Telemotive AG

SimBox

User Manual

05.02.2013

Version: 1.0

Table of Contents

Table of Figures

LICENSE AGREEMENT

PLEASE READ THE LICENSE AGREEMENT OF THIS LICENSE CONTRACT CAREFULLY, BEFORE YOU INSTALL THE SOFTWARE. BY THE INSTALLATION OF THE SOFTWARE YOU AGREE TO THE CONDITIONS OF THIS LICENSE CONTRACT.

THIS SOFTWARE-LICENSE AGREEMENT, IN THE FOLLOWING CALLED AS "LICENSE", CONTAINS ALL RIGHTS AND RESTRICTIONS FOR FINAL USERS THAT REGULATE THE USE OF THE ACCOMPANYING SOFTWARE, OPERATING INSTRUCTIONS AND OTHER DOCUMENTS, IN THE FOLLOWING CALLED AS "SOFTWARE".

1. THIS LICENSE CONTRACT IS AN AGREEMENT BETWEEN LICENSOR AND LICENSEE, WHO IS BEING LICENSED TO USE THE NAMED SOFTWARE.
2. LICENSEE ACKNOWLEDGES THAT THIS IS ONLY A LIMITED NONEXCLUSIVE LICENSE. THIS MEANS THAT THE LICENSEE HAS NO RIGHT TO ALLOCATE SUBLICENSES. LICENSOR IS AND REMAINS THE OWNER OF ALL TITLES, RIGHTS, AND INTERESTS IN THE SOFTWARE.
3. THE SOFTWARE IS A COPYRIGHT PROPERTY OF THE TELEMOTIVE AG. THE PROGRAM OR PARTS OF IT MAY NOT BE FURTHER LICENSED TO THIRD PARTS, RENTED, SELLS, OR BE FURTHER MARKETED, OTHERWISE, IN ANY FORM WITHOUT EXPLICIT WRITTEN APPROVAL BY TELEMOTIVE AG. THE USER MAY NEITHER CHANGE THE SOFTWARE AND THEIR COMPONENTS, MODIFY NOR, OTHERWISE, REDEVELOPMENT OR DECOMPILE IN ANY FORM.
4. THIS SOFTWARE IS SUBJECT TO NO WARRANTY. THIS SOFTWARE IS SOLD AS IS, WITHOUT ANY WARRANTY. IF AT ANY TIME, A USER CHANGES THEIR SYSTEM, WE HOLD NO RESPONSIBILITY TO CHANGE OUR SOFTWARE TO MAKE IT WORK AGAIN.
5. THIS LICENSE PERMITS LICENSEE TO INSTALL THE SOFTWARE ON MORE THAN ONE COMPUTER SYSTEM, AS LONG AS THE SOFTWARE WILL NOT BE USED ON MORE THAN ONE COMPUTER SYSTEM SIMULTANEOUSLY. LICENSEE WILL NOT MAKE COPIES OF THE SOFTWARE OR ALLOW COPIES OF THE SOFTWARE TO BE MADE BY OTHERS, UNLESS AUTHORIZED BY THIS LICENSE AGREEMENT. LICENSEE MAY MAKE COPIES OF THE SOFTWARE FOR BACKUP PURPOSES ONLY. LICENSEE NOT ENTITLED TO TRANSMIT OR TO TRANSFER THE SOFTWARE OR YOUR RIGHTS FROM THIS LICENSE AGREEMENT.
6. LICENSOR IS NOT LIABLE TO LICENSEE FOR ANY DAMAGES, INCLUDING COMPENSATORY, SPECIAL, INCIDENTAL, EXEMPLARY, PUNITIVE, OR CONSEQUENTIAL DAMAGES, CONNECTED WITH OR RESULTING FROM THIS LICENSE AGREEMENT OR LICENSEE’S USE OF THIS SOFTWARE.
7. LICENSEE AGREES TO DEFEND AND INDEMNIFY LICENSOR AND HOLD LICENSOR HARMLESS FROM ALL CLAIMS, LOSSES, DAMAGES, COMPLAINTS, OR EXPENSES CONNECTED WITH OR RESULTING FROM LICENSEE’S BUSINESS OPERATIONS.
8. LICENSOR HAS THE RIGHT TO TERMINATE THIS LICENSE AGREEMENT AND LICENSEE’S RIGHT TO USE THIS SOFTWARE UPON ANY MATERIAL BREACH BY LICENSEE. DURATION OF THE LICENSE CONTRACT IS INDEFINITELY DETERMINED.
9. LICENSEE AGREES TO RETURN TO LICENSOR OR TO DESTROY ALL COPIES OF THE SOFTWARE UPON TERMINATION OF THE LICENSE CONTRACT.
10. THIS LICENSE AGREEMENT REPLACES AND SUPERSEDES ALL PRIOR NEGOTIATIONS, DEALINGS, AND AGREEMENTS BETWEEN LICENSOR AND LICENSEE REGARDING THIS SOFTWARE.
11. THIS LICENSE CONTRACT IS SUBJECT TO GERMAN LAW.
12. IF A REGULATION OF THIS LICENSE CONTRACT IS VOID BY LAW, THE VALIDITY OF THE REMAINING REGULATIONS IS NOT AFFECTED. IF THERE IS SUCH A REGULATION IT WILL BE REPLACED BY A VALID, ACCORDING TO THE LEGAL REGULATIONS AND ENFORCABLE REGULATION WITH SIMILAR INTENTION AND SIMILAR ECONOMIC CONSEQUENCES.
13. THE LICENSE CONTRACT IS EFFECTVE BY DELIVERY OF THE SOFTWARE OF THE LICENSOR TO THE LICENSEE AND/OR BY USAGE OF THE SOFTWARE BY THE LICENSEE. THIS LICENSE CONTRACT IS ALSO VALID WITHOUT LICENSOR’S SIGNATURE.
14. THE LICENSE AUTOMATICALLY GOES OUT IF THE LICENSEE DOES NOT AGREE TO THE LICENSE REGULATIONS DESCRIBED HERE OR OFFEND AGAINST THE LICENSE REGULATIONS OF THIS LICENSE CONTRACT. WITH ENDING THE LICENSE CONTRACT THE LICENSEE IS OBLIGED TO EXTINGUISH ALL COPIES OF THE SOFTWARE OR TO DESTROY IT.
15. THE LICENSEE STICKS FOR ALL DAMAGES WHICH ORIGINATES THE LICENSOR FROM THE INJURY OF THESE LICENSE REGULATIONS.

PRODUCT LIABILITY

FOR ALL OFFERS, SALES AND SUPPLIES DO EXPLICIT APPLY THE FOLLOWING CONDITIONS, EVEN IF THE BUYER, ORDERER AND SUCHLIKE PRESCRIBES OTHER CONDITIONS. ALTERATIONS ARE ONLY VALID, IF THEY ARE AGREED IN WRITING.

1. THE TECHNICAL DOCUMENTATION IS PART OF THE PRODUCTS. THE PRODUCT LIABILITY AND THE PRODUCT GUARANTEE WILL BE EXCLUDED, IF CONTENTS AND IN PARTICULAR THE SAFETY REFERENCES AND INSTRUCTION FOR ACTION OF THE DOCUMENTATION ARE NOT CONSIDERED.
2. THE PRODUCTS DO BELONG TO THE GROUP OF TESTTOOLS. BY APPLICATION OF THE EQUIPMENT A DISTURBANCE OT THE TESTED SYSTEM CANNOT BE COMPLETELY EXCLUDED. FOR THIS REASON, THE WARRANTY OF A PERFECTLY FUNCTIONING SYSTEM CANNOT BE TAKEN OVER BY THE MANUFACTURER.
   APPLICATION OF THE PRODUCT TAKES PLACE AT ONE'S OWN RISK.
3. THE LIABILITY OF THE SUBSTITUTION OF DAMAGES ACCORDING TO §1 PRODUCT LIABILITY LAW, IS EXPRESSLY EXCLUDED IN THE CONTEXT OF §9 PRODUCT LIABILITY LAW, AS FAR AS COMPELLING LEGAL TERMS DO NOT PROVIDE ANYTHING ELSE.

   IN NO EVENT WILL THE PRODUCER BE LIABLE FOR ANY INDIRECT, INCIDENTAL, SPECIAL OR CONSEQUENTIAL DAMAGES, INCLUDING LOSS OF PROFITS, LOSS OF REVENUES, LOSS OF DATA, LOSS OF USE, ANY OTHER ECONOMIC ADVANTAGE OR DAMAGES CAUSED BY PRETENSIONS OF THIRD PARTY TOWARDS THE CUSTOMER OUT OF THIS AGREEMENT, UNDER ANY THEORY OF LIABILITY, WHETHER IN AN ACTION IN CONTRACT, STRICT LIABILITY, TORT (INCLUDING NEGLIGENCE) OR OTHER LEGAL OR EQUITABLE THEORY.

   THE BURDEN OF PROOF IS WITH THE CUSTOMER.
4. THE TELEMOTIVE AG DOES ENSURE THE LEGAL WARRANTY ACCORDING TO GERMAN LAW.

   EXCEPT FOR WARRANTIES EXPRESSLY SET FORTH IN THIS AGREEMENT, ANY AND ALL PRODUCTS ARE DELIVERED "AS IS" AND THE PRODUCER MAKES AND THE CUSTOMER RECEIVES NO ADDITIONAL EXPRESS OR IMPLIED WARRANTIES. THE PRODUCER HEREBY EXPRESSLY DISCLAIMS ANY AND ALL OTHER WARRANTIES OF ANY KIND OR NATURE CONCERNING THE PRODUCTS, WHETHER EXPRESS OR IMPLIED, INCLUDING WITHOUT LIMITATION, ANY WARRANTY OF TITLE, MERCHANTABILITY, QUALITY, ACCURACY, OR FITNESS FOR A PARTICULAR PURPOSE OR THE CUSTOMER'S PURPOSE. THE PRODUCER EXPRESSLY DISCLAIMS ANY WARRANTIES THAT MAY BE IMPLIED FORM USAGE OF TRADE, COURSE OF DEALING, OR COURSE OF PERFORMANCE, EXCEPT FOR THE EXPRESS WARRANTIES STATED IN THIS AGREEMENT. THE PRODUCTS ARE PROVIDED WITH ALL FAULTS, AND THE ENTIRE RISK OF SATISFACTORY QUALITY, PERFORMANCE, ACCURACY, AND EFFORT IS WITH CUSTOMER. THE PRODUCER DOES NOT WARRANT THAT THE PRODUCTS WILL OPERATE WITHOUT INTERRUPTION OR BE ERROR FREE.
5. THE TELEMOTIVE AG IS JUSTIFIED TO EXCHANGE DEFECTIVE GOODS AGAINST HOMOGENEOUS ACCEPTABLE ONES OR TO ELIMINATE THE FAULT WITHIN AN APPROPRIATE PERIOD. IN THIS CASE A DEMAND FOR REDHIBITORY ACTION OR REDUCTION OF PRICE EXPIRES. WARRANTY CLAIMS PRESUPPOSE A DUE NOTICE OF DEFECTS.
6. RESALE, TRANSFER, DONATION, EXCHANGES OR THE RENTAL OF THE OFFERED PRODUCTS AT THIRD PARTY, IS PERMITTED WITHOUT CLEARANCE OF THE TELEMOTIVE AG.
7. GERMAN LAW IS DEEMED TO BE AS LEGAL BASIS.

1 Basic Information

The so-called “SimBox” (see Figure 1) is a CAN-simulating box which, on the one hand, simulates CAN messages and simple, electronic devices like ignition, electric window lift, photoelectric sensor, speed signal etc. and, on the other hand, represents a simple solution to simulate rest bus.

With the help of this SimBox, the user considerably saves costs caused by expensive prototypes of control units, oversized devices or expensive software during development phase. The SimBox helps to simulate such simple CAN messages at a more favourable price, i. e. the SimBox offers a flexible and cost-effective alternative for the majority of applications.

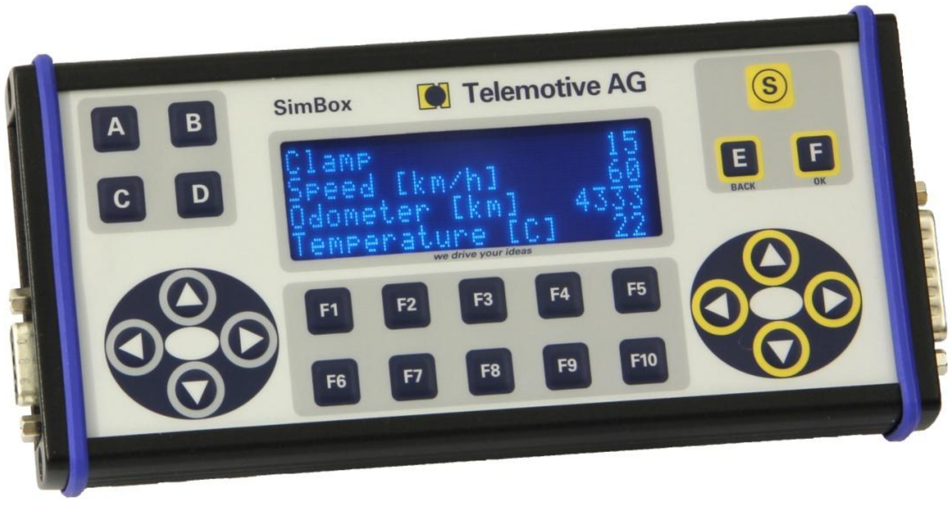


Figure 1: SimBox

<u>**Features:**</u>

Display

4 lines with 20 characters each to show status or CAN messages

Membrane keyboard

24 softkeys per configuration (configuring the SimBox/assigning specific functions to the keys is handled by the customer and not by Telemotive AG)

Start-key

Key to start the SimBox or to open menus

Power management

Wake-up signal initiated by Start-key or by CAN bus; minimum standby current during sleep mode

Two-way CAN

Possibility to select high-speed or low-speed CAN

RS 232

Configuration interface

Simple, functional client for configuration

Creating and editing configurations

Installing configurations and firmware updates on the SimBox

Functions

Sending CAN messages either by pressing the respective key or by using the adjustable timer (possible range of timer > = 10 ms)

Considering offset values to be able to convert CAN messages

Restriction: skipping values (e. g. changing from first gear to third gear)

2 Connecting the SimBox

Figure 2 illustrates the interfaces of the SimBox.

