

# AN701 – Axiomatic Proprietary SSP Message Format

## Message Transmission

Communication between smart devices, such as phones or tablets, and the CAN2BT device is based on a Bluetooth Serial Port Profile (SPP). By default, the CAN2BT firmware declares itself as a SPP device (UUID: 00001101-0000-1000-8000-00805F9B34FB)

The messages are transferred in binary format, least significant byte first. The list of supported proprietary messages is shown below. Both versions of the CAN to Bluetooth device, AX141100 and AX141150, support this message format. However, not all messages described below will work in all devices, the biggest differences being in the Flash data handling and RTC commands.

Most of this material is also available in the user manuals for the AX141100 and AX141150 devices.

## Overall message format

| Byte 0     | Byte 1   | Byte 2      | Byte 3      | ... | Byte n+2           | Byte n+3 | Byte n+4 | Byte n+5 | Byte n+6 |
|------------|----------|-------------|-------------|-----|--------------------|----------|----------|----------|----------|
| <msg type> | <length> | <payload 0> | <payload 1> | ... | <payload length-1> | <crc32>  | <crc32>  | <crc32>  | <crc32>  |

In the description above, the *<msg type>* is as listed in the table below. *<length>* is the full message length without the four CRC32 bytes. CRC32 was selected because support for it is readily available on Android devices (the Axiomatic CAN2Bluetooth Configuration app uses **Android's built-in CRC32 function**).

All data that is expressed as Byte 0, Byte 1, ... in the message descriptions below is expected to be either 16 bits or 32 bits wide data, broken down to bytes (8 bits), least significant byte first. The only exception is the PIN code data; that data should be formatted one digit per byte. The PIN codes are hard formatted to have 6 digits.

There is an ack response sent by the CAN2BT device after receiving the configuration messages. The ack response contains status info: **0x00 – OK, 0x01 – Failed**.

Available CAN baud rate options include: 1 – 10k, 2 – 20k, 3 – 50k, 4 – 100k, 5 – 125k, 6 – 250k, 7 – 500k, 9 – 1M. In case the CAN baud rate is configured as 0 (or 8), it is considered unconfigured and will default to 250k.

## Message types

| Message type                         | Direction     | <msg type> byte |
|--------------------------------------|---------------|-----------------|
| New connection notification          | CAN2BT -> SD* | 0x00            |
| CAN data with StdID                  | SD* -> CAN2BT | 0x12            |
| CAN data with ExtID                  | SD* -> CAN2BT | 0x13            |
| CAN data with StdID                  | CAN2BT -> SD* | 0x21            |
| CAN data with ExtID                  | CAN2BT -> SD* | 0x31            |
| MAP write                            | SD* -> CAN2BT | 0x34            |
| MAP write response                   | CAN2BT -> SD* | 0x43            |
| MAP read                             | SD* -> CAN2BT | 0x45            |
| MAP read response                    | CAN2BT -> SD* | 0x54            |
| Download Flash contents              | SD* -> CAN2BT | 0x56            |
| Download Flash contents response     | CAN2BT -> SD* | 0x65            |
| Download Flash contents ACK          | SD* -> CAN2BT | 0x57            |
| Download Flash contents ACK response | CAN2BT -> SD* | 0x75            |
| Change configuration                 | SD* -> CAN2BT | 0x67            |
| Change configuration response        | CAN2BT -> SD* | 0x76            |
| LED state                            | SD* -> CAN2BT | 0x78            |
| LED state response                   | CAN2BT -> SD* | 0x87            |
| Multiple CAN Rx frames**             | CAN2BT -> SD* | 0xFE            |
| Multiple CAN Tx frames**             | SD* -> CAN2BT | 0xEF            |

\* SD = Smart Device (e.g., a phone or a tablet)

\*\* Supported starting from AX141150 firmware v3.00

## New connection notification

| Connected to: <configured node name> (default: CAN-BLE) |       |       |       |       |       |       |
|---------------------------------------------------------|-------|-------|-------|-------|-------|-------|
| <'C'>                                                   | <'A'> | <'N'> | <'-'> | <'B'> | <'L'> | <'E'> |

