CAN bridge / smart repeater

0. What is it?

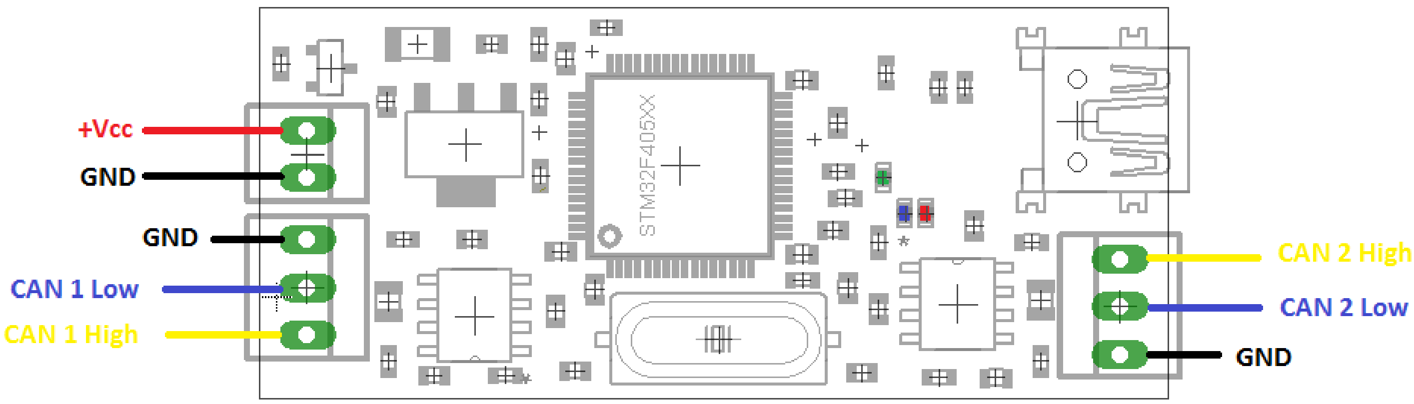
This is a bridge with two CAN ports: CAN1 and CAN2. The CAN bus packet received by CAN1 port will be immediately transmitted from CAN2 port, and vise versa. There are additional configurable functions to filter and modify the passing through data:

- packet acceptance filter for every port based on CAN ID mask matching,
- baudrate configuration for every port,
- bitwise ID and data modification of the passing through packets (controlled by mask matching conditions)

Device specification:

Parameter	Value
Power supply voltage	5V-20V
Current consumption	60mA at 5V input
CAN baudrate	up to 1Mbps (any non-standard baudrate supported)
CAN ID mask filters	2 (1 for each CAN channel)
CAN data replacement patterns	40 (only for CAN1)
Microcontroller	STM32F105R8T6
CAN transceiver	SN65HVD232DR
PCB size	48.26 mm x 20.85 mm (1.9 in x 0.82 in)

Board connections:



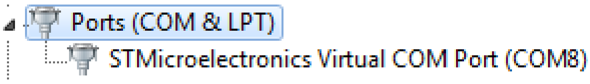
LEDs:

- Green: power on, also toggling every time when the CAN1 receiving and accepting message,
- Blue: toggling every time when the CAN2 receiving and accepting message,
- Red: constantly on if initialization failed or toggling after transmission fault.

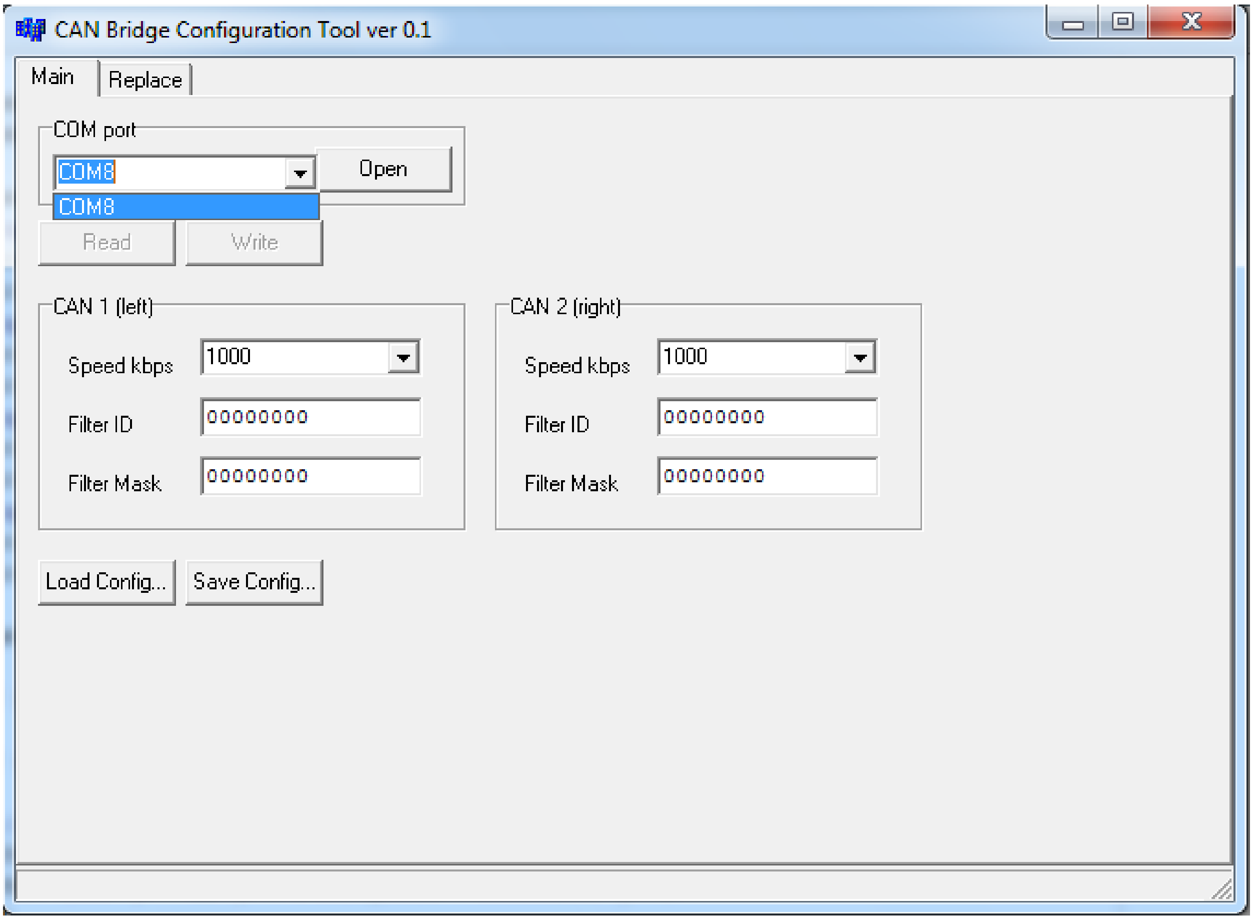
1. How to install and run software

1) Visit www.st.com and search for **STM32 Virtual COM Port Driver** in search box. Then download and install driver. You can try direct link for 1.4.0 version

2) Connect the CAN bridge using USB mini cable to computer. Make sure that the green LED is on the board and you have STMicroelectronics Virtual COM Port in Device Manager under COM ports



3) Download and run **CAN Bridge Configuration Tool**. (This is a link to Goggle sites, so probably you should say "Yes" to download). Just unzip, no installation required.
4) From the drop-down menu select COM port corresponding to CAN bridge and click *Open*.