The right side offers a 26-pin SUB-D-connector which manages all CAN signals as well as any external supply. The wire harness shown in Figure 3 is included with the SimBox.

A PC or laptop may be connected with the help of its respective wire to the SUB-D-slot on the left side (RS232-port; see Figure 3).

Updates and configurations are installed through this interface. The necessary software is included with the SimBox.

This device has a micro-fuse (1 ampere).

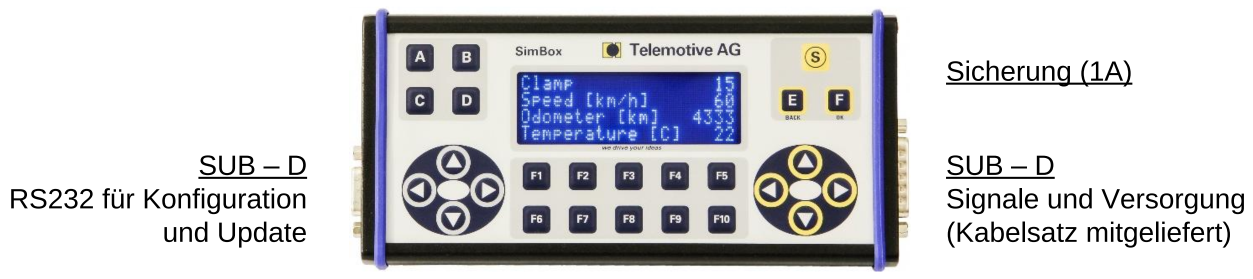


Figure 2: Interfaces of the SimBox

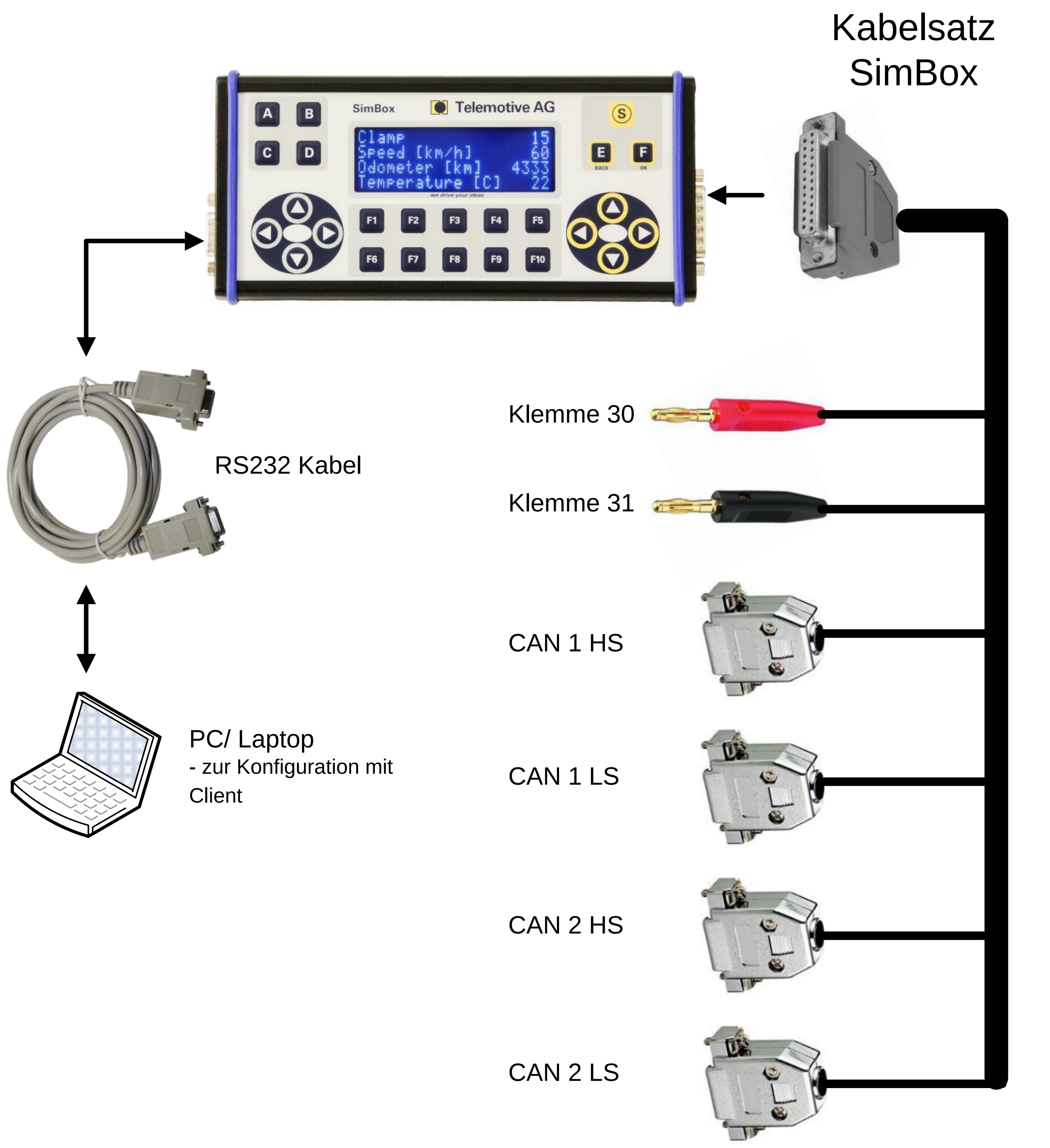


Figure 3: Connections of the SimBox

2.1 Power Supply

The SimBox is operated with a nominal voltage of 12V DC. Moreover, it may be used even if there are variations from 8,5V DC to 16V DC.

A micro-fuse (1 ampere) is available on the right side of the casing.

2.2 CAN

The SimBox offers two CAN-2.0B interfaces. Both may be configured – independently of each other – either as high-speed or low-speed CAN.

High-speed and low-speed transceivers have separate connectors and terminating resistors in order to avoid additional stress in existing CAN networks.

The CAN interfaces enable a wake-up of the SimBox based on the selected transceiver (high-speed/low-speed). The SimBox automatically switches on as soon as it recognizes any bus activity.

2.3 Operating Elements

Figure 4 presents the keyboard and the display of the SimBox.

It consists of 25 keys and a display having 4 lines with 20 characters each.

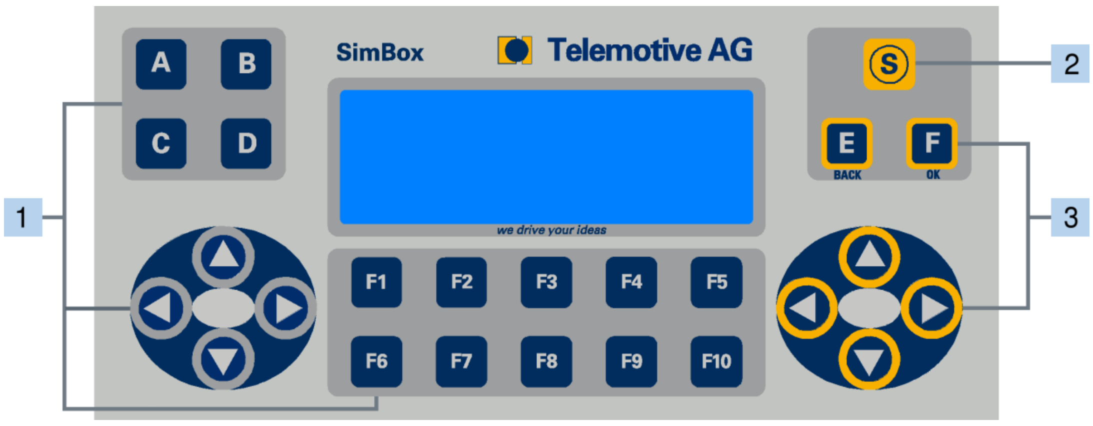


Figure 4: Keyboard and display of the SimBox

(1) Softkeys
(2) Start/Setup: activating the SimBox, accessing system functions
(3) Softkeys whose configuration/function may vary temporarily in system mode

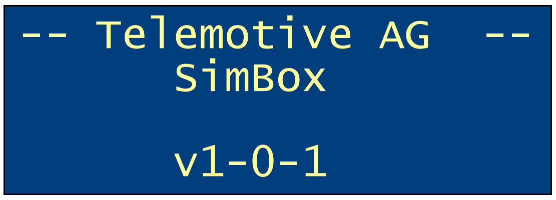
2.4 Activation and Standby

The SimBox may be activated manually by pressing the Start/Setup-key. Moreover, it starts as soon as any activity is registered on the CAN channels.

In the case of sustained power supply, the SimBox remembers the most recently activated transceiver. However, it is only possible to check one transceiver for any activity on both CAN channels. If the high-speed CAN was activated, the device may only start due to activity on the high-speed CAN. Thus, the low-speed CAN is not observed.

Various occasions cause the SimBox to turn into sleep mode with minimum standby current. This happens, for instance, after configuration or firmware have been updated. In case the sleeping SimBox is connected to a CAN bus with running bus activity and, moreover, the appropriate transceiver of the SimBox was active, it receives immediately a CAN message with a wake-up signal and, thus, does not have to be started manually.

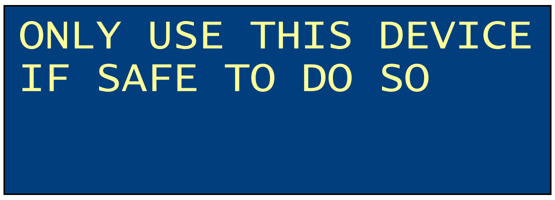
During start-up, the SimBox temporarily shows a start screen (see Figure 5) followed by a disclaimer (see Figure 6) which is displayed for approximately three seconds.



```
-- Telemotive AG  --
      SimBox

      v1-0-1
```

Figure 5: Start screen



```
ONLY USE THIS DEVICE
IF SAFE TO DO SO
```

Figure 6: Disclaimer

2.5 Standard Operating Mode

During start-up, the device imports its configuration from the flash memory. The user may define this configuration which influences the SimBox's functions and any displayed information.

2.6 Special Keys

All keys of the keyboard may be configured freely. However, this does not include the key "S" (Start/Setup) as it serves to start and activate the system menu.

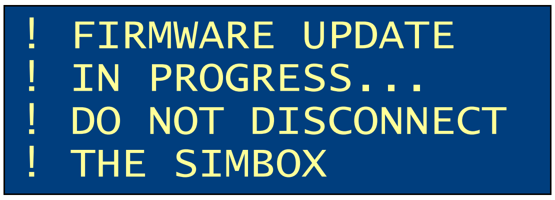
In case the system menu has been activated, all yellow-framed keys are reserved to navigate within the menu. Thus, the function, which is normally allocated to these keys, is not available as soon as the system menu is active. Having left the system menu, these keys reobtain their configured function.

Note: Key functions in system mode depend on the used firmware version.

2.7 Firmware Update

In case a new firmware is released, the user may install it himself. Please refer to chapter 3.5: Firmware Update.

Please note: During its update, the SimBox may not be disconnected from the computer or power supply!



```
! FIRMWARE UPDATE
! IN PROGRESS...
! DO NOT DISCONNECT
! THE SIMBOX
```

Figure 7: Firmware update

3 SimBox Client

With the help of a client application, the user is able to configure the SimBox as well as to adjust it to specific purposes. Installing this client application requires a computer with Microsoft Windows.

3.1 Installing the Client

The SimBox client is provided as installation package with an integrated setup programme. To install the client, simply start the installation programme and follow all instructions given on the screen. It may be necessary to have administrator rights for installation.

The installation programme copies the client, its respective deinstallation programme and all necessary DLLs to the computer. Moreover, it creates an entry in the start menu.

To remove the client, simply start the deinstallation programme or, alternatively, delete the client's directory and menu entry manually.

3.2 Start Window

Having started the SimBox client, the following start window appears:

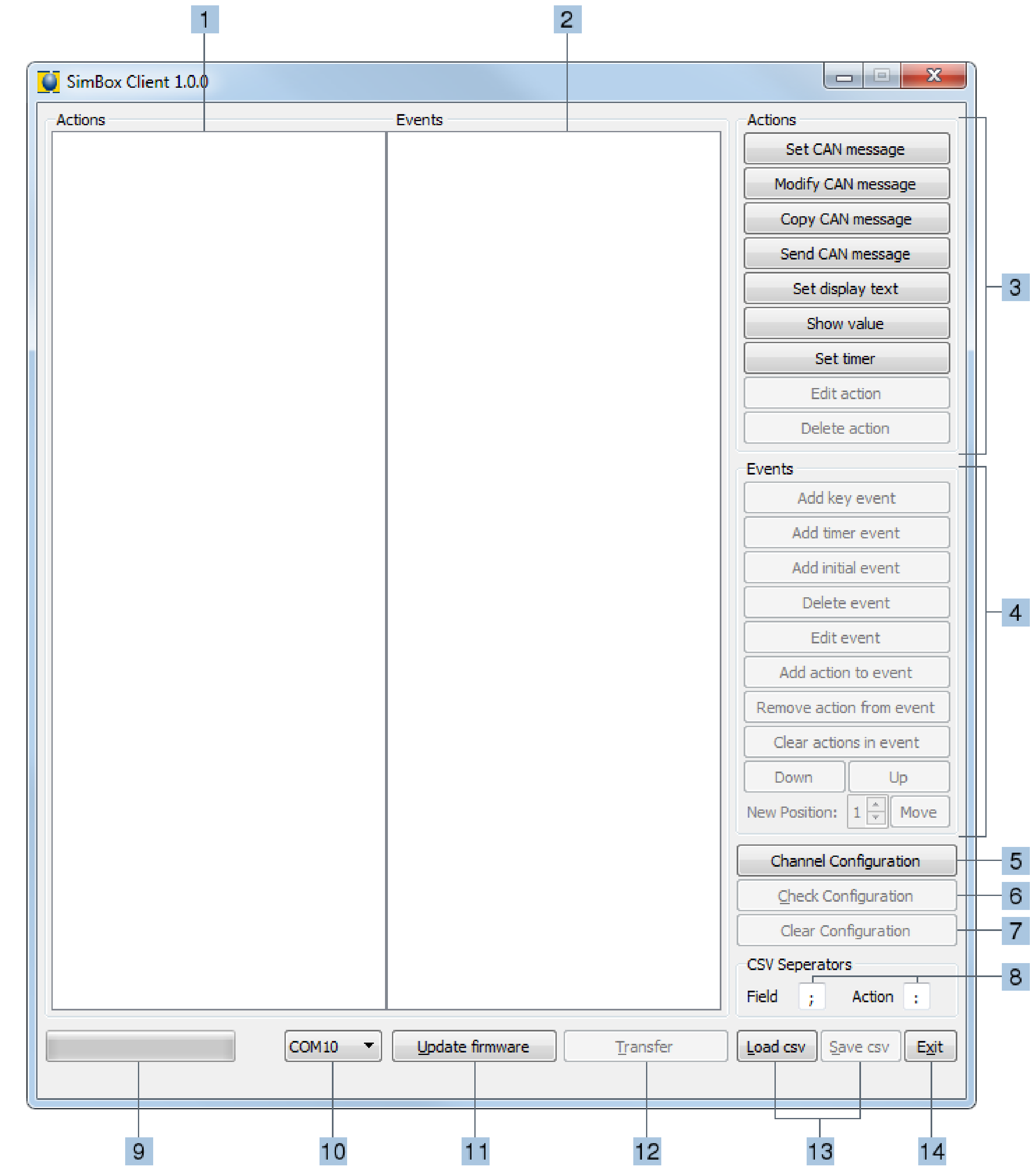


Figure 8: Start window of SimBox client

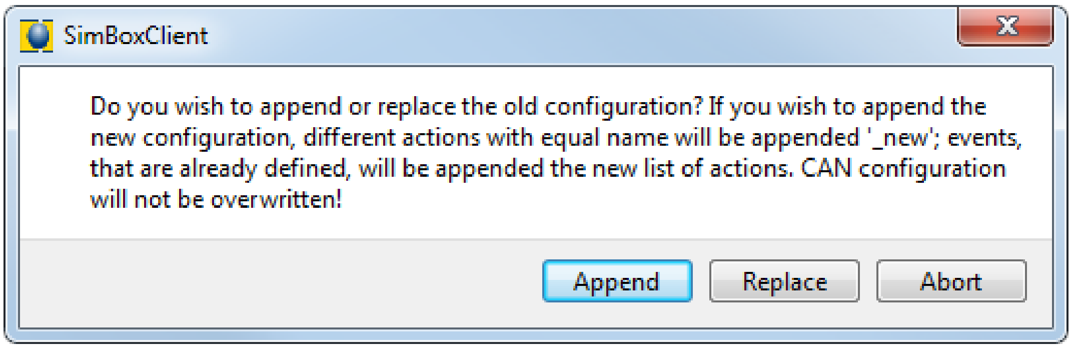
(1) List of configured actions
(2) List of configured events
(3) Create and edit actions
(4) Create and edit events
(5) Configure CAN interfaces as well as their respective names
(6) Check active configuration
(7) Create empty configuration
(8) Set parameter of CSV file
(9) Progress indicator of configuration and firmware downloads
(10) Select COM interface
(11) Start firmware update
(12) Send configuration to SimBox
(13) Open and save a configuration file
(14) End programme

The client is divided into two lists: the left one summarizes the configured actions; the right one shows the events. Control keys are available on the right side:

The group box “Actions” allows to create, manipulate and delete actions. The group box “Events”, which is located below “Actions”, allows the same possibilities with regard to events. The bottom bar of the client contains keys which enable the user to open, save and send a configuration to the SimBox as well as to execute a firmware update.

3.3 Opening a Configuration

An existing configuration may be opened by pressing “Load CSV“ (see Figure 8, number 13). This command leads to a dialogue window to select directory and file respectively. All configuration files need to be available in CSV format. In case another configuration is already active in the client, the system asks if the new file shall be added to the old configuration, if the old configuration shall be replaced by the new one or if the opening process shall be aborted.



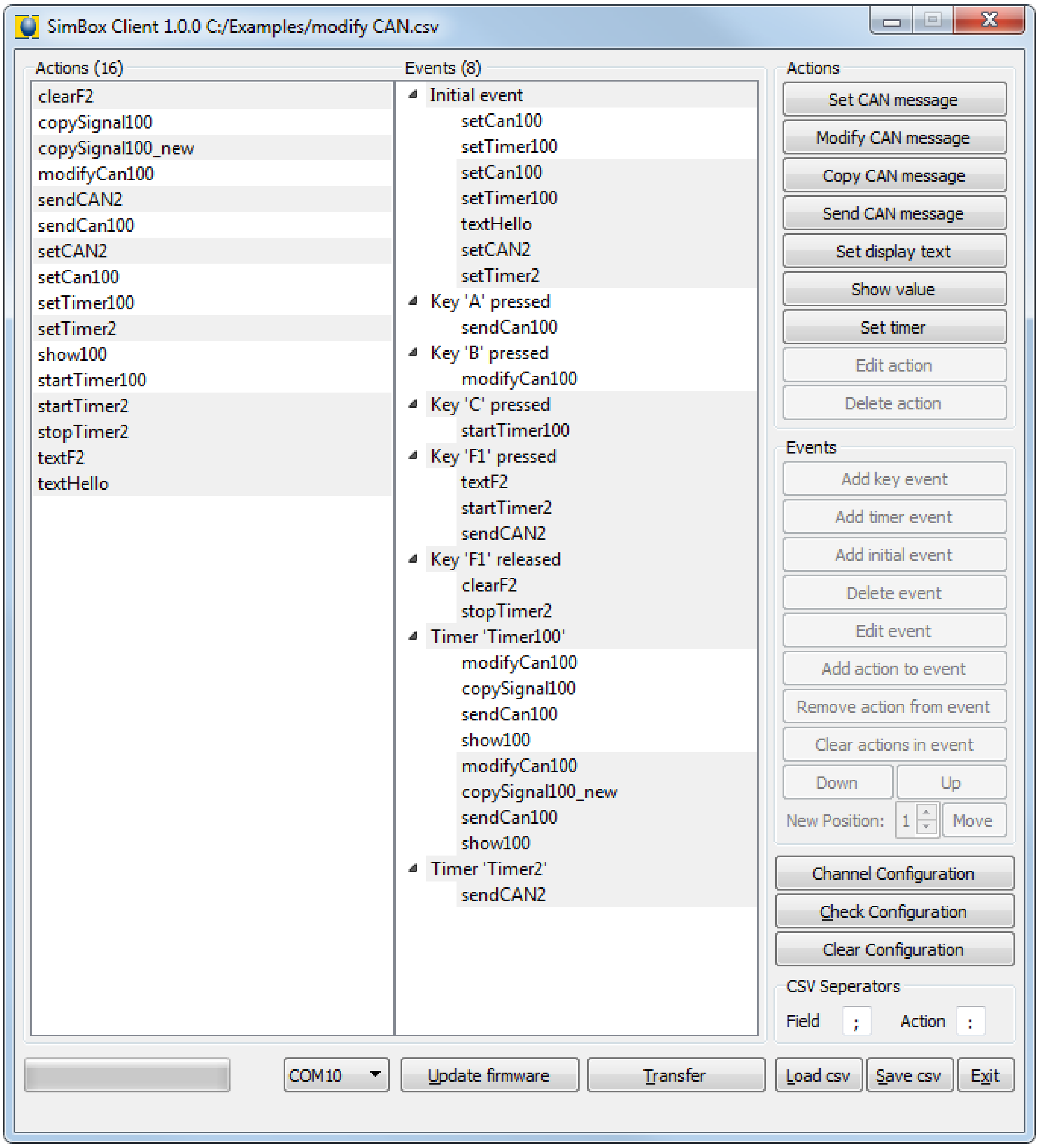
Do you wish to append or replace the old configuration? If you wish to append the new configuration, different actions with equal name will be appended '_new'; events, that are already defined, will be appended the new list of actions. CAN configuration will not be overwritten!

Figure 9: Options when opening a CSV file

When adding a new configuration, all new data are highlighted in colour. If an action is to be added whose name is already allocated to another action, the process is only continued in the case of non-identical actions. In order to avoid any identification conflicts, the name of the new action will then end with “_new”. All recently added actions are highlighted in colour when listed in the client (see Figure 10).

In case the user intends to add an event he has already defined in an old configuration, the system compares the action lists of both events. As long as they are identical, the old entry is kept. However, as soon as the action lists differ, the event’s list of the new configuration is attached to the action list of the existing event. In the respective action lists displayed in the client, all actions belonging to the recently added configuration are highlighted in colour.

When adding a new configuration to an active one, the configuration of the CAN channels remains unchanged. Having successfully completed the adding process, it may be necessary to solve conflicts prior to using this configuration.


SimBox Client 1.0.0 C:/Examples/modify CAN.csv

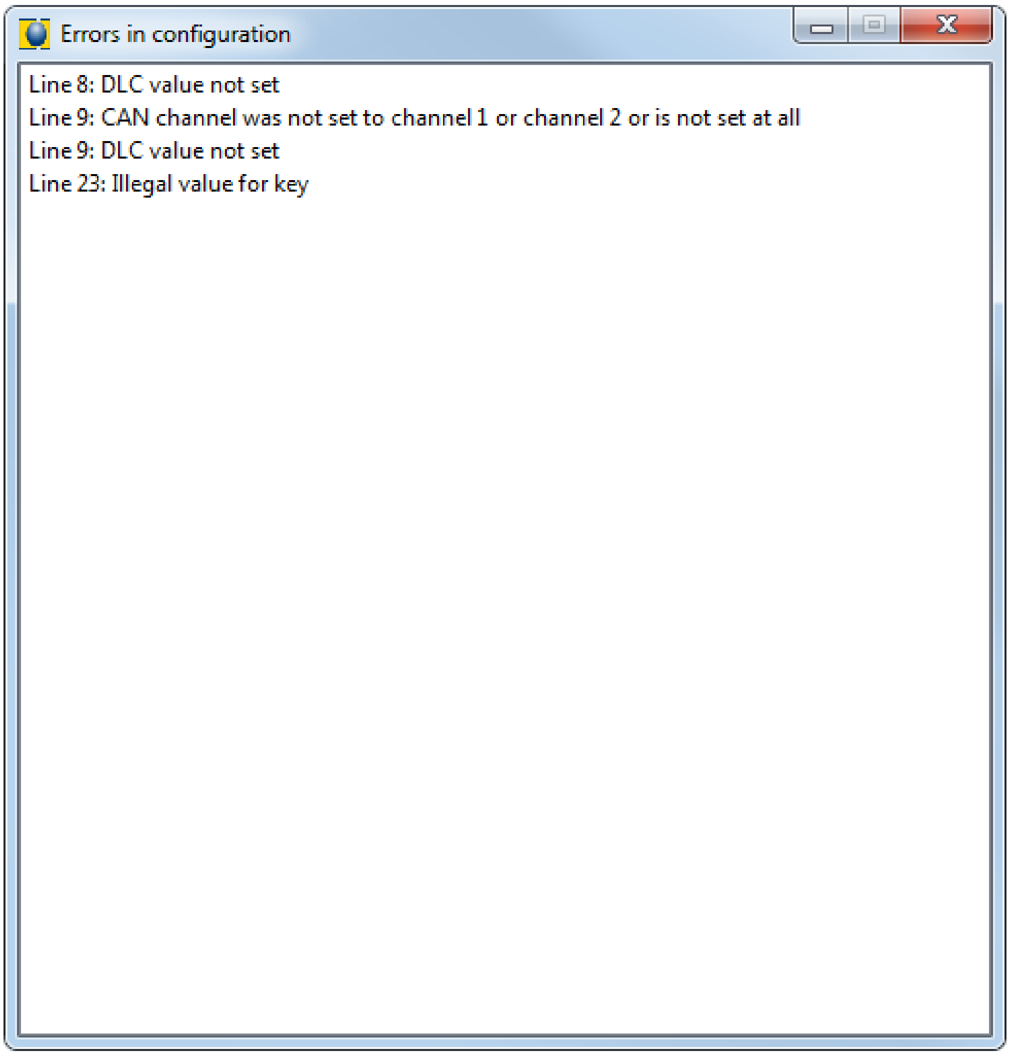
Actions (16)
clearF2
copySignal100
copySignal100_new
modifyCan100
sendCAN2
sendCan100
setCAN2
setCan100
setTimer100
setTimer2
show100
startTimer100
startTimer2
stopTimer2
textF2
textHello

Events (8)
- Initial event
  - setCan100
  - setTimer100
  - setCan100
  - setTimer100
  - textHello
  - setCAN2
  - setTimer2
- Key 'A' pressed
  - sendCan100
- Key 'B' pressed
  - modifyCan100
- Key 'C' pressed
  - startTimer100
- Key 'F1' pressed
  - textF2
  - startTimer2
  - sendCAN2
- Key 'F1' released
  - clearF2
  - stopTimer2
- Timer 'Timer100'
  - modifyCan100
  - copySignal100
  - sendCan100
  - show100
  - modifyCan100
  - copySignal100_new
  - sendCan100
  - show100
- Timer 'Timer2'
  - sendCAN2

New Position: 1
CSV Seperators
Field ;
Action :
COM10

Figure 10: Client after adding a new configuration

Having opened a configuration, the client shows any recognized data sets in both lists, i. e. actions and events. As soon as the system detects errors within the file, a separate window appears listing them accordingly (see Figure 11). Information on line numbers (“Line”) refers to the lines of the CSV file.


Errors in configuration

```
Line 8: DLC value not set
Line 9: CAN channel was not set to channel 1 or channel 2 or is not set at all
Line 9: DLC value not set
Line 23: Illegal value for key
```

Figure 11: Error list after opening a configuration

Values, the client could not recognize, are output as standard values. Please consider that a faulty configuration may lead to an unexpected behaviour of the SimBox.

3.4 Sending a Configuration

Having selected the COM interface (see Figure 8, number 10), the configuration is sent to the SimBox by pressing the key “Transfer” (see number 5). The configuration is prepared for storage within the SimBox and written into its flash memory by the client.

The SimBox needs to be put into update mode before starting the flashing process, e. g. by pressing at the same time the key “S” and the arrow key “Down” located in the right control section (see Figure 12 and also chapter 9).