## CAN messages, single messages

| CAN frame format, StdID (ID bit 11 = RTR flag, bits 10 to 0, StdID) |        |       |      |      |     |           |
|---------------------------------------------------------------------|--------|-------|------|------|-----|-----------|
| <ID 0>                                                              | <ID 1> | <len> | <D0> | <D1> | ... | <D len-1> |

| CAN frame format, ExtID (ID bit 31 = RTR flag, bits 28 to 0, ExtID) |        |        |        |       |      |      |     |           |
|---------------------------------------------------------------------|--------|--------|--------|-------|------|------|-----|-----------|
| <ID 0>                                                              | <ID 1> | <ID 2> | <ID 3> | <len> | <D0> | <D1> | ... | <D len-1> |

## CAN messages, multiple frames combined

| Multiple CAN frames, Rx and Tx, ExtID and StdID (n = max 6 frames, individual frames described below) |                |     |                |
|-------------------------------------------------------------------------------------------------------|----------------|-----|----------------|
| <CAN frame #1>                                                                                        | <CAN frame #2> | ... | <CAN frame #n> |

| CAN frame format inside a multi-frame message, StdID |          |                          |       |      |      |     |           |
|------------------------------------------------------|----------|--------------------------|-------|------|------|-----|-----------|
| 0x21                                                 | <ID LSB> | <ID 3MSBs   bit 8 = RTR> | <len> | <D0> | <D1> | ... | <D len-1> |

| CAN frame format inside a multi-frame message, ExtID |          |        |        |                          |       |      |      |     |           |
|------------------------------------------------------|----------|--------|--------|--------------------------|-------|------|------|-----|-----------|
| 0x31                                                 | <ID LSB> | <ID 1> | <ID 2> | <ID 5MSBs   bit 8 = RTR> | <len> | <D0> | <D1> | ... | <D len-1> |

## MAP access

| J1939 MAP access (data types: 1=u8, 2=u16, 4=u32, 5=f32) |         |         |         |             |         |         |         |         |
|----------------------------------------------------------|---------|---------|---------|-------------|---------|---------|---------|---------|
| <Remote node addr>                                       | <SP A0> | <SP A1> | <SP A2> | <data type> | <SP D0> | <SP D1> | <SP D2> | <SP D3> |

## Configuration messages

| Configuration mode (d0 ... d5 are single digits of the PIN code set using command 0x21) |      |      |      |      |      |      |      |
|-----------------------------------------------------------------------------------------|------|------|------|------|------|------|------|
| Enter config mode                                                                       | 0xC0 | <d0> | <d1> | <d2> | <d3> | <d4> | <d5> |
| Exit config mode                                                                        | 0xC1 |      |      |      |      |      |      |

| CAN Rx Filters          |      |             |             |             |             |           |           |           |           |  |
|-------------------------|------|-------------|-------------|-------------|-------------|-----------|-----------|-----------|-----------|--|
| Add CAN Filter          | 0x01 | <filter b0> | <filter b1> | <filter b2> | <filter b3> | <mask b0> | <mask b1> | <mask b2> | <mask b3> |  |
| Remove CAN Filter       | 0x02 | <filter b0> | <filter b1> | <filter b2> | <filter b3> |           |           |           |           |  |
| Enter promisc. mode     | 0x03 |             |             |             |             |           |           |           |           |  |
| Exit promisc. mode      | 0x04 |             |             |             |             |           |           |           |           |  |
| Get current CAN Filters | 0x09 |             |             |             |             |           |           |           |           |  |

| J1939 PGN special handling (CAN->BT, tx interval in ms, 0ms == tx on reception, flags: 0 – default, 1 – set BTID) |             |              |          |          |              |                      |                      |                |                |
|-------------------------------------------------------------------------------------------------------------------|-------------|--------------|----------|----------|--------------|----------------------|----------------------|----------------|----------------|
| Add PGN to forward list                                                                                           | <b>0x07</b> | <PGN lsb b0> | <PGN b1> | <PGN b2> | <PGN msb b3> | <tx interval lsb b0> | <tx interval msb b1> | <flags lsb b0> | <flags msb b1> |
| Remove PGN from forward list                                                                                      | <b>0x08</b> | <PGN lsb b0> | <PGN b1> | <PGN b2> | <PGN msb b3> |                      |                      |                |                |
| Get PGNs to forward                                                                                               | <b>0x0A</b> |              |          |          |              |                      |                      |                |                |