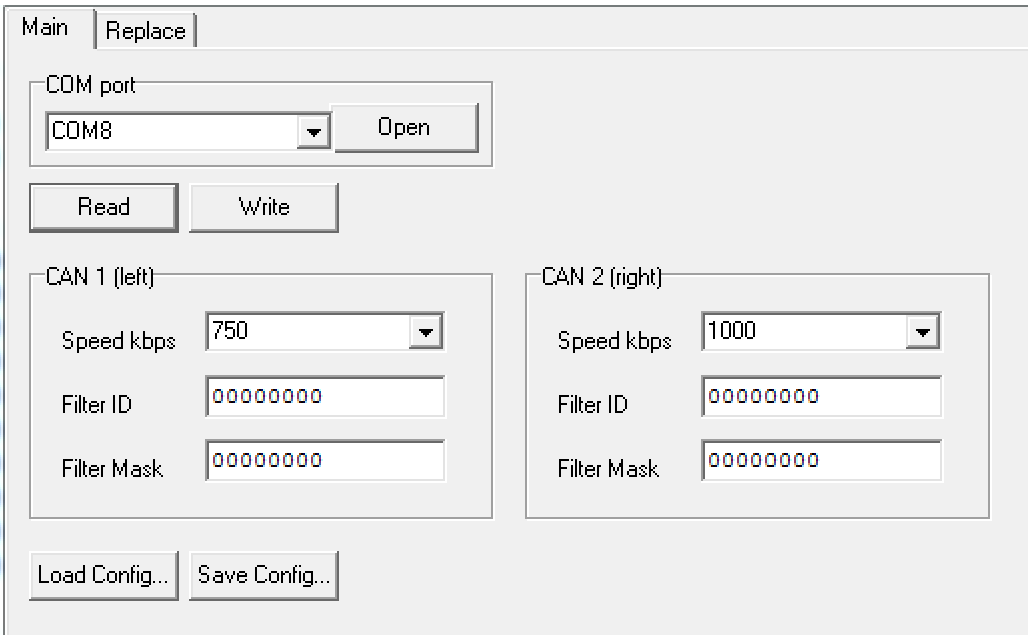
5) You can read parameters from the device by clicking *Read*
6) You can write new settings by clicking *Write* button. **Warning**: **new settings will be applied only after power cycle of the device.**

If COM port is not listed in drop-down menu, try to reconnect USB device. If it doesn't help, then check the driver installation. Try to download and install the latest driver.
If *Read* button doesn't work, i.e. its saying "Failed to send data" or "Failed to read data" try to check if the COM port you open is really corresponds to device.

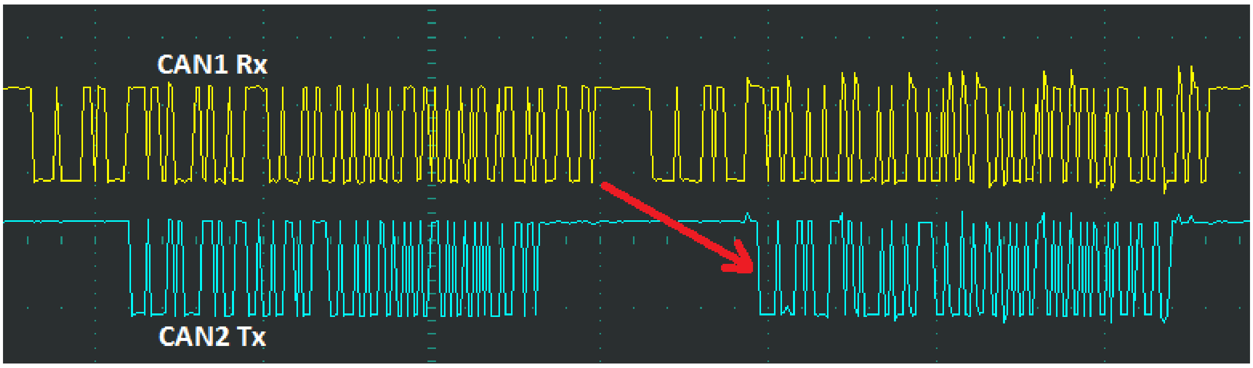
2. How to change baudrate

You can select from drop-down list desirable baudrate value separately for CAN 1 and CAN2, or you can enter any integer number.

Here is example of speed conversion. CAN1 is transmitting and receiving at 750 kbps and CAN2 is transmitting and receiving at 1000 kbps.



Here is oscilloscope waveform for stream downconversion from CAN1 to CAN2 corresponding to settings above:



Important notice: because of lack of internal buffer be careful with baudrate downconversion. In this example make sure that the next packet is received at CAN2 after the previous one is started for transmission at CAN1.

3. How to setup ID filter

The packet acceptance filter is controlled by two hexadecimal values:

1) Filter ID

Each bit specifies the level of the corresponding bit of the expected identifier:

0: Dominant (0) bit is expected

1: Recessive (1) bit is expected

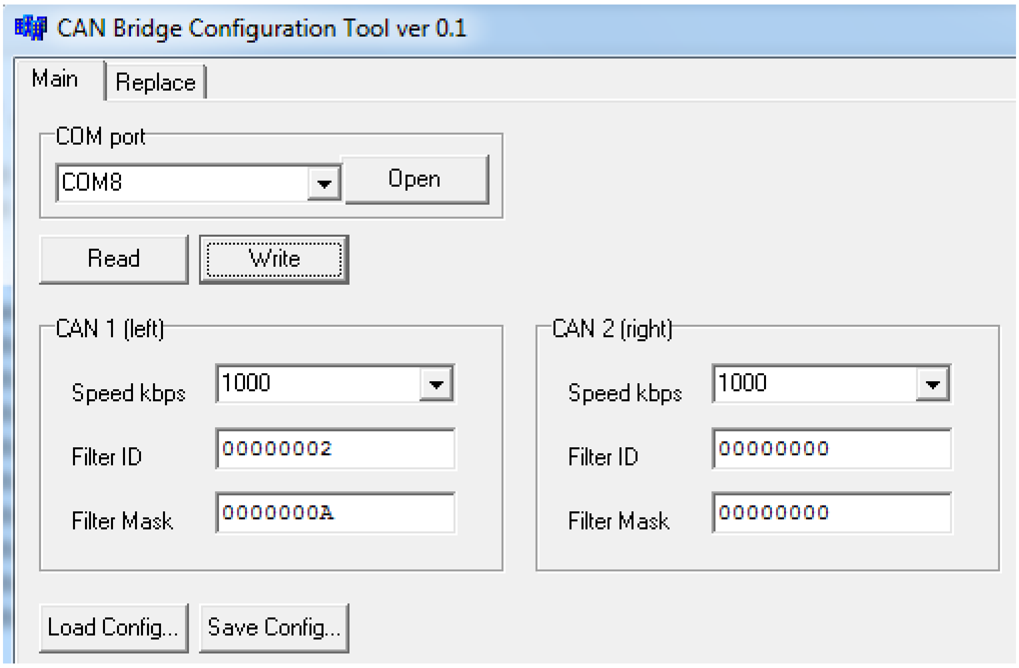
2) Mask ID

Each bit specifies whether the bit of the received identifier must match with the corresponding bit of the expected identifier or not:

0: Don't care, the bit is not used for the comparison

1: Must match, the bit of the incoming identifier must have the same level has specified in the corresponding identifier parameter filter (Filter ID).

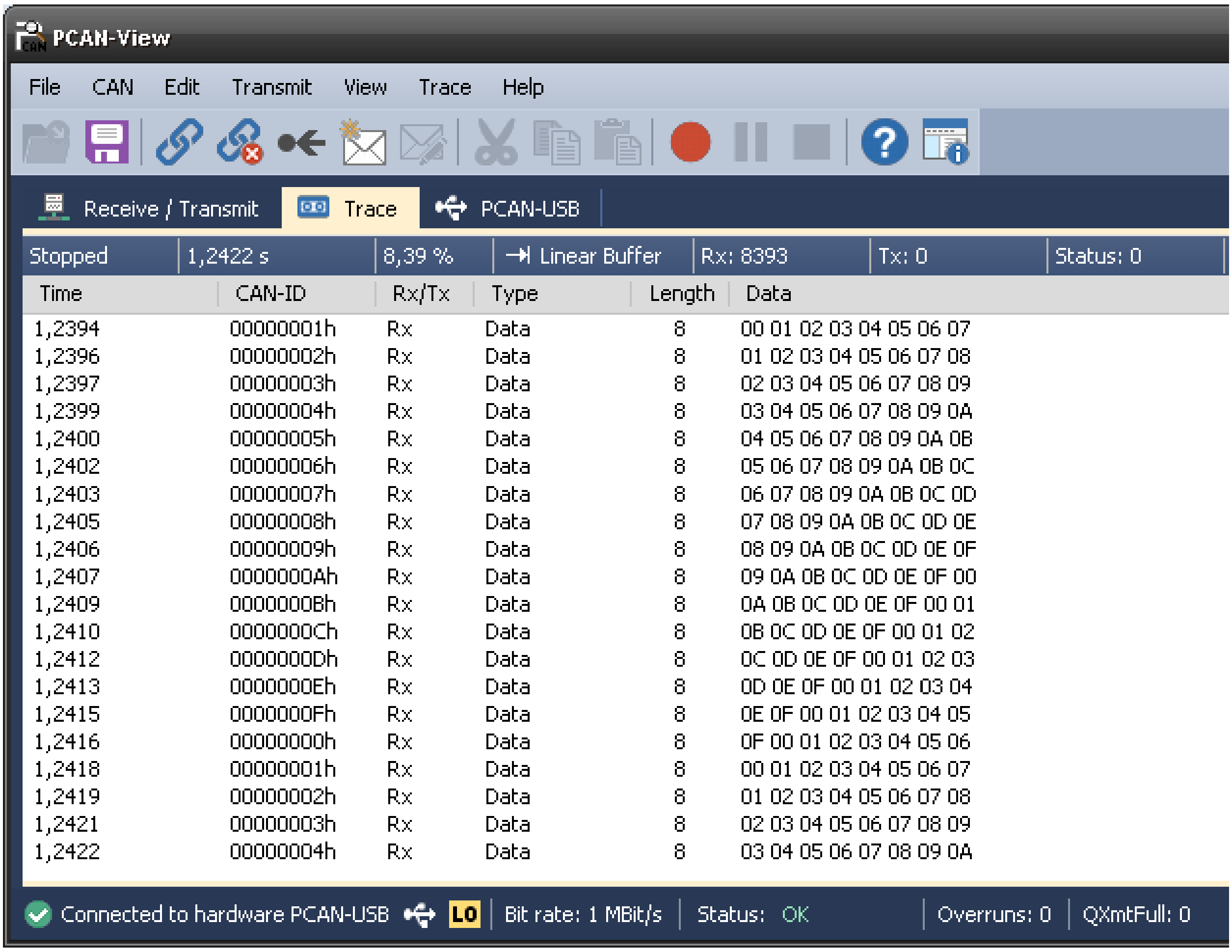
As example considering following configuration for CAN 1



The Filter Mask is 0xA = 1010 binary and the Filter ID is 2 = 0010 binary. It means that the bit #1 and bit #3 of CAN identifier will be checked, and the bit #1 is expected to be 1 and bit #3 to be 0. Thus only identifiers with binary ending of …0X1X will be accepted by CAN1, i.e. in hex 0x?2, 0x?3, 0x?6, 0x?7.