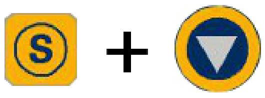


Figure 12: Keys to set update mode

It is not possible to send an empty configuration to the SimBox – if necessary, the client deactivates the key “Transfer”. The client creates different temporary files when preparing data accordingly. These files are automatically stored in the same folder by using the name of the active configuration with changed extensions (.dat, .hex). If there is no active file, all files are stored to the programme directory.

Please note:

Having started the transfer process, the current status of the configuration is converted to an interim format; this "snapshot" is sent to the SimBox. It is possible to edit the configuration in the client, however, sending again an identical configuration, i. e. like before, is excluded. Please save your configuration before starting the transfer process and avoid any operating steps within the client during transfers.

3.5 Firmware Update

The client's command "Update firmware" helps to install a new firmware on the SimBox (see Figure 8, number 11). Having pressed this key, a file browser appears with files in hex-format. Afterwards, the user is asked to put the SimBox into update mode with the help of the key "S" and the arrow key "Down" (see Figure 12).

Chapter 9 offers additional information regarding the activation of the update mode.

Please note:

It is important that the firmware is sent to the SimBox without any errors. Please avoid operating steps within the client during data transfers and do not start additional programmes which may extremely overload your computer.

3.6 Saving a Configuration

In order to save a configuration, the command "Save CSV" (see Figure 8, number 13) is executed which opens a dialogue to select a file. In case there is already a file having the selected name, the user is asked whether he wants to overwrite this file. Saving an empty configuration is not possible, so that the client probably deactivates the above mentioned key.

3.7 Deleting all Elements

Pressing "Clear configuration" deletes the entire configuration from the client (see Figure 8, number 7).

3.8 Checking a Configuration

The command "Check configuration" offers the possibility to check an active configuration for any errors (see Figure 8). The system scans general settings such as CAN-channel data, non-ambiguous names for actions and events etc. This check option is supposed to support the user when creating a configuration, i. e. it does not contain any claim to completeness. It is particularly not able to revise the implementation of a desired functionality but only its formal characteristics. A configuration check, which was completed without any error messages, does not guarantee a correct execution of the respective configuration on the SimBox.

3.9 Setting Separators in CSV Files

With regard to the CSV files used to save configurations, the SimBox client applies two different separators: a field separator and a separator for action lists. If necessary, the user may change these two separators by entering the chosen characters in "Field" and "Action" (see Figure 8, number 8). However, only experts should have the right to set these characters as incorrect entries adversely affect the readability and writability of CSV files. Separators may not be used for other purposes within the respective configuration.

The field separator corresponds to the field separator of the CSV format. Various applications, which are able to edit CSV files, handle it in different ways depending on regional settings. Many of them use a comma; however, European ones may also prefer a semicolon.

The action separator serves to arrange single actions clearly within an action list which is allocated to an event. In this regard, a colon is used as standard character.

3.10 Closing the Programme

The programme may be closed with the command "Exit" available in the graphical interface (see Figure 8, number 14).

Should the client contain a configuration with unsaved changes, the user is asked whether he wants to save accordingly before the programme is closed.

4 Configuration

The SimBox behaviour is event-driven. An event may be, for example, a keypress carried out by the user. This event causes the SimBox to process a sequence of actions which have been configured for this event. Sending a CAN message and displaying the respective information on the screen are examples for such actions. The following chapters describe events and actions.

Apart from manual events resulting from keypresses by the user, the SimBox also offers configurable timers which, for example, send CAN messages in predefined intervals without any operating steps on the SimBox.

The actions, which may be configured for the SimBox, mostly represent individual steps of a process whose scope of application increases by different ways of combination. CAN messages, for example, may be divided into two actions: one to define the content and one to send the message. Thus, the message content needs to be defined only once and, prior to every sending action, a further action may be used to increase a sequence counter in this message.

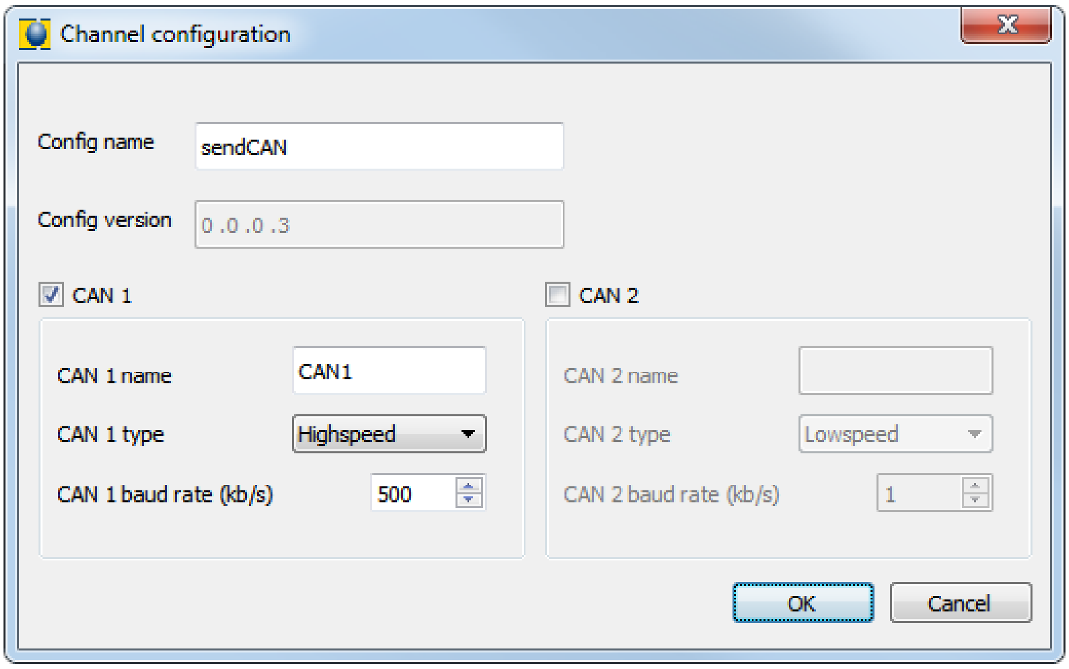
When creating a new configuration, it is recommended to identify all objects necessary for this specific configuration – this should be the first step followed by starting the SimBox and entering one's data. The process of entering configuration steps begins with the creation of individual actions. Afterwards, a single action and even several ones may be marked, defined as action list and allocated to an event, e. g. a keypress. Chapter 7 offers some configuration examples and explains how to create them.

It is also possible to edit existing actions and events. The SimBox client is able to save and read the configuration as CSV file. If necessary, these files may be changed by the user or created by automatic tools.

The SimBox is able to store and process more than 3000 configuration steps, i. e. lines in the CSV file. This offers the possibility to send more than 1500 different CAN messages provided that each message is configured with the help of two actions: one to define the message's content and one to send it. Other actions, such as displaying a text, also occupy one place each in the flash memory of the SimBox.

4.1 General Configuration Settings

The settings of the CAN channels and the name of the configuration are defined with the help of the field "CAN channel configuration" (see Figure 8, number 5). The field "Config name" (see Figure 13) is reserved to enter the configuration's name which can be recalled on the SimBox.


Channel configuration

Config name: sendCAN

Config version: 0 .0 .0 .3

[✓] CAN 1

CAN 1 name: CAN1

CAN 1 type: Highspeed

CAN 1 baud rate (kb/s): 500

[ ] CAN 2

CAN 2 name:

CAN 2 type: Lowspeed

CAN 2 baud rate (kb/s): 1

Figure 13: Configuration settings

The check boxes, which are available for both CAN channels, provide activating and deactivating options for each channel. Every channel may receive its own name and settings as regards transceiver type and baud rate. Similar to the configuration's name, the channel name only serves information purposes and can be recalled on the SimBox.

The field “Config version” contains a version number which is also stored in the CSV-configuration file. With the help of this number, the SimBox client is able to identify file format and supported features as soon as it reads a file. The client uses a well-defined version specification when writing the configuration.

5 Actions

An action is initiated by a specific event, e. g. pressing an element of the keyboard.

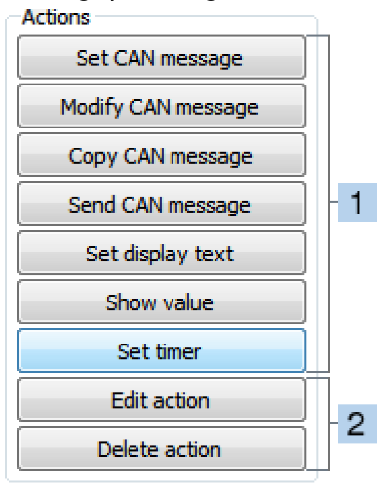


Figure 14: Administration of action lists

(1) Create new actions
(2) Edit and delete existing actions

The following actions may be created:

- “Set CAN message”
- “Modify CAN message”
- “Copy CAN message”
- “Send CAN message”
- “Set display text”
- “Show value” (on display)
- “Set timer”

Having pressed one of the fields mentioned above, a dialogue window appears to create an action of the selected type. The bottom of this window contains an input field which is to be used to allocate a non-ambiguous name to this action (see Figure 15, number 1). As soon as an action has been created, its name is added to the action list of the main window and, moreover, it is also used when linking this action with an event (keypress, timer).

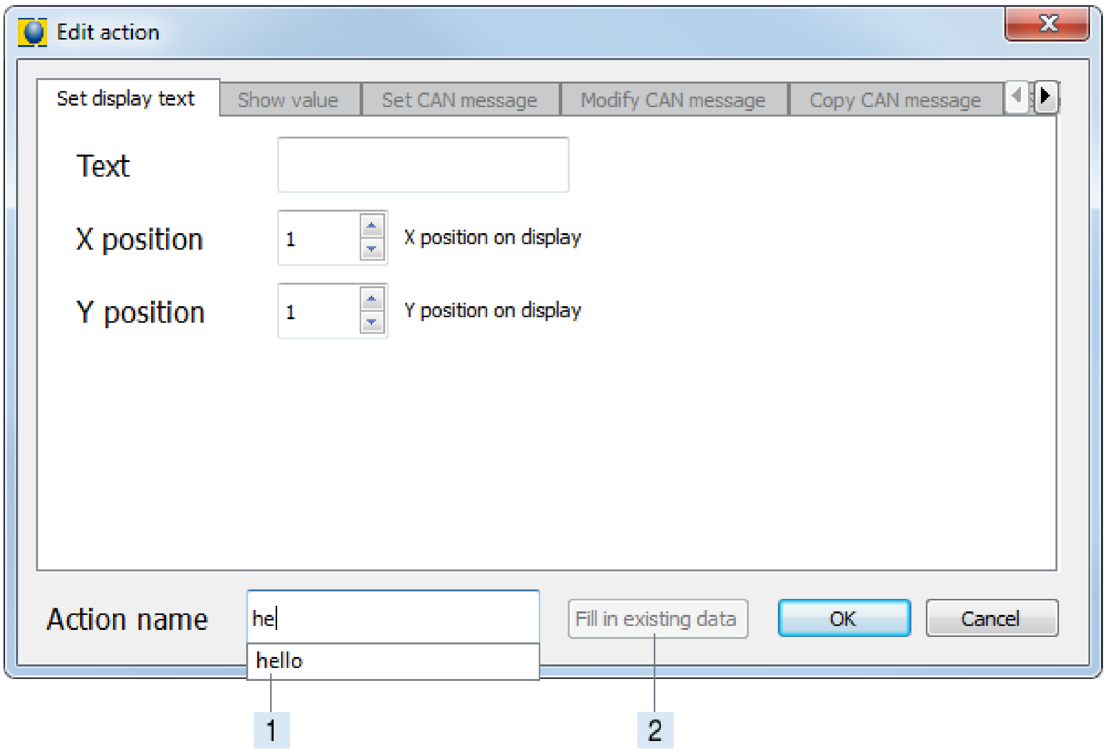


Figure 15: Defining an action name

(1) Name of respective action
(2) Import files of an existing action

It is not possible to press the button “OK” unless an action name has been defined. The client informs the user in case the entered name already exists for another action. The user may then define a different name or cancel the whole input procedure.

Moreover, the user has the possibility to facilitate an input procedure by importing data of an existing action. To do so, simply enter the respective name and press the button “Fill in existing data” (see Figure 15, number 2). This field is inactive as long as an existing name has not yet been entered or if the existing action type does not correspond to the one currently created.

5.1 Setting a Text

The command “Set display text” (see Figure 14) helps to display any text on the screen. Once executed, a window appears as shown in Figure 15.

Having defined the text, which is to be set, its X- and Y-positions are selected on the display. The respective action is added to the action list with the help of the button “OK”.

The display has 4 lines with 20 characters each, i. e. the Y-value may range from 1 to 4 and the X-value from 1 to 20. Value “1” for both, X- and Y-position, represents the left, upper edge of the display.

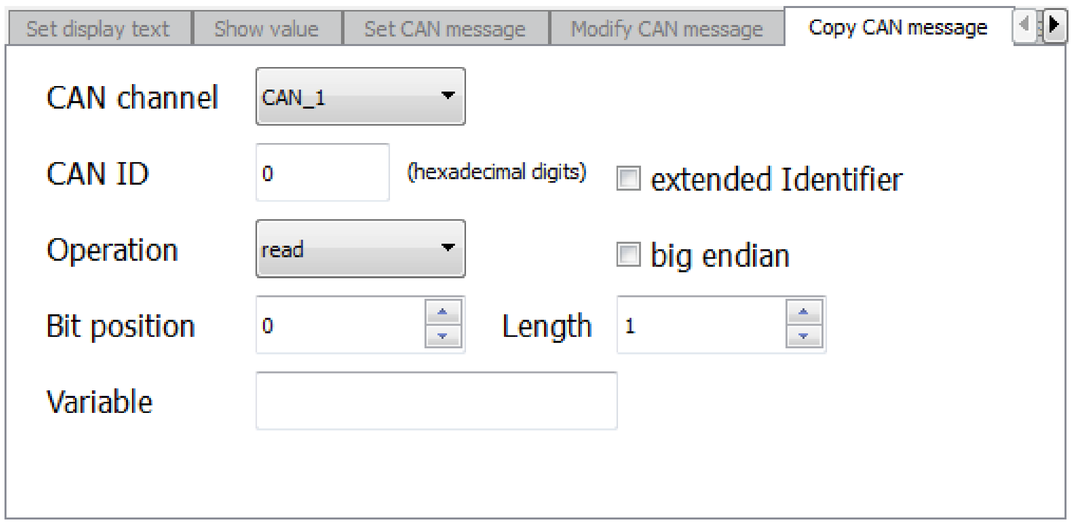
5.2 Showing Values on Display

Two steps are required to show a specific signal of a CAN message on the display. First, press “Copy CAN message” (see Figure 14) which saves the signal’s value as variable within the SimBox. Secondly, the variable’s value may be shown on the display using “Show Value”. The following paragraphs describe how to configure both actions required.