| CAN bus configuration (br index = 1-10k, 2-20k, 3-50k, 4-100k, 5-125k, 6-250k, 7-500k, 9-1M, state = 0-off, 1-on) |             |            |
|-------------------------------------------------------------------------------------------------------------------|-------------|------------|
| CAN bus baud rate                                                                                                 | <b>0x05</b> | <br index> |
| Set BT CAN TX echo                                                                                                | <b>0x06</b> | <state>    |
| Get CAN bus baud rate                                                                                             | <b>0x0B</b> |            |

| Connection functions          |             |                                              |
|-------------------------------|-------------|----------------------------------------------|
| Scan available devices        | <b>0x10</b> |                                              |
| Connect to remote device      | <b>0x11</b> | <scan index>                                 |
| Disconnect from remote device | <b>0x12</b> |                                              |
| Set autoconnect               | <b>0x13</b> | <scan index>                                 |
| Define accepted BD ADDR       | <b>0x14</b> | <scan index, 0 = currently connected device> |
| Disable autoconnect           | <b>0x15</b> |                                              |

## AN701

Version 1.04

| PIN Codes (o0...o5 are single digits of the <i>OLD</i> PIN code and n0...n5 are single digits of the <i>NEW</i> PIN) |             |      |      |      |      |      |      |      |      |      |      |      |      |
|----------------------------------------------------------------------------------------------------------------------|-------------|------|------|------|------|------|------|------|------|------|------|------|------|
| Set Pairing PIN Code                                                                                                 | <b>0x20</b> | <o0> | <o1> | <o2> | <o3> | <o4> | <o5> | <n0> | <n1> | <n2> | <n3> | <n4> | <n5> |
| Set Config PIN Code                                                                                                  | <b>0x21</b> | <o0> | <o1> | <o2> | <o3> | <o4> | <o5> | <n0> | <n1> | <n2> | <n3> | <n4> | <n5> |
| Set Rem.Acc. PIN Code                                                                                                | <b>0x22</b> | <o0> | <o1> | <o2> | <o3> | <o4> | <o5> | <n0> | <n1> | <n2> | <n3> | <n4> | <n5> |

| Data logging (when defining ID, bit 31 = ExtID, bit 30 = RTR, bits 29-0 define the ID) |             |               |         |         |         |
|----------------------------------------------------------------------------------------|-------------|---------------|---------|---------|---------|
| Define PGN to log                                                                      | <b>0x30</b> | <PGN 0>       | <PGN 1> | <PGN 2> | <PGN 3> |
| Define SrcAddress to log                                                               | <b>0x31</b> | <Src Address> |         |         |         |
| Define ID to log                                                                       | <b>0x32</b> | <ID 0>        | <ID 1>  | <ID 2>  | <ID 3>  |
| List current rules                                                                     | <b>0x33</b> |               |         |         |         |
| Erase all rules                                                                        | <b>0x34</b> |               |         |         |         |

| Flash functions             |             |                |                |              |              |
|-----------------------------|-------------|----------------|----------------|--------------|--------------|
| Erase all flash data        | <b>0x40</b> |                |                |              |              |
| Erase specific flash blocks | <b>0x41</b> | <start blk b0> | <start blk b1> | <end blk b0> | <end blk b1> |
| Set flash address           | <b>0x42</b> | <block b0>     | <block b1>     | <page b0>    | <page b1>    |

| RTC (hour=0...23, min&sec=0...59, day=1...31, weekday=1(mon)...7(sun), month=1...12, year=16... (16=2016)) |             |        |       |       |       |           |         |        |
|------------------------------------------------------------------------------------------------------------|-------------|--------|-------|-------|-------|-----------|---------|--------|
| Get time                                                                                                   | <b>0x50</b> |        |       |       |       |           |         |        |
| Set time                                                                                                   | <b>0x51</b> | <hour> | <min> | <sec> | <day> | <weekday> | <month> | <year> |

| LEDs (mode: 0 – default, 1 – use commands, state: 0 – off, 1 - on) |             |        |  |  |  |  |  |  |
|--------------------------------------------------------------------|-------------|--------|--|--|--|--|--|--|
| Set LED mode                                                       | <b>0x52</b> | <mode> |  |  |  |  |  |  |