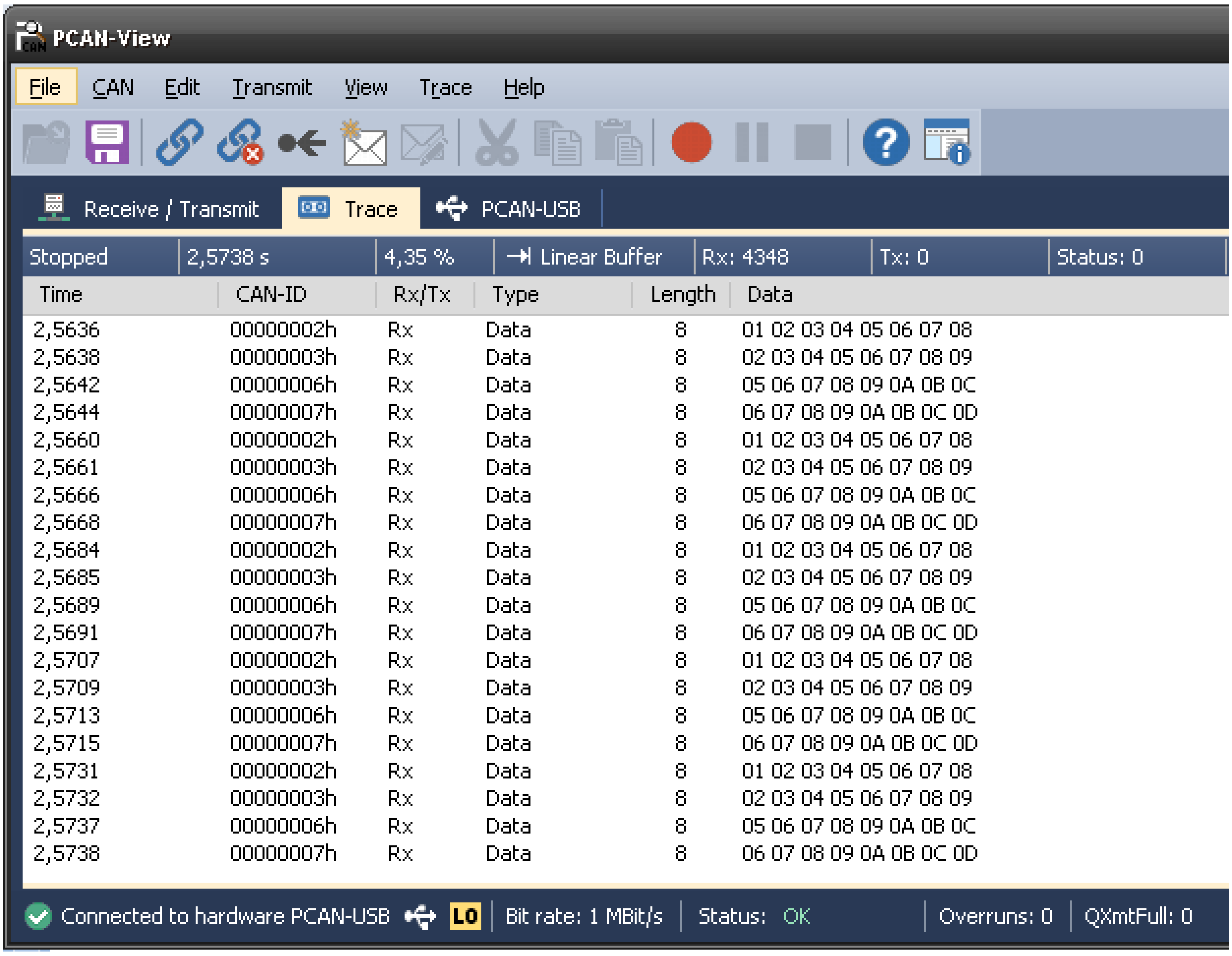
Lets test this filter using CAN bus monitoring software.

Here is input stream of data at CAN1. This is basically 16 different CAN messages with identifiers from 0x0 to 0xF.



Time	CAN-ID	Rx/Tx	Type	Length	Data
1,2394	00000001h	Rx	Data	8	00 01 02 03 04 05 06 07
1,2396	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
1,2397	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
1,2399	00000004h	Rx	Data	8	03 04 05 06 07 08 09 0A
1,2400	00000005h	Rx	Data	8	04 05 06 07 08 09 0A 0B
1,2402	00000006h	Rx	Data	8	05 06 07 08 09 0A 0B 0C
1,2403	00000007h	Rx	Data	8	06 07 08 09 0A 0B 0C 0D
1,2405	00000008h	Rx	Data	8	07 08 09 0A 0B 0C 0D 0E
1,2406	00000009h	Rx	Data	8	08 09 0A 0B 0C 0D 0E 0F
1,2407	0000000Ah	Rx	Data	8	09 0A 0B 0C 0D 0E 0F 00
1,2409	0000000Bh	Rx	Data	8	0A 0B 0C 0D 0E 0F 00 01
1,2410	0000000Ch	Rx	Data	8	0B 0C 0D 0E 0F 00 01 02
1,2412	0000000Dh	Rx	Data	8	0C 0D 0E 0F 00 01 02 03
1,2413	0000000Eh	Rx	Data	8	0D 0E 0F 00 01 02 03 04
1,2415	0000000Fh	Rx	Data	8	0E 0F 00 01 02 03 04 05
1,2416	00000000h	Rx	Data	8	0F 00 01 02 03 04 05 06
1,2418	00000001h	Rx	Data	8	00 01 02 03 04 05 06 07
1,2419	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
1,2421	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
1,2422	00000004h	Rx	Data	8	03 04 05 06 07 08 09 0A

Here the messages passed through device and transmitted from CAN2:



Time	CAN-ID	Rx/Tx	Type	Length	Data
2,5636	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
2,5638	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
2,5642	00000006h	Rx	Data	8	05 06 07 08 09 0A 0B 0C
2,5644	00000007h	Rx	Data	8	06 07 08 09 0A 0B 0C 0D
2,5660	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
2,5661	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
2,5666	00000006h	Rx	Data	8	05 06 07 08 09 0A 0B 0C
2,5668	00000007h	Rx	Data	8	06 07 08 09 0A 0B 0C 0D
2,5684	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
2,5685	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
2,5689	00000006h	Rx	Data	8	05 06 07 08 09 0A 0B 0C
2,5691	00000007h	Rx	Data	8	06 07 08 09 0A 0B 0C 0D
2,5707	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
2,5709	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
2,5713	00000006h	Rx	Data	8	05 06 07 08 09 0A 0B 0C
2,5715	00000007h	Rx	Data	8	06 07 08 09 0A 0B 0C 0D
2,5731	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
2,5732	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
2,5737	00000006h	Rx	Data	8	05 06 07 08 09 0A 0B 0C
2,5738	00000007h	Rx	Data	8	06 07 08 09 0A 0B 0C 0D

Basically only messages with identifiers 0x2, 0x3, 0x6, 0x7 are remaining in output stream.

4. How to setup CAN data modification

It is possible to setup CAN data and ID modification patterns in the grid at page *Replace*.
Each replacement pattern is controlled by 4 hexadecimal values (for ID) and 4 arrays of hexadecimal numbers (for data):

1) ID Mask `ID Mask`

Each bit specifies whether the bit of the received identifier must match with the corresponding bit of the expected identifier or not:
0: Don't care, the bit is not used for the checking of message acceptance by its ID
1: Must match, the bit of the incoming identifier must have the same level specified in the corresponding identifier filter parameter (ID Filter).

2) ID Filter `ID Filter`

Each bit specifies the level of the corresponding bit of the expected identifier:
0: Dominant (0) bit is expected
1: Recessive (1) bit is expected

3) New ID Mask `New ID Mas`

If the message is being accepted for modification, each bit specifies if the corresponding bit of identifier will be replaced by value from New ID Value parameter:
0: This bit of identifier will be transmitted as in received message.
1: This bit will be replaced by corresponding bit from the New ID Value parameter

4) New ID Value `New ID Val`

Each bit specifies the new value of CAN identifier if the message is accepted for modification and the same bit in New ID Mask is equal to 1.

