
byte[] nData = new byte[8]; // Will be resized in the RX_J1939 function call
int nDataLen = 0;

int nReceiveMode = jCOM1939.RX_J1939(ref lPGN, ref nDest, ref nSrc, ref nPriority, ref nData, ref nDataLen);
```

### Set Message Mode

**Host:** MM_J1939

**Description:**
Checks for received messages from the J1939 gateway

**Parameters:**
- ref long lPGN, ref int nDest, ref int nSrc, ref int nPriority, ref byte[] nData, ref int nDataLen
  - lPGN = Received PGN
  - nDest = Destination Address
  - nSrc = Source Address
  - nPriority = PGN Priority
  - nData = Received data
  - nDataLen = Number of received data bytes

**Return Code:**
- RX_Message = A regular PGN was received
- RX_NoMessage = No message was received
- RX_FaultyMessage = Message was received but contained errors
- RX_HEART = Heartbeat message received
- RX_ACK_FA = Acknowledgment for Filter Add received
- RX_ACK_FD = Acknowledgment for Filter Delete received
- RX_ACK_TX = Acknowledgment for Transmit Data received
- RX_ACK_RESET = Acknowledgment for Reset received
- RX_ACK_SETPARAM = Acknowledgment for Set Protocol Parameters received
- RX_ACK_SETPARAM1 = Acknowledgment for Set Protocol Parameters w/ Response received
- RX_RS = J1939 Protocol Status received
- RX_ACK_MSGMODE = Acknowledgment for Message Mode received
- RX_ACK_LOOPBACK = Acknowledgment for Loopback received
- RX_VERSION = HW/SW Version Report received

**Usage:**
```c
long lPGN = 0;
int nDest = 0;
int nSrc = 0;
int nPriority = 0;
byte[] nData = new byte[8]; // Will be resized in the RX_J1939 function call
int nDataLen = 0;

int nReceiveMode = jCOM1939.RX_J1939(ref lPGN, ref nDest, ref nSrc, ref nPriority, ref nData, ref nDataLen);
```

### Message Loopback

**Host:** LOOPBACK_J1939

**Description:**
Checks for received messages from the J1939 gateway

**Parameters:**
- ref long lPGN, ref int nDest, ref int nSrc, ref int nPriority, ref byte[] nData, ref int nDataLen
  - lPGN = Received PGN
  - nDest = Destination Address
  - nSrc = Source Address
  - nPriority = PGN Priority
  - nData = Received data
  - nDataLen = Number of received data bytes

**Return Code:**
- RX_Message = A regular PGN was received
- RX_NoMessage = No message was received
- RX_FaultyMessage = Message was received but contained errors
- RX_HEART = Heartbeat message received
- RX_ACK_FA = Acknowledgment for Filter Add received
- RX_ACK_FD = Acknowledgment for Filter Delete received
- RX_ACK_TX = Acknowledgment for Transmit Data received
- RX_ACK_RESET = Acknowledgment for Reset received
- RX_ACK_SETPARAM = Acknowledgment for Set Protocol Parameters received
- RX_ACK_SETPARAM1 = Acknowledgment for Set Protocol Parameters w/ Response received
- RX_RS = J1939 Protocol Status received
- RX_ACK_MSGMODE = Acknowledgment for Message Mode received
- RX_ACK_LOOPBACK = Acknowledgment for Loopback received
- RX_VERSION = HW/SW Version Report received

**Usage:**
```c
long lPGN = 0;
int nDest = 0;
int nSrc = 0;
int nPriority = 0;
byte[] nData = new byte[8]; // Will be resized in the RX_J1939 function call
int nDataLen = 0;

int nReceiveMode = jCOM1939.RX_J1939(ref lPGN, ref nDest, ref nSrc, ref nPriority, ref nData, ref nDataLen);
```
Parameters: long lPGN
Return Code: None

Usage:

```c
long lPGN = 65280;  // Just an example PGN
jCOM1939.FD_J1939(lPGN);
```

jCOM1939_TX_J1939

Description: Transmits a PGN and its data, allowing a message loopback
Parameters:
- long lPGN, int nDest, int nSrc, int nPriority, byte[] nData, int nDataLen
- lPGN = PGN
- nDest = Destination Address
- nSrc = Source Address
- nPriority = PGN Priority
- nData = Data
- nDataLen = Number of data bytes
- bLoopback = true/false

Return Code: None

Usage:

```c
long lPGN = 65280;
int nDest = 0;
int nSrc = 0;
int nPriority = 0;
byte[] nData = new byte[8]; // Fill this one with data
int nDataLen = 0;

jCOM1939_TX_J1939(lPGN, nDest, nSrc, nPriority, nData, nDataLen, true);
```

jCOM1939_RESET_J1939

Description: Resets the J1939 gateway
Parameters: None
Return Code: None

Usage:

```c
jCOM1939_RESET_J1939();
```

jCOM1939_SET_J1939

Description: Sets J1939 protocol parameters
Parameters:
- byte nSrcAddr, byte nAddrBottom, byte nAddrTop, byte nOpMode, byte[] pNAME,
- bool bRequestRESTATUSMessage
- nSrcAddr = Preferred node address
- nAddrBottom = Bottom of alternative node address range
- nAddrTop = Top of alternative node address range
- nOpMode = Operating Mode
- pNAME = J1939 NAME
- bRequestRESTATUSMessage = true if status message is required after address claim process

Return Code: None

Usage:

```c
byte nSrcAddr = 128;  // Just examples...
byte nAddrBottom = 129;
byte nAddrTop = 247;
byte[] pNAME = new byte[8]; // Enter the NAME here

jCOM1939_SET_J1939(nSrcAddr, nAddrBottom, nAddrTop, jCOM1939.OPMODE_Event, pNAME, true);
```
### jCOM1939.RQ_J1939

**Description:** Resets the J1939 gateway  
**Parameters:** byte nID  
**Return Code:** None  

**Usage:**
```c
jCOM1939.RQ_J1939(jCOM1939.MSG_ID_RS); // Request status report
```

### jCOM1939.FLASH

**Description:** Invoke the in-circuit programmer  
**Parameters:** None  
**Return Code:** None  

**Usage:**
```c
jCOM1939.FLASH();
```

### jCOM1939.SA_J1939

**Description:** Activates/Deactivates ACK message  
**Parameters:** byte nActive (0 = inactive; 1 = active)  
**Return Code:** None  

**Usage:**
```c
jCOM1939.SA_J1939(0);
```

### jCOM1939.SH_J1939

**Description:** Set heartbeat frequency in milliseconds  
**Parameters:** byte nFrequency  
**Return Code:** None  

**Usage:**
```c
jCOM1939.SA_J1939(1000);
```

### jCOM1939.MM_J1939

**Description:** Sets the gateway message mode  
**Parameters:** byte nMessageMode  
**Return Code:** None  

**Usage:**
```c
jCOM1939.MM_J1939(jCOM1939.MSGMODE_ECU);
```