| Misc. functions  |             |         |       |         |       |       |       |       |       |
|------------------|-------------|---------|-------|---------|-------|-------|-------|-------|-------|
| SW reset         | <b>0xF0</b> | <'r'>   | <'e'> | <'s'>   | <'e'> | <'t'> |       |       |       |
| Start bootloader | <b>0xF1</b> | <'b'>   | <'l'> | <'o'>   | <'a'> | <'d'> | <'e'> | <'r'> |       |
| Default settings | <b>0xF2</b> | <'d'>   | <'e'> | <'f'>   | <'a'> | <'u'> | <'l'> | <'t'> | <'s'> |
| Bluetooth ID     | <b>0x60</b> | <chr 1> | ...   | <chr n> |       |       |       |       |       |

### Download flash contents

| Download flash contents (if start blk=0, end blk=65535, downloading all data) |               |             |             |              |              |            |            |
|-------------------------------------------------------------------------------|---------------|-------------|-------------|--------------|--------------|------------|------------|
| <start blk 0>                                                                 | <start blk 1> | <end blk 0> | <end blk 1> | <start pg 0> | <start pg 1> | <end pg 0> | <end pg 1> |

| Download flash contents response |          |          |     |          |  |  |  |
|----------------------------------|----------|----------|-----|----------|--|--|--|
| <chunk no#>                      | <data 0> | <data 1> | ... | <data n> |  |  |  |

| Download flash contents ACK |  |  |  |  |  |  |  |
|-----------------------------|--|--|--|--|--|--|--|
| <chunk no#>                 |  |  |  |  |  |  |  |

| Download flash contents ACK response (status 0:OK, 1:no more data available) |  |  |  |  |  |  |  |
|------------------------------------------------------------------------------|--|--|--|--|--|--|--|
| <status>                                                                     |  |  |  |  |  |  |  |

### LED commands

| LED state (LED #0: Green, LED #1: Red. LED state: 0=off, 1=on) |             |  |  |  |  |  |  |
|----------------------------------------------------------------|-------------|--|--|--|--|--|--|
| <LED no#>                                                      | <LED state> |  |  |  |  |  |  |

| LED state response (status 0:OK, 1:failed) |  |  |  |  |  |  |  |
|--------------------------------------------|--|--|--|--|--|--|--|
| <status>                                   |  |  |  |  |  |  |  |

### Poll PGN message

| Poll J1939 PGN (Destination Address and Source Address bytes are optional) |          |          |          |      |      |
|----------------------------------------------------------------------------|----------|----------|----------|------|------|
| <PGN b0>                                                                   | <PGN b1> | <PGN b2> | <PGN b3> | <DA> | <SA> |

## Example message breakdown

The *Enter Configuration Mode* command is used as an example:

**0x67 0x09 0xc0 0x30 0x30 0x30 0x30 0x30 0x30 0x26 0xd8 0xdd 0x85**

The message consists of the following bytes:

**0x67** = change configuration

**0x09** = message length (first byte included = 0x67, last byte included = 0x30)

**0xc0** = enter configuration mode

**0x30 0x30 0x30 0x30 0x30 0x30** = pin code (000000)

**0x26 0xd8 0xdd 0x85** = CRC32 checksum

## Other example messages

These example messages are built using the message format described above and can be sent as is to an AX14100 (or AX141150) device.

Scan remote nodes: **0x67 0x03 0x10 0x10 0x47 0x43 0x84**

Use bridge mode: **0x67 0x03 0x03 0xce 0x06 0xfd**

Get time: **0x67 0x03 0x50 0x80 0x06 0x9f 0xf2**

Start bootloader: **0x67 0x0a 0xf1 0x62 0x6c 0x6f 0x61 0x64 0x65 0x72 0x2c 0xe7 0xe6 0x64**

Send Ext.ID CAN Frame (*ID=0x18FF0002, len=8, data: 0x11 0xff 0x34 0x22 0x56 0xee 0xab 0x09*):

**0x13 0x0f 0x02 0x00 0xff 0x18 0x08 0x11 0xff 0x34 0x22 0x56 0xee 0xab 0x09 0xca 0x35 0x66 0x5e**

Send Std.ID CAN Frame (*ID=0x1F0, len=8, data: 0x11 0xff 0x34 0x22 0x56 0xee 0xab 0x09*):

**0x12 0x0d 0xf0 0x01 0x08 0x11 0xff 0x34 0x22 0x56 0xee 0xab 0x09 0xbf 0xc8 0xa1 0xe7**

Use commands for LED driving: **0x67 0x04 0x52 0x01 0x3a 0xb7 0x36 0x7f**

Turn on Red LED: **0x78 0x04 0x01 0x01 0x64 0xfe 0xda 0xe6**

Exit configuration mode: **0x67 0x03 0xc1 0x52 0xa5 0x97 0x75**

## Example messages, from AX141100 to another Bluetooth device

These example messages are generated by an AX141100 device.