5) Data Mask `Data Mask [D0 .. D7]`

This is 8 byte array specifies the acceptance of CAN message for modification by particular data contents. The bytes are separated by space. The byte corresponding to first byte of CAN data (data[0]) is the most left here. It has the same meaning as ID Mask parameter but for CAN data.
0: Don't care, the bit in data byte is not used for the checking of message acceptance by data contents
1: Must match, the bit of the incoming data byte must have the same level specified in the corresponding bit and byte in parameter Data Filter.

6) Data Filter `Data Filter [D0 .. D7]`

This is 8 byte array specifies the values of data bytes expected in CAN message accepted for modification.
0: If the same bit in Data Mask equals to 1, then this bit should be 0 in the received data byte for message acceptance
1: If the same bit in Data Mask equals to 1, then this bit should be 1 in the received data byte for message acceptance

7) New Data Mask `New Data Mask [D0 .. D7]`

If the message is being accepted for modification, each bit specifies if the corresponding bit of data will be replaced by value from New Data Value parameter. This is 8 byte array, the first byte of CAN data (data[0]) is the most left here.

0: This bit of corresponding data byte will be transmitted as in received message without modification.

1: This bit will be replaced in message data by corresponding bit from the New Data Value parameter

8) New Data Value `New Data Value [D0 .. D7`

If the message is being accepted for modification, each bit specifies the new value of the bit in corresponding data byte.

Please note:

- The CAN message will be accepted for modification if both ID and data are passed their filters.
- If the message has length less than 8 bytes, leave the last bytes by 0 for Data Mask and New Data Mask.
- Arrays are treated as numbers, for example single 0 will be considered as 00 for all bytes. Also it is possible to use any number of spaces to delimit the bytes.
- After programming the new configuration setup to device it is recommended to read back the configuration and check if the formatting of arrays.

We will demonstrate the usage of the replacement patterns by two examples: separately for ID modification and for data modification.

4.1 CAN ID modification example

Let's consider following entry at Replace table:

CAN Bridge Configuration Tool ver 0.1

Main Replace

	ID Mask	ID Filter	New ID Mas	New ID Val	Data Mask [D0 .. D7]	Data Filter [D0 .. D7]	New Data Mask [D0 .. D7]	New Data Value [D0 .. D7
1	000007FF	00000108	00000118	00000018	0	0	0	0

Here is:

ID Mask = 0x7FF

ID filter = 0x108

New ID Mask = 0x118

New ID Value = 0x018

It means that all first 11 bits of ID will be checked and only messages with identifier equal to 0x?108 will accepted for modification. Then the bits #3, #4 and #8 will be modified, because 0x118=100011000 binary. And the new ID is obvious from following:

100001000 – is only accepted ID,

100011000 – are modified bits,

000011000 – are new bits,

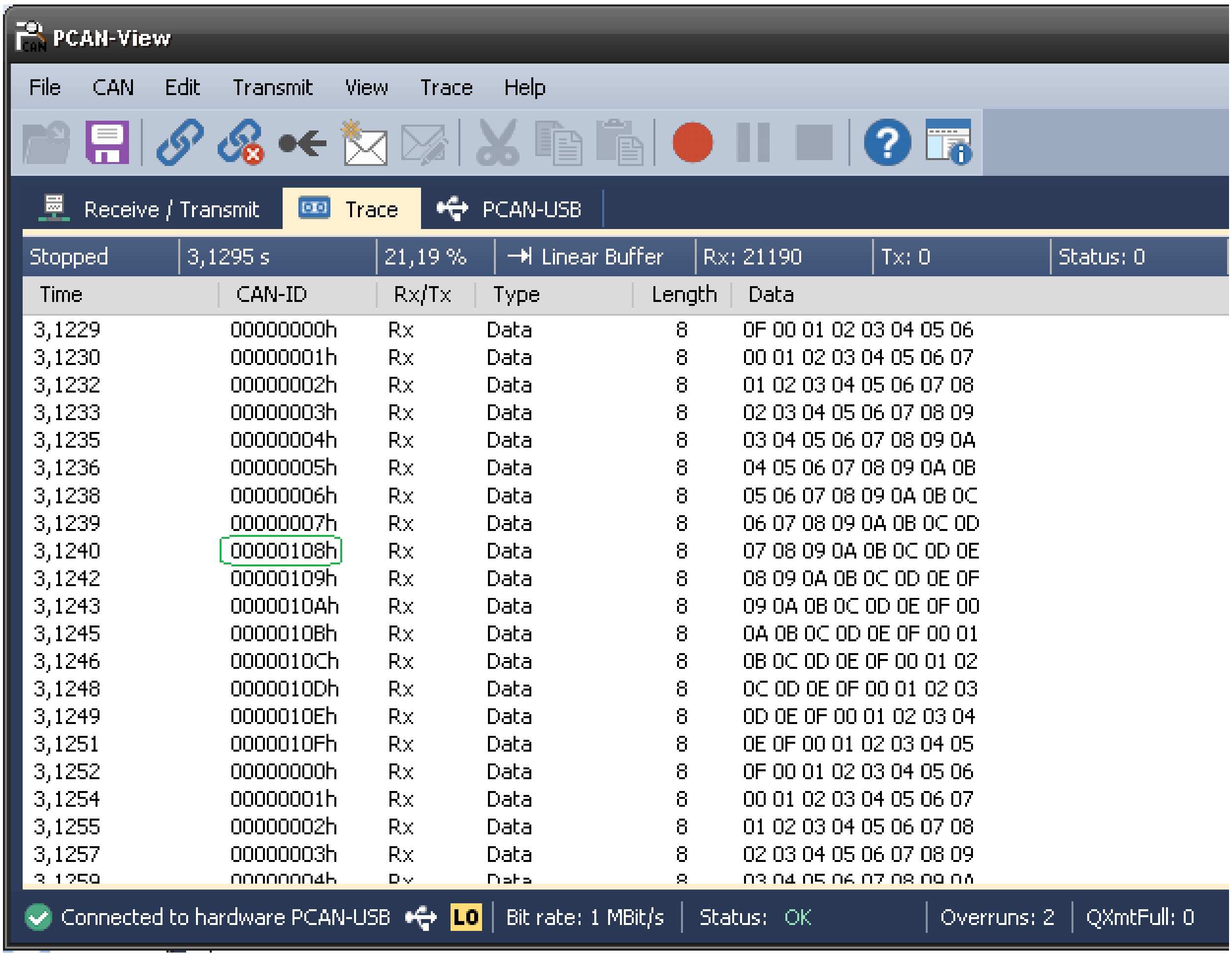
000011000 – is new ID.