5.2.1 Copying a CAN Signal

A CAN signal is defined by the CAN message, which is used to transfer it, and by its bit positions within the data area of this CAN message. This information is to be entered with the help of the input mask shown in Figure 16.

To create an action, first select the tab “Copy CAN message” (see Figure 14).



CAN channel: CAN_1
CAN ID: 0 (hexadecimal digits)
☐ extended Identifier
Operation: read
☐ big endian
Bit position: 0
Length: 1
Variable:

Figure 16: Copy CAN signal

In order to save a CAN signal as variable, select value “read” as “Operation”, specify the area for this message through “Bit position” and “Length” and define a name for variable and action.

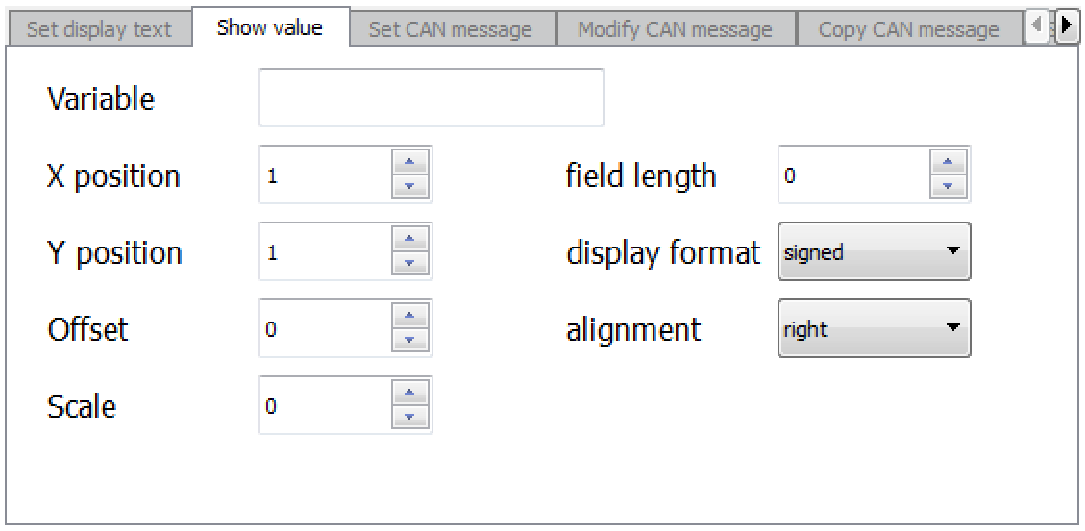
Choosing “read” as operating type, the value of selected bits is saved in the variable – choosing “write” instead, the variable’s content is saved in the CAN-signal. The value is copied without considering any leading signs (“unsigned”).

The order of bytes may be defined by the setting “big endian” in the case of signals which are saved across several bytes of the CAN message. A marked check box leads to the setting “big endian” and “Motorola”; a blank check box to “little endian” and “Intel”.

The LSB of the CAN signal is to be indicated as bit position. Its position depends on the setting made for the box “big endian”. Should the signals be defined as Vector-DBC file, information may be added using this file.

5.2.2 Showing a Value

The tab “Show value” is used to display the respective value (see Figure 14).



Variable:
X position: 1
field length: 0
Y position: 1
display format: signed
Offset: 0
alignment: right
Scale: 0

Figure 17: Show value

The field “Variable” serves to enter the variable’s name of the recently created action. “X position” and “Y position” specify the value’s position on the display (see chapter 5.1); “field length” defines how many positions shall be reserved for this value.

The value’s format may be changed with the field “display format”. Possible options are “signed” (with leading signs), “unsigned” (without leading signs) and “hex” (hexadecimal format).

The value’s alignment is defined with the help of the field “alignment” offering the options “left” and “right”. The value of the message is convertible through “Offset” and “Scale.

The following formular applies to positive values of “Scale”:

$$displayed\ \text{value} = (\text{message value} - \text{Offset}) \cdot \text{Scale}$$

As regards negative values of “Scale”, the following formular is valid:

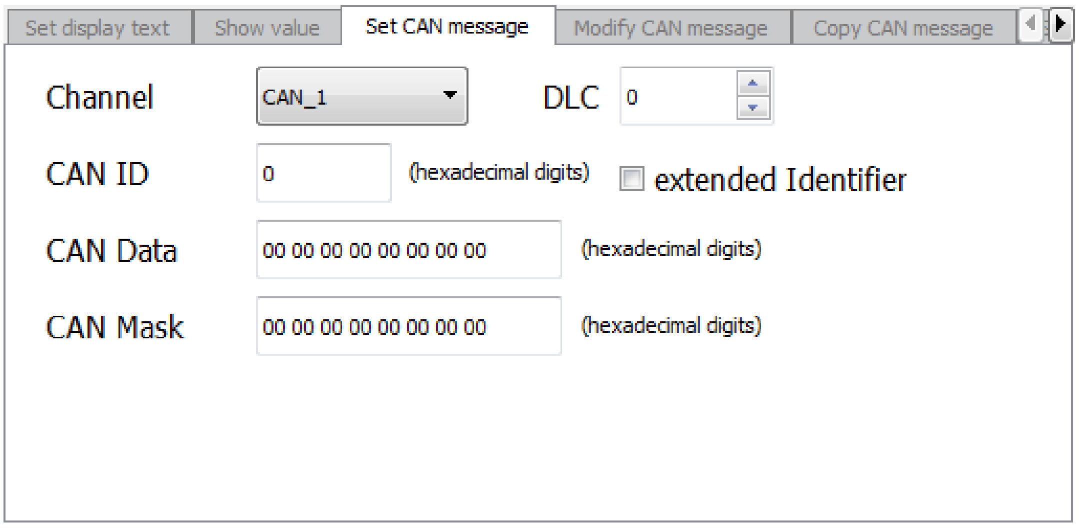
$$displayed\ \text{value} = (\text{message value} - \text{Offset})/(-\text{Scale})$$

Please note:

In case the number of characters, needed to display the respective value, exceeds the possible field length, the additional characters will be written beyond the end of the field, i. e. shown on the display.

5.3 Creating a Message

The SimBox is only able to send a CAN message if the respective content has been defined. In order to create such an action, select the command “Set CAN message” (see Figure 14). The following window appears:



Channel: CAN_1
DLC: 0
CAN ID: 0 (hexadecimal digits)
extended Identifier: ☐
CAN Data: 00 00 00 00 00 00 00 00 (hexadecimal digits)
CAN Mask: 00 00 00 00 00 00 00 00 (hexadecimal digits)

Figure 18: Set CAN message

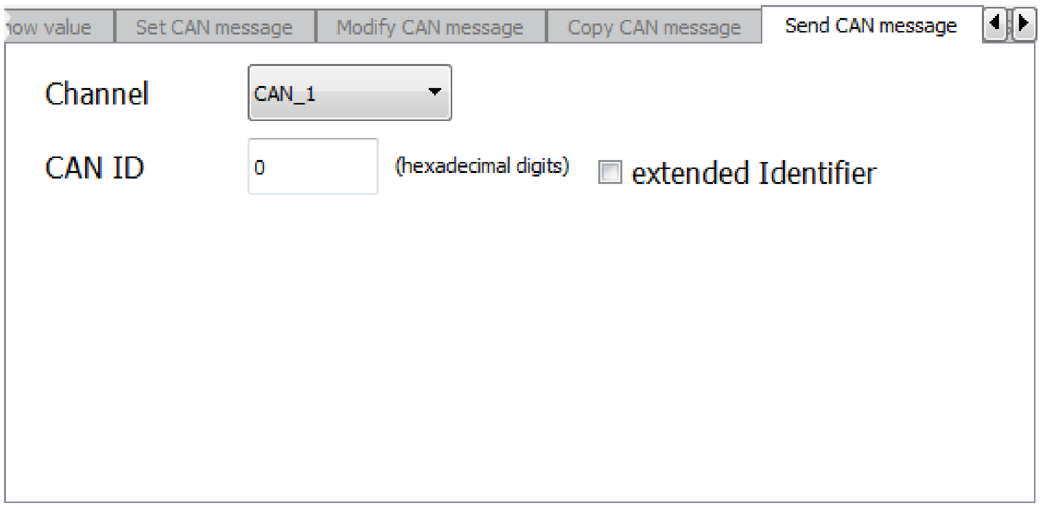
With the field “Channel”, the user selects the channel on which the message shall be sent. The payload length is set in bytes using the field “DLC”.

“CAN ID” serves to define the identifier (in hexadecimal format). The check box “extended Identifier” is marked in case the message shall be sent with an extended identifier having a length of 29 bits – otherwise, the system uses a standard identifier with 11 bits.

Data content is entered in “CAN Data” and hexadecimal format. As regards allocation, data content is linked with the help of the field “CAN Mask”. Only those bits of the CAN message are changed which have been marked with “1” in this mask – all others remain unchanged.

5.4 Sending a Message

Select the command “Send CAN message” if a message is to be sent (see Figure 14).



Channel: CAN_1
CAN ID: 0 (hexadecimal digits) ☐ extended Identifier

Figure 19: Send CAN message

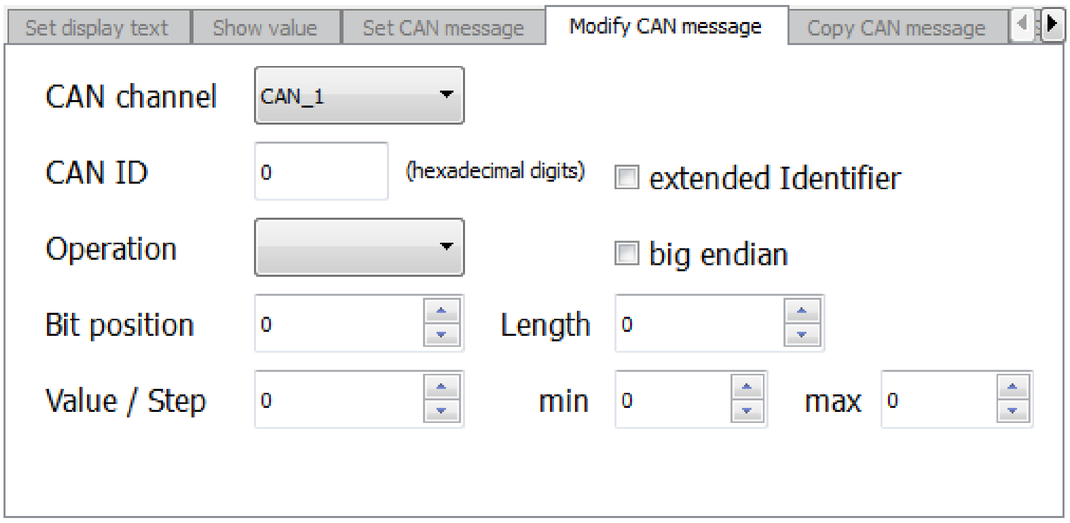
The desired channel is selected with “Channel”; the respective message with “CAN ID”. Afterwards, a name is to be chosen for this action and the dialogue may be closed with “OK”. This action is then added to the window “Action”.

Prior to sending a CAN message, its content needs to be defined – otherwise, unexpected results may occur (see chapter 5.3). The SimBox uses CAN ID and channel number to identify a message.

As regards cyclic messages, please use a timer event (see chapter 5.6 and 6.3).

5.5 Changing a Signal of a CAN Message Dynamically

In order to change a message during the SimBox’s operation, the command “Modify CAN message” is to be executed (see Figure 14). The window below appears and the action’s parameters may be changed:



CAN channel: CAN_1
CAN ID: 0 (hexadecimal digits) ☐ extended Identifier
Operation: ☐ big endian
Bit position: 0 Length: 0
Value / Step: 0 min: 0 max: 0

Figure 20: Modify CAN message

The desired CAN channel is selected with “CAN channel”; the identifier of the respective message is entered in “CAN ID” using hexadecimal notation. The field “Operation” serves to select the type of change. With the help of “Bit position”, the user may define the LSB number (least-significant bit) of the bits which are to be changed. The value chosen in “Length” stipulates how many bit positions shall be changed. By default, the system uses the byte order “Little Endian” (“Intel”); however, this may be changed by marking “Big Endian” (“Motorola”).

The field “Value / Step” helps to define which value is to be allocated and which intervals are used to increment this value. With the options “min” and “max”, the user is able to set the upper and lower limit of all operations which may restrict value ranges. Please set both options to “0” if limiting is not desired.

The following paragraphs describe potential settings for “Operation”.

5.5.1 Setting a Value

To set a message’s content, the user has to select “Set” within “Operation”. The field “Value” is reserved to define the respective value.

5.5.2 Increasing/Decreasing a Value (Add/Subtract)

The options “Add” and “Subtract” influence the value of a CAN signal by increasing and decreasing respectively. The limits set with “min” and “max” are not considered. If any overflow occurs, the system ignores the highest bits (bit positions beyond defined length).

5.5.3 Increasing/Decreasing a Value with Limitation (AddLimit/SubtractLimit and Increment/Decrement)

A signal value may be increased or decreased with the options “ADDLIMIT” and “SUBTRACTLIMIT” respectively; both can be found within “Operation”. Having reached the minimum or maximum limit, the system does not continue to change the respective value.

“INCREMENT” and “DECREMENT” are synonyms for “ADDLIMIT” and “SUBTRACTLIMIT”.