New connection notification (connected to CAN-BLE):

**0x00 0x09 0x43 0x41 0x4e 0x2d 0x42 0x4c 0x45 0xdb 0xc4 0x5e 0xd4**

Get time (12:18:02, 16.Aug.2016):

**0x76 0x0a 0x50 0x0c 0x12 0x02 0x10 0x02 0x08 0x10 0xe8 0x67 0x58 0xad**

Received 29bit ID frame, no special CAN RX filter config needed for PDU2 (ID: 0x18ff01f8, len: 8, data 0x 11 22 34 aa bb cc 01 02)

**0x31 0x0f 0xf8 0x01 0xff 0x18 0x08 0x11 0x22 0x34 0xaa 0xbb 0xcc 0x01 0x02 0x49 0xd6 0x25 0x52**

Received 11bit ID data. Note: this needs bridge mode or a special CAN RX filter configuration to work! (ID: 0x187, len: 8, data 0x 11 22 34 aa bb cc 01 02)

**0x21 0x0d 0x87 0x01 0x08 0x11 0x22 0x34 0xaa 0xbb 0xcc 0x01 0x02 0x78 0x11 0x6f 0x1d**

## Message flow examples

Figure 1 shows the message flow when a PC or a smart device commands the AX141100 to enter configuration mode and then exit configuration mode. The PIN code in this example is the default pin ('000000') and is the correct PIN for entering configuration mode.

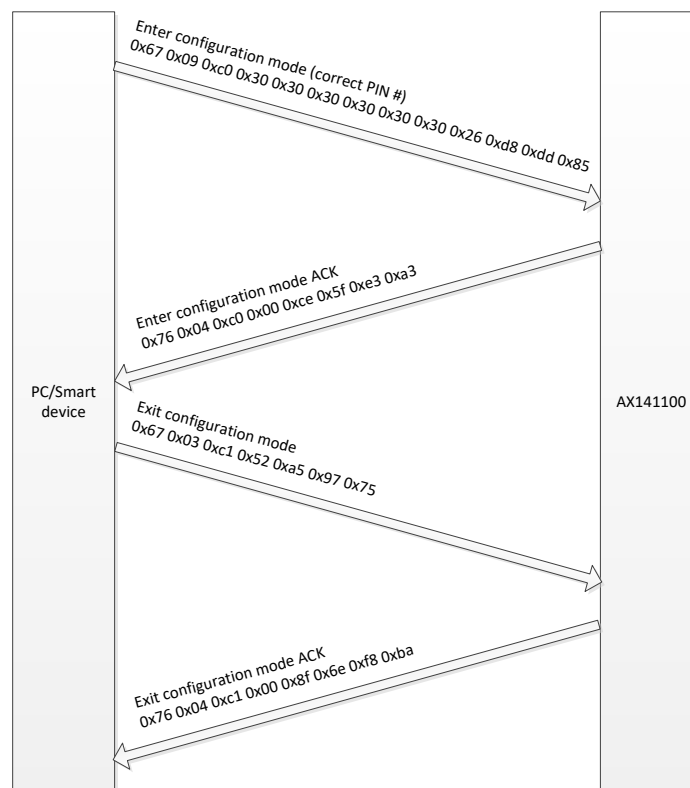


Figure 1 - Enter configuration mode / exit configuration mode



Figure 2 shows the message flow when trying to enter configuration mode with a wrong PIN code. In this case, the AX141100 device sends a NACK.