(Basically the bit #3 was not changed, it should be the same results with New ID Mask = 0x110

New ID Value = 0x010)

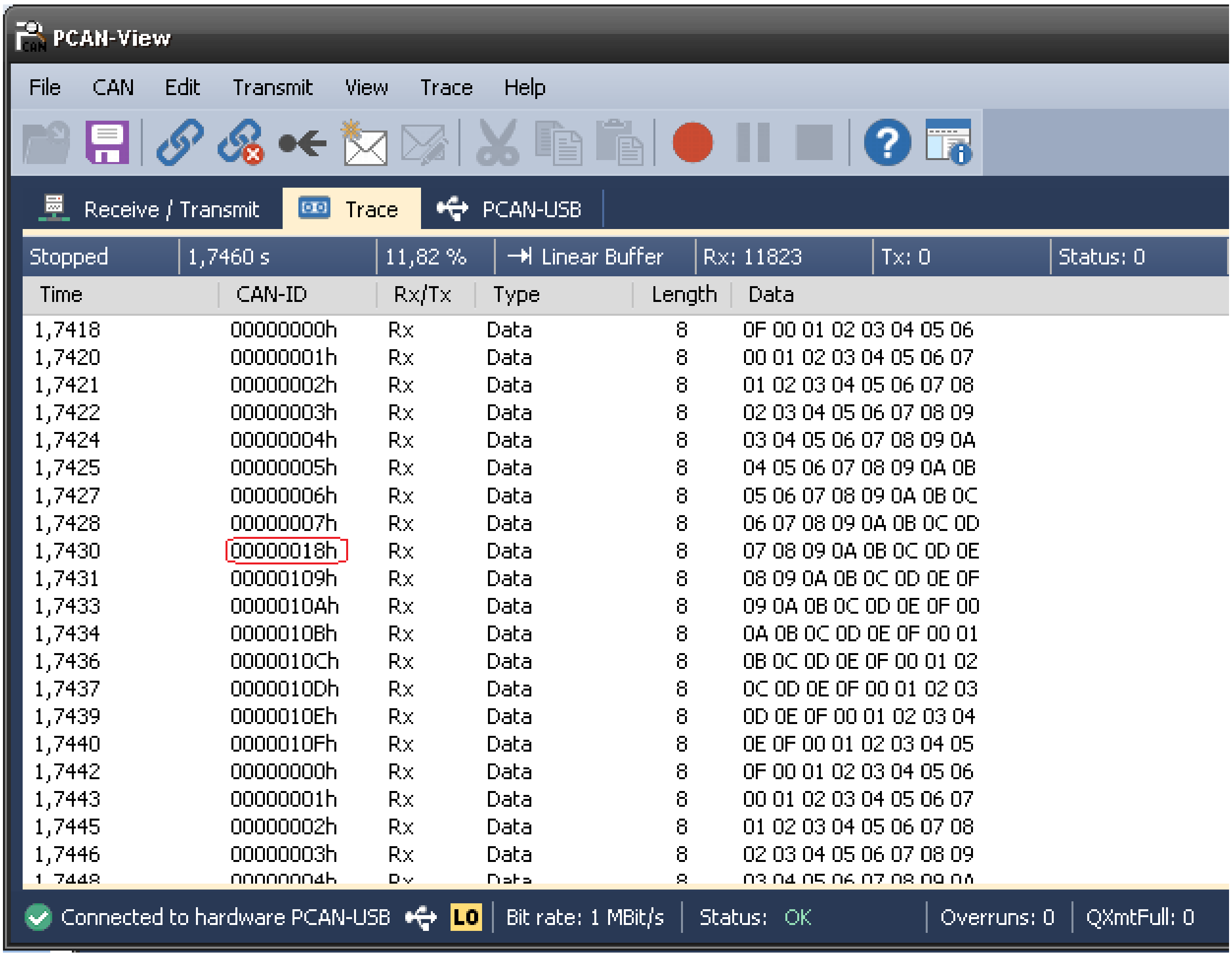
All parameters for data all 0, it means we are accepting any data content and don’t touch the data bytes.

Lets test this replacement pattern using CAN bus monitoring software.
Here is input stream of data captured at CAN1 port. The ID to be modified is highlighted by green color.



Time	CAN-ID	Rx/Tx	Type	Length	Data
3,1229	00000000h	Rx	Data	8	0F 00 01 02 03 04 05 06
3,1230	00000001h	Rx	Data	8	00 01 02 03 04 05 06 07
3,1232	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
3,1233	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
3,1235	00000004h	Rx	Data	8	03 04 05 06 07 08 09 0A
3,1236	00000005h	Rx	Data	8	04 05 06 07 08 09 0A 0B
3,1238	00000006h	Rx	Data	8	05 06 07 08 09 0A 0B 0C
3,1239	00000007h	Rx	Data	8	06 07 08 09 0A 0B 0C 0D
3,1240	00000108h	Rx	Data	8	07 08 09 0A 0B 0C 0D 0E
3,1242	00000109h	Rx	Data	8	08 09 0A 0B 0C 0D 0E 0F
3,1243	0000010Ah	Rx	Data	8	09 0A 0B 0C 0D 0E 0F 00
3,1245	0000010Bh	Rx	Data	8	0A 0B 0C 0D 0E 0F 00 01
3,1246	0000010Ch	Rx	Data	8	0B 0C 0D 0E 0F 00 01 02
3,1248	0000010Dh	Rx	Data	8	0C 0D 0E 0F 00 01 02 03
3,1249	0000010Eh	Rx	Data	8	0D 0E 0F 00 01 02 03 04
3,1251	0000010Fh	Rx	Data	8	0E 0F 00 01 02 03 04 05
3,1252	00000000h	Rx	Data	8	0F 00 01 02 03 04 05 06
3,1254	00000001h	Rx	Data	8	00 01 02 03 04 05 06 07
3,1255	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
3,1257	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
3,1259	00000004h	Rx	Data	8	03 04 05 06 07 08 09 0A

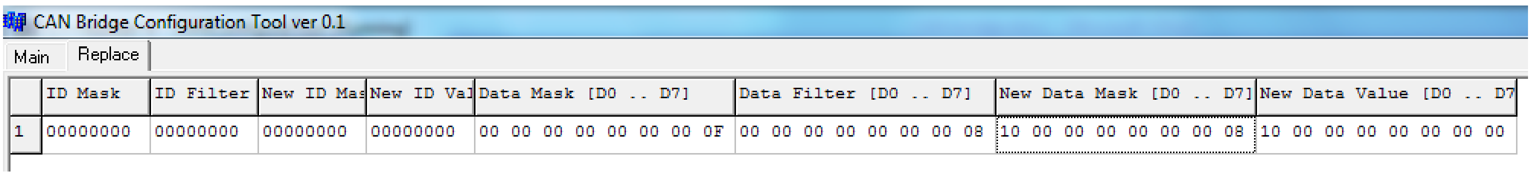
And here is data passed through device to CAN2 port. As results the identifier 0x108 changed to 0x18 in CAN data stream:



Time	CAN-ID	Rx/Tx	Type	Length	Data
1,7418	00000000h	Rx	Data	8	0F 00 01 02 03 04 05 06
1,7420	00000001h	Rx	Data	8	00 01 02 03 04 05 06 07
1,7421	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
1,7422	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
1,7424	00000004h	Rx	Data	8	03 04 05 06 07 08 09 0A
1,7425	00000005h	Rx	Data	8	04 05 06 07 08 09 0A 0B
1,7427	00000006h	Rx	Data	8	05 06 07 08 09 0A 0B 0C
1,7428	00000007h	Rx	Data	8	06 07 08 09 0A 0B 0C 0D
1,7430	00000018h	Rx	Data	8	07 08 09 0A 0B 0C 0D 0E
1,7431	00000109h	Rx	Data	8	08 09 0A 0B 0C 0D 0E 0F
1,7433	0000010Ah	Rx	Data	8	09 0A 0B 0C 0D 0E 0F 00
1,7434	0000010Bh	Rx	Data	8	0A 0B 0C 0D 0E 0F 00 01
1,7436	0000010Ch	Rx	Data	8	0B 0C 0D 0E 0F 00 01 02
1,7437	0000010Dh	Rx	Data	8	0C 0D 0E 0F 00 01 02 03
1,7439	0000010Eh	Rx	Data	8	0D 0E 0F 00 01 02 03 04
1,7440	0000010Fh	Rx	Data	8	0E 0F 00 01 02 03 04 05
1,7442	00000000h	Rx	Data	8	0F 00 01 02 03 04 05 06
1,7443	00000001h	Rx	Data	8	00 01 02 03 04 05 06 07
1,7445	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
1,7446	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
1,7448	00000004h	Rx	Data	8	03 04 05 06 07 08 09 0A

4.2 CAN data modification example

Let's consider another one entry in Replace table



	ID Mask	ID Filter	New ID Mas	New ID Val	Data Mask [D0 .. D7]	Data Filter [D0 .. D7]	New Data Mask [D0 .. D7]	New Data Value [D0 .. D7
1	00000000	00000000	00000000	00000000	00 00 00 00 00 00 00 0F	00 00 00 00 00 00 00 08	10 00 00 00 00 00 00 08	10 00 00 00 00 00 00 00

Data Mask = 00 00 00 00 00 00 00 0F

Data Filter = 00 00 00 00 00 00 00 08

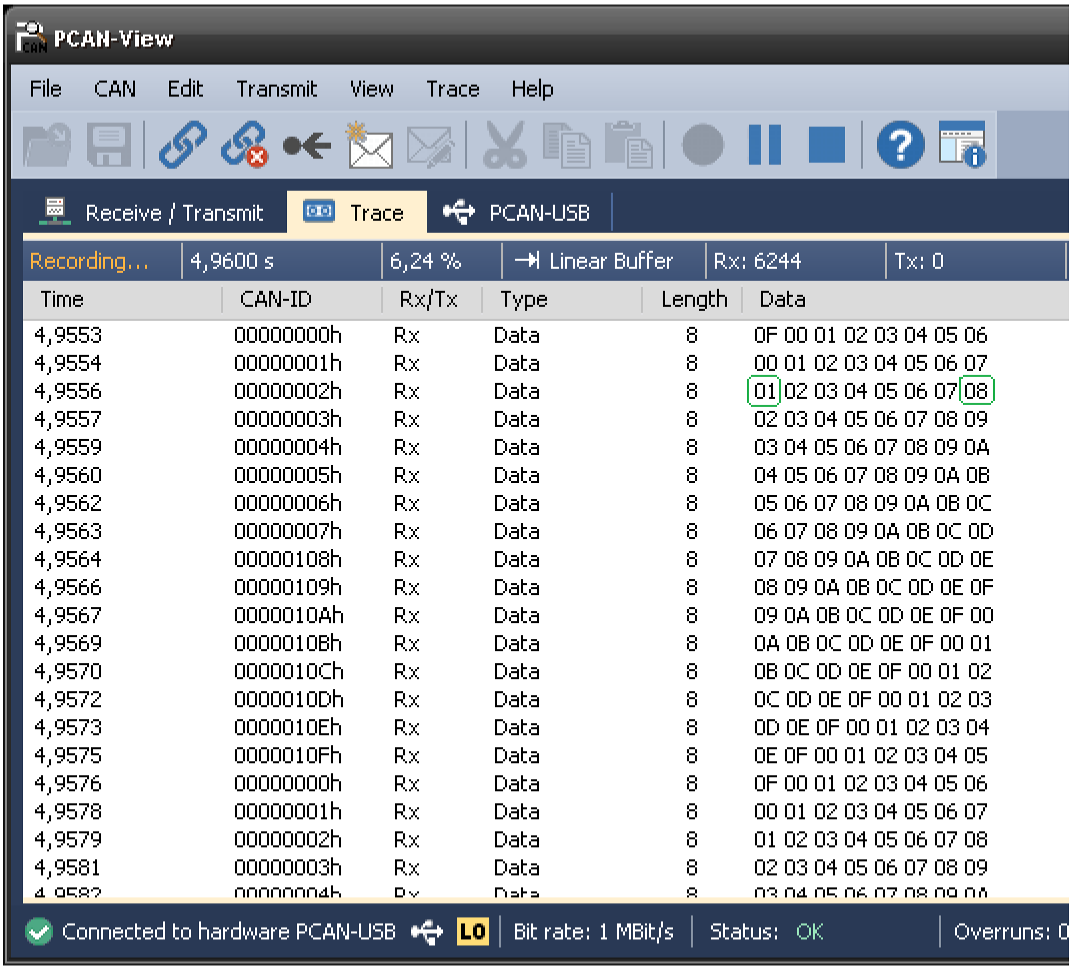
New Data Mask = 10 00 00 00 00 00 00 08

New Data Value = 10 00 00 00 00 00 00 00

As result the only messages with byte #7 in form of 0x?8 will be accepted. And then bit #3 in this byte will be reset to zero. Also the bit #4 in first byte will be set to one.