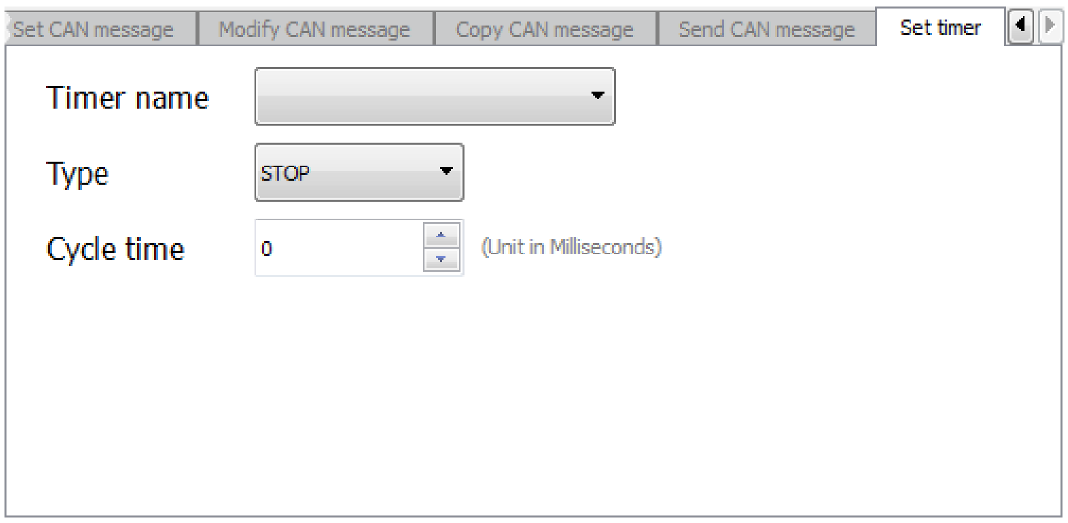
5.5.4 Increasing/Decreasing a Value with Overflow (AddWrap/SubtractWrap)

By selecting “ADDWRAP” or “SUBTRACTWRAP” within “Operation”, a signal value may be increased and decreased respectively. Having reached the minimum or maximum limit, any overflowing data is transferred to the other end of the value range (“Wraparound”).

5.6 Timer

Select the command “Set timer” if a timer action is to be created (see Figure 14).

The field “Timer name” serves to choose the respective name. The configuration needs to know the timer’s name already, i. e. there has to be a timer event with the name of the desired timer. It is not possible to create a timer action without selecting the respective timer within the list “Timer name”.



Timer name
Type: STOP
Cycle time: 0 (Unit in Milliseconds)

Figure 21: Change timer

The cycle time is set in milliseconds using the field ‚Cycle time‘. Please note that all SimBox timers operate according to a certain pattern (10ms).

The type of timer action is selected using the field “Type”. Previous to the first use of a timer, the actions “Set”, “Set Cyclic” or “Set Single” need to be called up in order to set cycle time and correct operating mode.

5.6.1 Start

This command starts the timer and, besides, resets the time expired of the current cycle.

5.6.2 Stop

This command stops the respective timer; timer events are no more carried out for this timer.

5.6.3 Reset

This command resets the time expired of the timer; the timer’s cycle starts anew.

If a timer has been started and is reset before expiration of its cycle time, the latter will start anew. A timer event is not caused by this reset. This action may be used to implement watchdog timer or follow-up time.

5.6.4 Set

Please refer to “Set Cyclic”.

5.6.5 Set Cyclic

This command initializes the cycle time of the respective timer and configures it as being cyclic.

The timer needs to be started with a separate starting command. As soon as the timer expires, a timer event is created and the cycle automatically starts anew. To cancel this timer event, the command “Stop” has to be executed.

5.6.6 Set Single

This command initializes the cycle time of the respective timer and configures it as being single (“Single Shot”).

The timer needs to be started with a separate starting command. As soon as the timer expires, a single event is created.

5.7 Changing an Action

If an action is to be changed, it needs to be marked in the action list with a mouse click. Having pressed “Edit action” (see Figure 14, number 2), the respective window appears which shows the stored parameters of this action – these may now be changed.

Please be careful with changing the name of an action. In case the user enters an existing name, the edited action will be saved with this name and, moreover, the existing action will be replaced. This is why the user receives a warning message as soon as he wants to save the edited action. “Yes” confirms his entry and overwrites the existing action; “No” reopens the dialogue to change the edited action; “Abort” cancels the changing process without saving any entries.

5.8 Deleting an Action

If an action is to be deleted, it needs to be marked before executing the command “Delete action” (see Figure 14).

5.9 Highlighting Actions in Event List

It is possible to highlight all events which are associated with an action. A right mouse click on any action opens a context menu; the item “Highlight attached events” marks all associated events (see Figure 22).

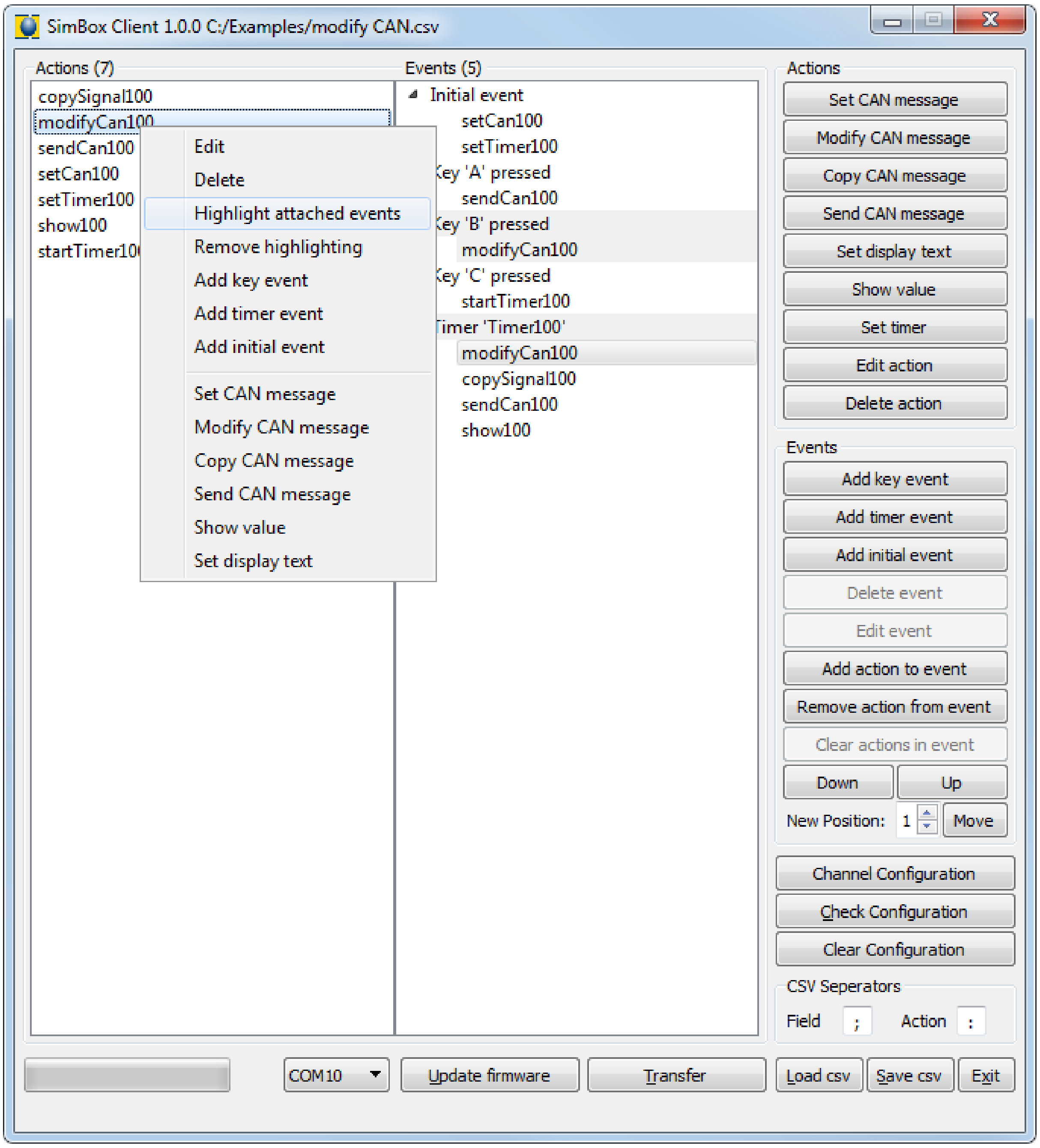


Figure 22: Context menu of actions

Afterwards, all events are marked which are associated with this action. Pressing “Remove high-lighting” resets all markings.

Please note:

Figure 22 already illustrates the status after having selected the marking function.

6 Events

The SimBox offers different kinds of events for configuration purposes:

initial events, key events and various timer events carried out within the SimBox. An event-action list is allocated to every event; it contains all actions which need to be processed as soon as the respective event occurs. The order of its actions defines the order how they are processed, i. e. the action mentioned first is carried out right at the beginning and, accordingly, the last one at the end. In order to create an event, an action needs to be selected by clicking on the respective name in the action list; keeping the Ctrl-key pressed allows to select several actions. This procedure, i. e. clicking the desired action(s), automatically allocates them to the event which is currently being created.

In case the user has already added a specific event in his configuration, it is not allowed to create it a second time. However, he may change the existing event by adding/deleting actions to/from its action list; a further alternative is to delete the existing one and create a new event.

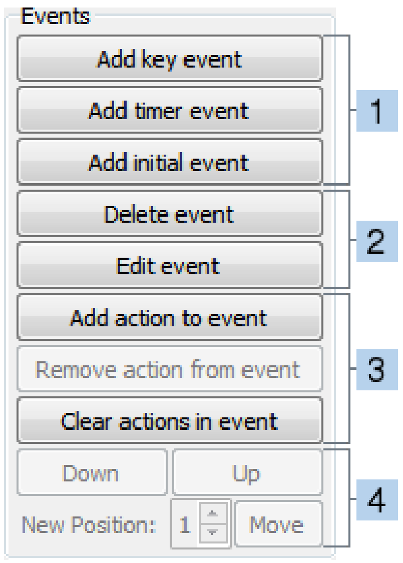


Figure 23: Option list for events

(1) Create events
(2) Edit events
(3) Add and delete actions
(4) Edit order within action list

6.1 Initial Event

Pressing “Add initial event” (see Figure 23, number 1) defines an initial event whose actions are carried out once as soon as the SimBox is started. This kind of event is an option to initialize data of CAN messages or timers.

An initial event may only be defined once – pressing this button a second time would only replace the existing event, however, a further event cannot be added. In comparison to all other events, an existing initial event may also be extended by adding further actions (see chapter 6.6).

6.2 Key Event

Every soft key of the SimBox may cause an event. First, one or several actions are selected and second, the button “Add key event” is pressed (see Figure 23, number 1). The following dialogue appears:

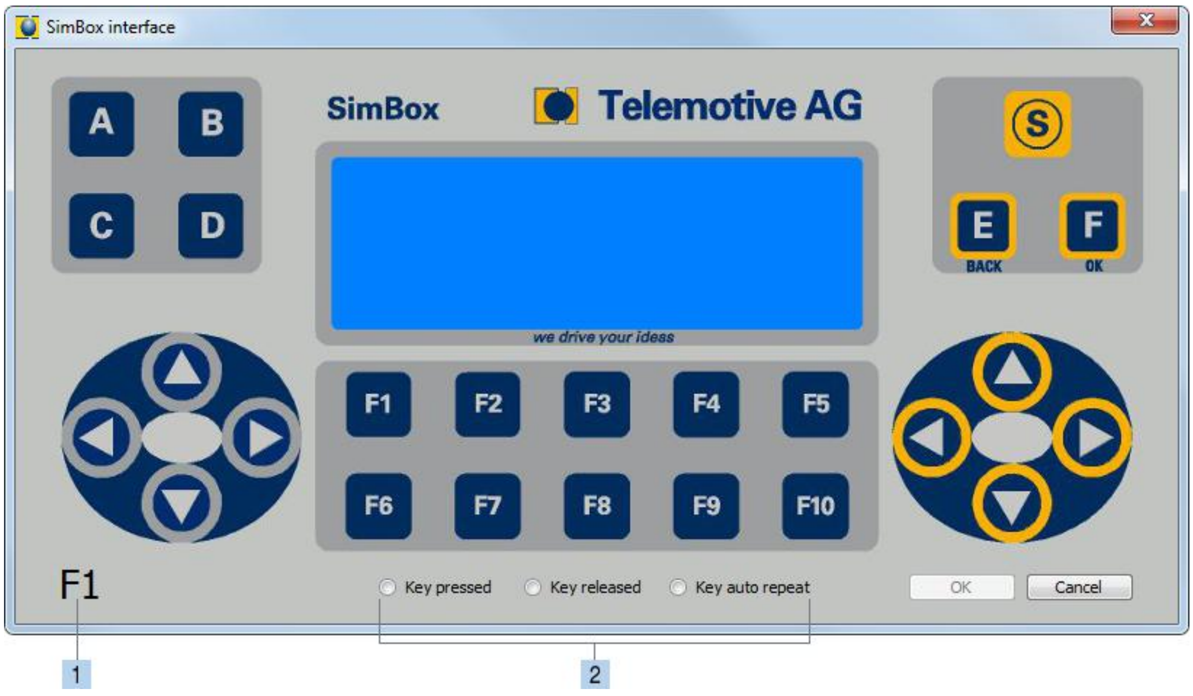


Figure 24: Defining a key event

(1) Selected key
(2) Type of key operation

A key event is defined with the help of the key's name and its type of operation (press, release, autorepeat).

Having clicked a key, its name is shown in the left, lower edge of the selection dialogue (see Figure 24, number 1). The type of operation is set with the options offered at the bottom of the dialogue (see Figure 24, number 2):

- "Key pressed": Event occurs as soon as the respective key is pressed.
- "Key released": Event occurs as soon as the respective key is released.
- "Auto repeat": Event occurs repeatedly as soon as the key is pressed for a longer time.

Please note:

The key event "Auto repeat" only occurs with a certain delay after having pressed the respective key. In case an action is to be carried out immediately when pressing a key and, moreover, in combination with an autorepeat, this action needs to be added to the action lists of the event types "Key pressed" and "Auto repeat" respectively.