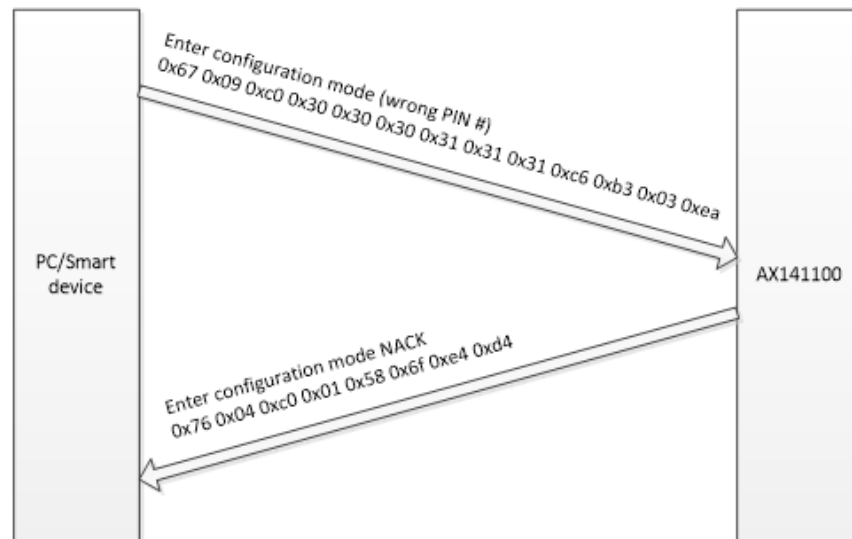


Figure 2 - Entering configuration mode with a wrong PIN code

Figure 3 describes the procedure for configuring the AX141100 device to work as a bridge. First the PC or smart device configures the AX141100 to enter configuration mode (which is required for setting the bridge mode active). Then the AX141100 is reset, to activate the new mode of operation. The reset is needed mostly for reconfiguring the CAN receive filters.

After the reset, the AX141100 sends a new connection ack message and then starts forwarding all received CAN messages. Note that in bridge mode all messages are forwarded, including frames with both 11-bit and 29-bit identifiers.