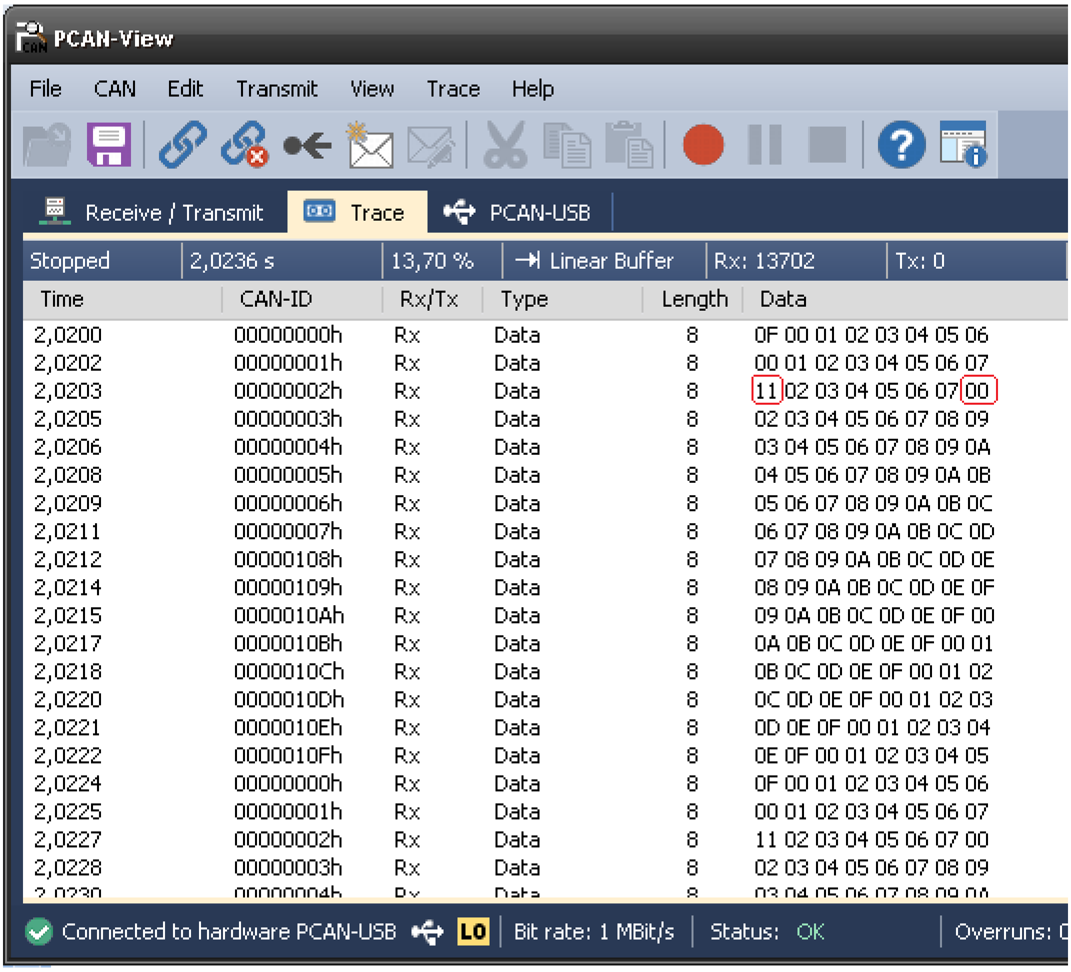
Lets test this replacement pattern using CAN bus monitoring software.

Here is input stream of data capture at CAN1 port. The data bytes to be modified is highlighted by green color:



Time	CAN-ID	Rx/Tx	Type	Length	Data
4,9553	00000000h	Rx	Data	8	0F 00 01 02 03 04 05 06
4,9554	00000001h	Rx	Data	8	00 01 02 03 04 05 06 07
4,9556	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
4,9557	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
4,9559	00000004h	Rx	Data	8	03 04 05 06 07 08 09 0A
4,9560	00000005h	Rx	Data	8	04 05 06 07 08 09 0A 0B
4,9562	00000006h	Rx	Data	8	05 06 07 08 09 0A 0B 0C
4,9563	00000007h	Rx	Data	8	06 07 08 09 0A 0B 0C 0D
4,9564	00000108h	Rx	Data	8	07 08 09 0A 0B 0C 0D 0E
4,9566	00000109h	Rx	Data	8	08 09 0A 0B 0C 0D 0E 0F
4,9567	0000010Ah	Rx	Data	8	09 0A 0B 0C 0D 0E 0F 00
4,9569	0000010Bh	Rx	Data	8	0A 0B 0C 0D 0E 0F 00 01
4,9570	0000010Ch	Rx	Data	8	0B 0C 0D 0E 0F 00 01 02
4,9572	0000010Dh	Rx	Data	8	0C 0D 0E 0F 00 01 02 03
4,9573	0000010Eh	Rx	Data	8	0D 0E 0F 00 01 02 03 04
4,9575	0000010Fh	Rx	Data	8	0E 0F 00 01 02 03 04 05
4,9576	00000000h	Rx	Data	8	0F 00 01 02 03 04 05 06
4,9578	00000001h	Rx	Data	8	00 01 02 03 04 05 06 07
4,9579	00000002h	Rx	Data	8	01 02 03 04 05 06 07 08
4,9581	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
4,9582	00000004h	Rx	Data	8	03 04 05 06 07 08 09 0A

And here is data passed through device to CAN2 port. As results the identifier the first byte is changed from 0x01 to 0x11 and the last one byte is reset to 0.



Time	CAN-ID	Rx/Tx	Type	Length	Data
2,0200	00000000h	Rx	Data	8	0F 00 01 02 03 04 05 06
2,0202	00000001h	Rx	Data	8	00 01 02 03 04 05 06 07
2,0203	00000002h	Rx	Data	8	11 02 03 04 05 06 07 00
2,0205	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
2,0206	00000004h	Rx	Data	8	03 04 05 06 07 08 09 0A
2,0208	00000005h	Rx	Data	8	04 05 06 07 08 09 0A 0B
2,0209	00000006h	Rx	Data	8	05 06 07 08 09 0A 0B 0C
2,0211	00000007h	Rx	Data	8	06 07 08 09 0A 0B 0C 0D
2,0212	00000108h	Rx	Data	8	07 08 09 0A 0B 0C 0D 0E
2,0214	00000109h	Rx	Data	8	08 09 0A 0B 0C 0D 0E 0F
2,0215	0000010Ah	Rx	Data	8	09 0A 0B 0C 0D 0E 0F 00
2,0217	0000010Bh	Rx	Data	8	0A 0B 0C 0D 0E 0F 00 01
2,0218	0000010Ch	Rx	Data	8	0B 0C 0D 0E 0F 00 01 02
2,0220	0000010Dh	Rx	Data	8	0C 0D 0E 0F 00 01 02 03
2,0221	0000010Eh	Rx	Data	8	0D 0E 0F 00 01 02 03 04
2,0222	0000010Fh	Rx	Data	8	0E 0F 00 01 02 03 04 05
2,0224	00000000h	Rx	Data	8	0F 00 01 02 03 04 05 06
2,0225	00000001h	Rx	Data	8	00 01 02 03 04 05 06 07
2,0227	00000002h	Rx	Data	8	11 02 03 04 05 06 07 00
2,0228	00000003h	Rx	Data	8	02 03 04 05 06 07 08 09
2,0230	00000004h	Rx	Data	8	03 04 05 06 07 08 09 0A