6.3 Timer

A timer event is created by selecting the desired actions in the action list and pressing "Add timer event" afterwards (see Figure 23, number 1); a dialogue appears to enter the name of the respective timer. It may get a new name, however, the system offers the already existing timer names as examples.

Having started the timer, its event is triggered and any allocated actions are processed as soon as the respective cycle time expires.

6.4 Changing an Event

With the help of the button "Edit event" (see Figure 23, number 2), events may be changed to a certain extent. In the case of key events, the key's name and its type of operation are editable; as regards timer events, the timer's name may be changed.

6.5 Deleting an Event

In order to delete an event, it is selected and the button “Delete event” is pressed (see Figure 23, number 2).

6.6 Allocating an Action to an Existing Event

There are several options to allocate an action to an event which has already been configured.

1. Allocate one or several actions of an action list:
   The user may mark one or several actions of an action list and add them to an event either with Drag & Drop or with the button “Add action to event” (see Figure 23, number 3). With regard to Drag & Drop, all previously marked actions are added at the end of an action list as soon as he releases the mouse button above its event or marks its event. On the other hand, if he releases the marked actions above an action of an event’s action list or if he marks such an action, these actions will be inserted exactly at this location.
   As regards the allocation of actions to an event by using the button “Add action to event”, the desired target needs to be marked in the right window named “Events”. Adding actions to an event’s action list corresponds to the procedure described above for Drag & Drop.
2. Add an action of an event’s action list to another event’s action list:
   The user may shift the entire action list of an event to the action list of any other event. However, this is only possible with the help of Drag & Drop – the user has to mark an event and release it above any other event or within the action list of any other event. This operation needs to be confirmed in order to avoid an inadvertent shifting of action lists.

6.7 Removing an Action from an Event

An action is removed from an event by marking the respective action and selecting “Remove action from event” (see Figure 23, number 3).

6.8 Releasing an Event from all Actions

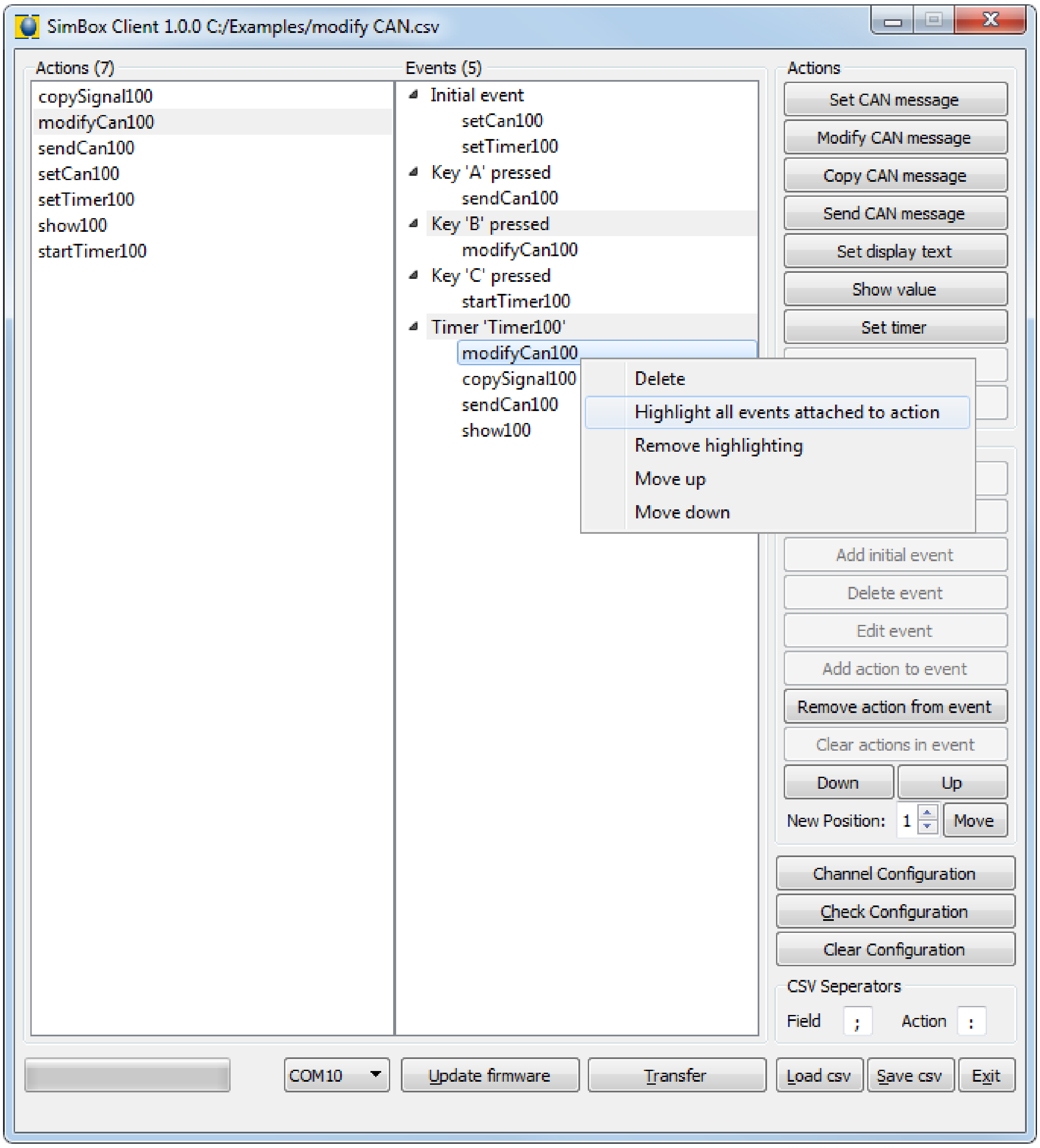
It is possible to delete the entire action list of an event. For this purpose, the respective event is selected and the button “Clear actions in event” (see Figure 23, number 3) needs to be pressed.

6.9 Moving an Action within an Action List

The processing sequence of actions depends on the individual positions within an action list. In case the user wants a specific action of the list to be processed before or after another action, he needs to mark the respective action to be able to change its position. It is shifted by one position upwards or downwards when the buttons “Up” or “Down” are pressed once (see Figure 23, number 4). In case this action is shifted beyond the end of the action list, it is moved to the list’s other end. Locating this action to a specific position may be carried out by marking it, selecting the desired position and choosing “Move” to shift it accordingly.

The order within action lists may also be changed with Drag & Drop – to do so, the respective action is marked within its action list and moved to the desired position.

6.10 Highlighting Actions of an Event List


SimBox Client 1.0.0 C:/Examples/modify CAN.csv

Actions (7)
- copySignal100
- modifyCan100
- sendCan100
- setCan100
- setTimer100
- show100
- startTimer100

Events (5)
- Initial event
  - setCan100
  - setTimer100
- Key 'A' pressed
  - sendCan100
- Key 'B' pressed
  - modifyCan100
- Key 'C' pressed
  - startTimer100
- Timer 'Timer100'
  - modifyCan100
  - copySignal100
  - sendCan100
  - show100

Figure 25: Highlighting action in event list

Being within the action list of an event, it is possible to highlight one of its associated actions. This operation helps to find easily all other events, which use this action, as well as the respective action itself as they are all listed in the left actions window. A context menu is opened with a right mouse click on any action within the action list of an event. The Client marks the respective action and all associated events as soon as “Highlight all events attached to this action” has been selected (see Figure 25).

Please note:

Figure 25 already shows the status after highlighting.

7 Tutorial: Creating a Configuration

Depending on the chosen configuration, the SimBox reacts to different events by executing their related actions. An event may be, for example, a keypress; displaying a text or sending a CAN message represent possible actions which are executed afterwards.

As regards the creation of a configuration, it is recommended to define all necessary actions before starting to link them with initiating events.

When creating a new configuration, the first step should be to set the name and parameters for both CAN channels of the SimBox with the help of the button "CAN channel configuration" (see Figure 8).

7.1 Configuration Example "Hallo Welt"

This chapter is to exemplify the configuration steps necessary to display the text "Hallo Welt" ("Hello World") whenever the SimBox is switched on.

The action, which is to be executed in this case, is "Set display text" (see Figure 14). Starting the SimBox firmware represents the initiating event. This may be set in the client by using the button "Add initial event" (see Figure 23).

Having opened the client, a name is to be defined for the configuration. This is entered and confirmed with "OK" in the dialogue which appears after pressing "CAN channel configuration" (see Figure 8).

Now, the action may be defined. "Set display text" (see Figure 14) opens a dialogue which is used to enter the desired text (see Figure 15), i. e. "Hallo Welt" is entered in the field "Text". "X pos" and "Y pos" are both filled with the value "1" corresponding to the left upper corner of the display; "hello" is added to "Action name" – afterwards, the dialogue is closed by pressing "OK".

This action is then linked with the initating event. For this purpose, the entry "hello" needs to be marked in the left window "actions" prior to selecting the button "Add initial event" (see Figure 23).

The configuration may now be saved as CSV file and transferred to the SimBox. All necessary steps are described in the following chapters:

- Saving: chapter 3.6
- Opening configuration in Client: chapter 3.3
- Transferring configuration to SimBox: chapter 3.4

As soon as the SimBox is switched on, the start screens of its firmware are shown for a short time and then the text "Hallo Welt" appears on the display. It might happen that the SimBox switches off after the configuration has been successfully transferred; in this case, the user may switch it on again by pressing the key "S".

7.2 Advanced Configuration Example "Hallo Welt"

The configuration described in the previous chapter shall now be modified in a way that only the text "Hallo" ("Hello") appears upon start of the SimBox. Pressing the key "F1" shall expand this text to "Hallo Welt" ("Hello World"); the key "F2" shall delete the entire text.

This configuration requires three similar actions with different parameters: writing the text "Hallo", adding "Welt" to the previous text and overwriting this area with blanks in order to delete the entire text. The initiating events are the SimBox's start and two subsequent keypresses.

The first step comprises a modification or creation of the necessary actions which were recently set for the configuration "Hallo Welt" in the previous chapter. Having opened the configuration, the action "hello" is marked in the action list and "Edit action" (see Figure 14) is used to modify the recent parameters. The word "Welt" is deleted in the field "Text".

Afterwards, "Set display text" (see Figure 14) helps to create two new actions. As regards the parameters "Text", "X pos", "Y pos" and "Action name" (see Figure 15), the first action is defined with "Welt", "7", "1" and "world", the second with "   " (ten blanks), "1", "1" and "clear". Afterwards, the new actions "world" and "clear" are linked with the keys "F1" and "F2" respectively. For this purpose, the related action name is marked in the action list and "Add key event" is used to open the dialogue for the keys' selection. On the illustration of the SimBox's keyboard, the desired key is selected and “Key pressed” is defined as type of event – “OK” confirms these settings. A new row is added for each one in the event window.

Having transferred this configuration to the SimBox, its display shows at first only the word “Hallo”. Pressing the key “F1” leads to the entire text “Hallo Welt”; “F2” helps to delete it accordingly.

As soon as “F1” is pressed once again, only the word “Welt” appears on the display. There are two alternatives in case the entire text shall be shown:

On the one hand, the action “world” may be modified in a way that the entire text “Hallo Welt” is written on position (1;1) of the display. On the other hand, the two actions “hello” and “world” may be marked at once and linked with the key “F1”. The subsequent entry to the event list specifies two actions for the keypress “F1”: “world” and “hello”.

7.3 Sending a CAN Message

This chapter is to exemplify the configuration steps necessary to send a CAN message after a keypress. The content of the message’s first byte shall be increased by one after a further keypress.

Various steps are necessary to send a CAN message:

- Defining a CAN message:
  Within the SimBox, a storage place needs to be reserved for this CAN message and filled with data – the action “Set CAN message” helps to define accordingly. For every CAN message, it is recommended to link this action once with the initiating event.
- Sending a CAN message:
  As soon as the message has been defined, it may be sent on the CAN bus with the help of the action “Send CAN message”.
- Changing data of a CAN message:
  In order to simulate different operating modes, data fields of the CAN message have to be changed. As regards the exchange of a few, specified values, the action “Set CAN message” – if necessary, in combination with a mask to set particular bits – may be used (see chapter 5.3). As far as the change of values like engine temperature or speed is concerned, it is recommended to select the action “Modify CAN signal” (see chapter 5.5).

This example explains how to control the sending and changing of a signal with the help of keypresses.

Having opened the client and an empty configuration respectively, the button “CAN channel configuration” (see Figure 8) helps to define the parameters of the CAN buses. The configuration of the SimBox is to have the same settings like all other participants of the CAN bus.

Afterwards, the different actions are defined. The CAN message needs to be specified with the ID 0x100 and filled with its original data. With regard to this example, the DLC is set to “2” and the entire data area is initialized with “0”, i. e. “CAN Data” needs to be filled with “00 00” and “CAN Mask” with “FF FF”. These values are entered in the dialogue which is opened with the button “Set CAN message” (see Figure 14).

This message may already be sent after having executed the action “Set CAN message”. Pressing “Send CAN message” (see Figure 14) opens a dialogue to enter the identical CAN ID.

In order to change the first byte, an action is specified with the help of “Modify CAN message” (see Figure 14) which contains the same address information as the CAN message:

Channel: CAN_1; CAN ID: 100; extended Identifier: no; Operation: ADD; Bit position: 0; Length: 8; Value/Step: 1; min: 0; max: 0; ActionName: modifyCan100.

These actions are then linked with the desired events: setCan100 with the initiating event (“Add initial event”), sendCan100 with key “A”, modifyCan100 with key “B” (“Add key event”). As regards the keys, the action shall be executed as soon as the respective key is pressed.

7.4 Timer-controlled Sending and Displaying of Value

To be able to process a lot of use cases, a CAN message may be sent automatically based on a specific cycle time. In order to send the respective message, a timer is defined within the SimBox whose expiration causes a timer event. This event may again lead to different actions, e. g. sending a CAN message.

The following steps are necessary to initiate a timer event:

- Setting a timer event:
  "Add timer event" serves to link actions with a timer event. The configuration is then informed about the respective timer name which may be used to set the timer with the help of actions.
- Defining a timer:
  This comprises the setting of a timer's cycle time. As regards unknown timers, a storage place is reserved in the SimBox and filled with the parameters according to one of the "Set"-data types of the timer action.
- Starting a timer:
  Setting the timer parameters does not automatically start the timer; this requires a timer action of the type "Start". Afterwards, the respective timer may be stopped and restarted, e. g. to establish bus rest of the SimBox.