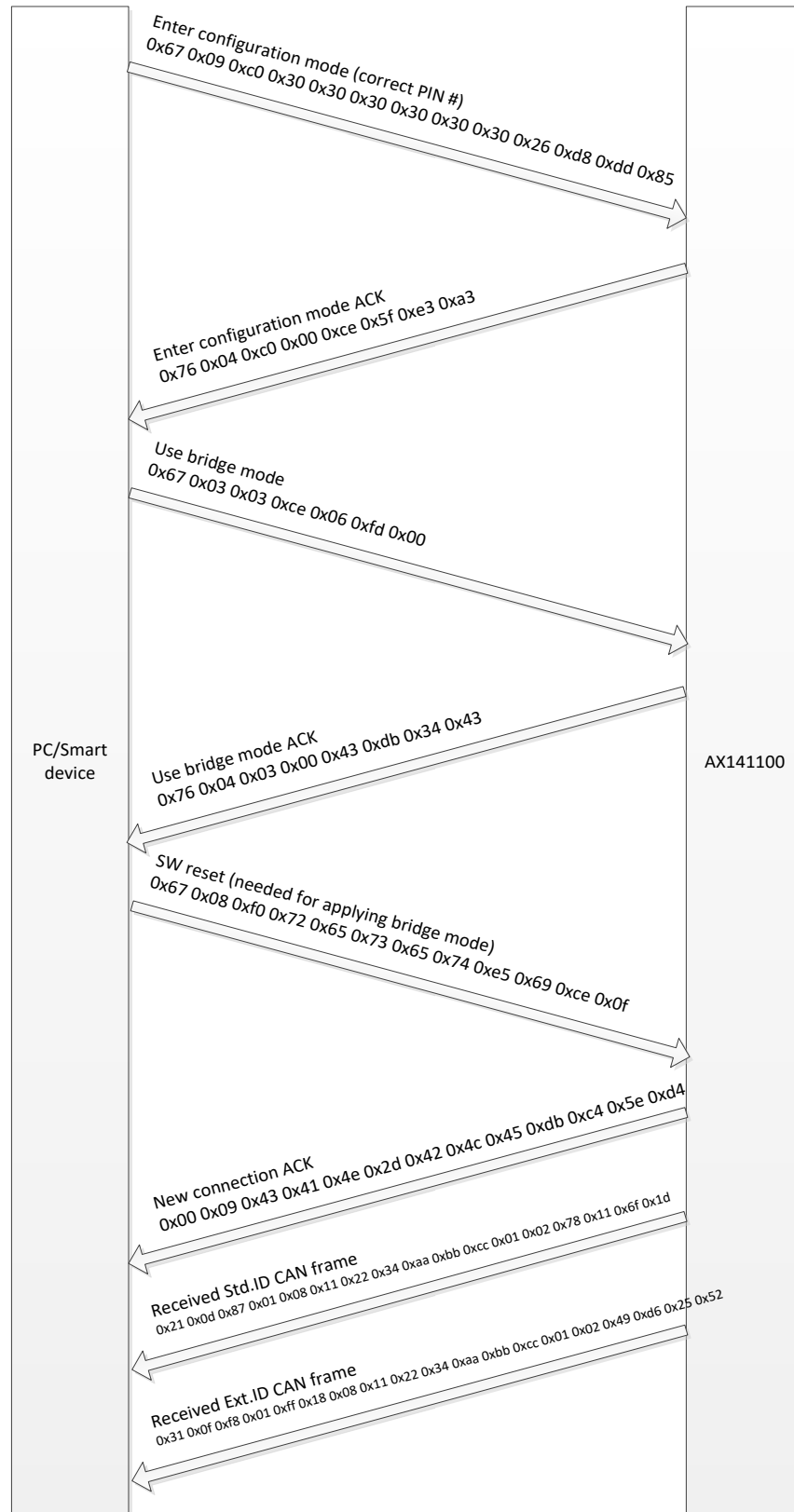


Figure 3 - Setting device to bridge mode for receiving all CAN messages.

Example java class for accessing the AX141100 using SPP:

```
1
2  import java.util.zip.CRC32;
3
4  public class bt_example extends MainActivity {
5
6      private static String selectedAddress;
7      // Well known SPP UUID
8      private static final UUID SPP_UUID =
9          UUID.fromString("00001101-0000-1000-8000-00805F9B34FB");
10
11      BluetoothAdapter bluetoothAdapter;
12      BluetoothDevice mBTDev;
13      BluetoothSocket btSocketCAN;
14      OutputStream outputStream;
15      InputStream inputStream;
16
17      public void initBTConnection(BluetoothAdapter mBluetoothAdapter) {
18
19          // Resetting selectedAddress
20          selectedAddress = EMPTY_BD_ADDR;
21          if (connectionBDADDR != null) connectionBDADDR.setText(selectedAddress);
22
23          bluetoothAdapter = mBluetoothAdapter;
24
25          Log.i(TAG, "BT connection initialized.");
26      }
27
28      void setSelectedAddress(String newAddress) {
29          selectedAddress = newAddress;
30      }
31
32      int startBTConnection(MainActivity activity) {
33          mBTDev = bluetoothAdapter.getRemoteDevice(selectedAddress);
34          Log.i(TAG, "Connecting to " + selectedAddress + "...");
35
36          try {
37              btSocketCAN = mBTDev.createRfcommSocketToServiceRecord(SPP_UUID);
38          } catch (IOException e) {
39              Log.e(TAG, "Error in startBTConnection() and socket create failed: " +
40                  e.getMessage() + ".");
41          }
42
43          // Establish the connection. This will block until it connects.
44          try {
45              btSocketCAN.connect();
46          } catch (IOException e) {
47              Log.e(TAG, "Something went wrong while connecting - Connection CLOSED!");
48              try {
49                  btSocketCAN.close();
50              } catch (IOException e2) {
51                  Log.e(TAG, "Unable to close locally...");
52              }
53              return -1;
54          }
55          Log.i(TAG, "Connection established and data link opened...");
56
57          // Create a data stream so we can talk to server.
58          try {
59              outputStream = btSocketCAN.getOutputStream();
60          } catch (IOException e) {
61              Log.e(TAG, "Error in startBTConnection() and CAN output stream creation
62                  failed:" + e.getMessage() + ".");
63              return -2;
64          }
65          Log.i(TAG, "Output stream opened...");
66
67          try {
68              inputStream = btSocketCAN.getInputStream();
69          } catch (IOException e) {
```

```
68         Log.e(TAG, "Error in startBTConnection() and CAN input stream creation
69         failed:" + e.getMessage() + ".");
70         return -3;
71     }
72     Log.i(TAG, "Input stream opened...");
73     readDataFromRemoteDevice readHandler = new readDataFromRemoteDevice();
74     readHandler.setHostMainActivity(activity);
75     readHandler.execute();
76     return 0; // OK
77 }
78
79 private class readDataFromRemoteDevice extends AsyncTask<Void, byte[], byte[]> {
80     MainActivity hostMainActivity;
81     long lCANFrameCount;
82     void setHostMainActivity( MainActivity activity ) { hostMainActivity =
83     activity; lCANFrameCount = 0; }
84
85     /**
86     * The system calls this to perform work in a worker thread and
87     * delivers it the parameters given to AsyncTask.execute()
88     */
89     protected byte[] doInBackground(Void... unused) {
90         DataInputStream mmInStream = new DataInputStream(inStream);
91         byte[] buffer = new byte[128]; // buffer store for the stream
92         while (true) {
93             try {
94                 if( mmInStream.read(buffer) > 0 ) {
95                     publishProgress(buffer);
96                 }
97             } catch (IOException e) {
98                 Log.v(TAG, "In onResume() and an exception occurred during read: "
99                 + e.getMessage());
100                 break;
101             }
102         }
103         return buffer;
104     }
105
106     protected void onProgressUpdate(byte[]... readMsgs) {
107         if (readMsgs == null || readMsgs.length == 0)
108             return;
109
110         CRC32 crc = new CRC32();
111         int msgLength, msgCRC;
112         byte[] rxMessage;
113
114         // Parsing the received messages
115         msgLength = readMsgs[0][1];
116         if ((msgLength <= 0) || (msgLength > 104)) {
117             // Illegal message length!
118         }
119         else {
120             // Length OK, making a local copy. +4 is for the CRC
121             rxMessage = Arrays.copyOfRange(readMsgs[0], 0, (msgLength + 4));
122
123             byte[] rawCRC = {rxMessage[msgLength + 3], rxMessage[msgLength + 2],
124             rxMessage[msgLength + 1], rxMessage[msgLength]};
125
126             ByteBuffer wrapped = ByteBuffer.wrap(rawCRC);
127             msgCRC = wrapped.getInt();
128
129             crc.update(rxMessage, 0, msgLength);
130
131         }
132     }
```

```

133
134
135         if (msgCRC == (int) crc.getValue()) {
136             // -> CRC OK
137             /* Handle responses... */
138             if (rxMessage[0] == (byte) 0x21) {
139                 // Std.ID CAN data
140                 // ...
141             } else if (rxMessage[0] == (byte) 0x31) {
142                 // Ext.ID CAN data
143                 // ...
144             } else if (rxMessage[0] == (byte) 0x43) {
145                 // MAP write response
146                 // ...
147             } else if (rxMessage[0] == (byte) 0x54) {
148                 // MAP read response
149                 // ...
150             } else if (rxMessage[0] == (byte) 0x65) {
151                 // Download flash contents response
152                 // ...
153             } else if (rxMessage[0] == (byte) 0x75) {
154                 // Download flash contents ack response
155                 // ...
156             } else if (rxMessage[0] == (byte) 0x76) {
157                 // Change configuration response
158                 // ...
159             } else if (rxMessage[0] == (byte) 0x87) {
160                 // LED state response
161                 // ...
162             }
163         }
164     }
165
166     /**
167     * The system calls this to perform work in the UI thread and delivers
168     * the result from doInBackground()
169     */
170     protected void onPostExecute(byte[] readMsg) {
171         Log.v(TAG, "...Data reading ended!");
172     }
173 }
174 }

```

## AN701

Version 1.04

| Version | Date               | Authors                        | Comments                          |
|---------|--------------------|--------------------------------|-----------------------------------|
| 1.00    | August 20, 2019    | Gustavo Del Valle / Sue Thomas |                                   |
| 1.01    | August 26, 2019    | Antti Keranen / Sue Thomas     |                                   |
| 1.02    | May 19, 2021       | Antti Keranen / Sue Thomas     | Figures 1,2,3 updated             |
| 1.03    | September 26, 2022 | Antti Keränen                  | New SPP messages added.           |
| 1.04    | July 6, 2023       | Kiril Mojssov                  | Legacy Updates & Marketing Review |
|         |                    |                                |                                   |