The example described in the previous chapter shall be expanded by a timer having a cycle time of 500 ms. This timer is linked to an action – the sending of a CAN message.

Prior to allocate actions to this timer, additional actions are created to display the data byte which is modified in the respective CAN message. Values of a CAN message can only be displayed as soon as they have been copied to a variable. The variable value is then shown after having specified format options.

At first, the action "Copy CAN message" (see Figure 14) is defined to copy a signal of the CAN message to a variable. Specifying a variable name automatically reserves an appropriate storage place. As soon as the button "Copy CAN message" (see Figure 14) has been pressed, the following values are entered in the respective fields; the information of the CAN signal corresponds to that of the action "Modify CAN message":

CAN channel: CAN_1; CAN ID: 100; Operation: read; Bit position: 0; Length: 8; Variable: variable100; Action name: copySignal100.

"Show value" (see Figure 14) serves to display the value by creating the respective action with the following parameters:

Variable: variable100; X position: 1; Y position: 2; Offset: 0; Scale: 0; field length: 3; display format: unsigned; alignment: right; Action name: show100.

In the previous example, the CAN message was sent after a keypress – the current example explains how the send a message automatically with the help of a timer in combination with the increase and display of the message's first byte. For this purpose, the actions modifyCan100, sendCan100, copySignal100 and show100 are marked at once and allocated to the respective timer by using "Add timer event" (see Figure 23).

Marking several entries in the action list requires to keep the key "Strg" pressed while selecting additional entries. They are allocated to the timer by choosing the timer's name – in this case "Timer100".

The next step is to create the actions which are necessary to control the timer. "Set timer" (see Figure 14) opens a dialogue to enter the timer's parameters. "Timer100" needs to be selected as timer name; the field "Type" is set to Set_Cyclic and "Cycle time" to 500 ms. The text "setTimer100" is added as action name.

A second action to start the timer is created in a similar way – the respective fields contain the following information:

Timer name: Timer100; Type: Start; Action name: startTimer100.

Prior to use the timer, its cycle time needs to be initialized – for this purpose, the most convenient way is the initiating event. Being in the main window of the client, the action "setTimer100" (located in the left field "Actions") is dragged to the initiating event "Initial event" (located in the right field "Events"). Now, the initiating event has been allocated to two actions: "setCan100" and "setTimer100".

The only missing step is the timer's start. An action with the name "startTimer100" has already been created. Therefore, it only needs to be linked to a key – "C" is chosen in this example.

Having pressed the C-key, the SimBox display shows how the value of the message's first byte is changed. After every cycle, the CAN message with the respective value is released for sending.

7.5 Timer-controlled Sending with Keypress

This chapter is to exemplify the configuration steps necessary to send a CAN message after a defined cycle time as long as a certain key is pressed.

The following examples describe how to send a single CAN message each. For this purpose, SET_CAN and SEND_CAN need to be defined as actions – SET_CAN is linked to the initiating event; the message's sending to the event chosen for the individual example.

7.5.1 Using a Cyclic Timer

A simple alternative to implement this example is the use of a cyclic timer to send the CAN message. Having started the timer once, it is able to run without any manual steps. Since the message's sending shall be stopped as soon as the respective key is released, a further configuration step is still missing – the timer needs to be stopped.

There are three steps necessary to control the timer:

- Initializing the timer parameters (cycle time), e. g. linked to the initiating event,
- Starting the timer, linked to pressing a key (Key pressed),
- Stopping the timer, linked to releasing this key (Key released).

The CAN message is sent according to this timer event.

The timer event is only activated upon expiration of the respective timer. Since the CAN message is to be sent immediately with the defined keypress (i. e. no delay), the action SEND_CAN is additionally linked to the event "Key pressed".

Two steps are necessary to send the CAN message: data initialization and sending command.

The following paragraphs outline the individual operating steps within the client:

At first, the overview for the general setup is opened with the help of "Channel configuration". Apart from the parameters of the CAN interfaces, the user may define here a name for the respective configuration.

Then, both actions, which are allocated to the CAN message, are created. It is already possible at this stage – if necessary – to create actions that give feedback signals to the display ("Set display text"; see CSV-source code in next chapter). The buttons "Set CAN Message" and "Send CAN Message" may be used to open the dialogues to create the respective actions – they are named "setCAN2" and "sendCAN2" in this example.

Before creating the actions to control the timer, the timer event needs to be defined. In this regard, the client is informed of the timer's name and, afterwards, the actions for the timer control are configured.

The timer event is created by allocating actions to a new timer. The desired actions are selected in the left window "Actions", i. e. "sendCAN2" in this example. "Add timer event" opens the dialogue to enter the timer's name. Having successfully created the event, an appropriate entry appears in the right window "Events".

Now, all three actions, able to control the timer, are created. The respective dialogue is opened with the help of "Set timer" – the timer name of the already created event may be selected there from a list.

Having created these actions, they can be allocated to the respective keyboard operation and initializing event. To some extent, this step might have been executed at an earlier stage, however, all actions are now fully defined and may thus be allocated at the same time. Please see the listing of the CSV file for exact information on the allocations.

7.5.2 Using a Single-Shot Timer

As an alternative, the task given in the present example may also be solved with a simple single-shot timer instead of a cyclic timer.

As regards the action to initialize the respective timer, the field "Type" is changed from SET_CYCLIC to SET_SINGLE. The defined keypress starts the timer which creates a single timer event after the preset time and stops. Being in this timer event, the timer may be started once again causing a new timer event after expiration. Thus, a further modification needs to be implemented – the action necessary to start the timer needs to be added to the action list of the timer event.

Cyclic timers offer various advantages in contrast to the solution presented in this chapter. Since the timer starts once again in its own event list, programme runtimes carry more weight with regard to exact cycle times.

Please note:

From firmware version 1.0.5 on, the SimBox supports the starting of timers out of their own event lists.

7.6 Timer-controlled Sending with Keypress and Delay

In accordance with the example described above, this chapter is to exemplify the configuration steps necessary to send a CAN message in regular intervals as long as a predefined key is pressed. However, the interval between first message and first repetition shall be different from the intervals of all following repetitions.

This task consists of two individual phases. The first one comprises the first interval, i. e. keypress until first repetition of the CAN message; the second one corresponds to the permanent repetition of the message until the key is released.

The second phase is very similar to the example presented in chapter 7.5. Thus, this configuration may be used as basis and expanded accordingly.

However, the different idle time, to be considered before the first repetition, is an important deviation. Its implementation requires an individual timer which is configured according to the desired idle time. This timer shall merely activate the second phase, i. e. it may not influence the further process – therefore, it is created as single-shot timer. The predefined keypress starts the first phase as well as this timer; the latter is stopped as soon as it expires or the respective key is released. Please consider that the timer has to be stopped upon release of the key. Otherwise, very short keypresses and immediate releases would activate the associated timer event as well as the second phase.

The second phase comprises the cyclic sending of the CAN message with identical intervals. This configuration has already been described in chapter 7.5.1, i. e. these steps may be transferred to the current example. However, a keypress is no longer used as activator but the end of the first phase which corresponds to the expiration of the timer mentioned above. Thus, the actions, necessary to start the cyclic timer and send the CAN message, need to be implemented in the event of the single-shot timer – it is called “TimerDelay” in this example.

This is also the first step to modify the configuration. Afterwards, all three actions for the single-shot timer’s control may be created – “START”, “STOP” and “SET_SINGLE”. They have to be allocated to the corresponding events; the same goes for the actions necessary to control the cyclic timer.

The actions to initialize both timers as well as the CAN message are allocated to the initiating event so that the parameters are immediately set upon start of the SimBox. The event “keypress” of the desired key forms the basis of the actions defined for the single-shot timer’s start (i. e. “TimerDelay”) and sending of the CAN message (i. e. the first one). As described above, the event of the timer “TimerDelay” is used for the actions to start the cyclic timer and send the CAN message (i. e. the first message after delay). Merely the action to send all other CAN messages (i. e. any further message) is linked to the event of the cyclic timer.

The event “release of key” has to form the basis for both actions to stop the timers since a different timer is active during keypress delay compared to the following stage. In case any of the actions has not been implemented, the configuration will – depending on the release moment – lead to an undesired result.

Please note:

The SimBox keyboard has a keypress delay as well as an autorepeat function – both set with definite parameters. In case these settings correspond to your requirements, the keyboard events “Key pressed” and “Auto repeat” may be directly used.

7.7 Simulating a Speed Indicator

This chapter is to exemplify shortly how measuring values, such as vehicle speed, may be configured in a way that the user is able to modify them. Please see the CSV-source code in the next chapter for further details. The configuration presented in chapter 7.4 forms the basis for the current example which will be modified and expanded accordingly.

The speed value shall be sent in a CAN message having a 16-bit value and intervals of 0,1 km/h. With the help of a predefined keypress, the user shall be able to change the simulated value upwards/downwards in intervals of 5 km/h within a range of 0 to 100 km/h. The autorepeat function of the keyboard shall also be used to offer large-scale modifications of the value when the respective keys are pressed for a longer time. As controlling element, the simulated value shall be presented on the display with km/h as measuring unit.

Any data storage is based on the CAN signal which serves to send the measuring value. The current speed signal is supposed to be stored in the CAN message having as format “Little Endian” and as ID 0x100 in byte 2 and 3. Intervals shall amount to 0,1 km/h, i. e. the CAN signal is processed with this measuring unit. Based on the desired range of 0 to 100 km/h, the raw-data value of the CAN signal needs to be modified within a range of 0 to 1000 – an interval of 5 km/h corresponds to an interval of 50 for the CAN signal.

In order to change the CAN signal’s value upwards/downwards, two similar actions “Modify CAN message” are created. The bit position of the CAN signal’s LSB is set to 16. Please note that, with regard to “Big Endian”-notation, the LSB is not located in the first, but in the last byte of the corresponding data range, i. e. the correct bit position is different. In the current example, the length of the CAN signal is two byte and, thus, a value of 16 bit needs to be added to the length field.

Miscellaneous types of operators may be entered for the action “Modify CAN message”. The categories “Add” and “Subtract” are available in three different variants each which are configurable depending on the specific use case. The aim of the current example is to stop, i. e. no longer change, the simulated value at the range limits if the respective keys for higher/lower speed are permanently pressed – when reaching 0, the action “Subtract” is no longer carried out and, thus, the simulated value is not suddenly turned into maximum speed. For this purpose, the variants “ADDLIMIT” and “SUBTRACTLIMIT” are used. Having activated the respective action, the value does not pass beyond the defined range limits as soon as these have been reached. The limits of the value range are specified in the fields “min” and “max” – for this purpose, the raw-data values of the CAN signal have to be entered, i. e. 0 and 1000 in the current example. In addition, the field “Value / Step” is used to set the interval according to the measuring unit of the raw data, i. e. 50 in this case. These actions can be found in the CSV list under “increaseSpeed” and “decreaseSpeed”.

In order to display the respective value, two actions are necessary: “Copy CAN message” and “Show value”. The first one serves to convert the raw-data value of the CAN signal into a variable which is then displayed with the help of the second action. Explanatory notes may be added before the current speed value is sent. With regard to the present example, the actions are called “copySpeed”, “showSpeed” and “textSpeed”.

The action “showSpeed” contains a conversion factor. The raw data of the speed signal are displayed in intervals of 0,1 km/h; however, the action “Show value” delivers only integers. This is why the raw-data value is divided by 10 and delivered in intervals of 1 km/h. The parameter “Scale” is set to -10: The negative sign implies that the value of the variable “varSpeed” is divided by 10 and then displayed whereas a positive one instructs the SimBox to multiply the variable value.

The next step is to link these actions with keyboard events. The actions “increaseSpeed”, “copySpeed” and “showSpeed” are allocated to the event “pressed” of the key “LEFT_UP”. Please consider the order defined for the action list of the event: The action “increaseSpeed” is supposed to be in first position since it serves to modify data; next is “copySpeed” in order to convert the modified data into a variable which is in turn followed by “showSpeed” serving to display the variable’s value. If “increaseSpeed” was in last position, the SimBox would deliver an old value of the speed signal upon keypress and, afterwards, it would send a converted value as CAN message – thus, the user would no longer be able to retrace the data actually sent.

In case a signal value shall be modified on a large scale, it is more convenient to use the autorepeat function of the keyboard instead of pressing a key several times. For this purpose, the current configuration is extended by allocating the same action list, used for the event “pressed”, to the event “auto repeat” of the key “LEFT_UP” – please consider again the order of actions.

In order to reduce the simulated speed value, the user needs to create two events for the key “LEFT_DOWN” (similar to the ones for the key “LEFT_UP”) and exchange the action “increaseSpeed” by “decreaseSpeed”.

The actions “textSpeed”, “copySpeed” and “showSpeed” are added to the initiating event so that the current value is displayed upon start of the SimBox. The order is again important: “copySpeed” is to be processed before “showSpeed”.

The configuration presented in this chapter is derived from the example of chapter 7.4. The latter uses the key “C” to start the cyclic timer, i. e. CAN messages are only sent upon keypress according to a defined cycle time. As an alternative, the timer’s start may be transferred to the initiating event so that the SimBox begins to send CAN messages as soon as it has been switched on. Please consider again the order: A timer has to be initialized first before starting it.

In order to offer enough space for the additional bits of the speed signal, the DLC information used for the CAN message is extended to four byte.

8 CSV Configurations for Presented Examples

Please note:

Due to the formatting of the current document, the source code of the individual configurations may have line breaks which are of course not used in a CSV file.

Chapter 7.1 – “Hallo Welt” (“Hello World”):

```
SimBoxClientConfiguration;Hello World
SPECVERSION;0.0.0.3
CAN_1;;0;lowspeed;1000;0000
CAN_2;;0;lowspeed;1000;0000
ACTION;SET_TEXT;hello;;0;0;1;1;Hallo Welt
INITIALIZATION;hello
```

Chapter 7.2 – Advanced “Hallo Welt” (“Hello World”):

```
SimBoxClientConfiguration;Hello World
SPECVERSION;0.0.0.3
CAN_1;;0;lowspeed;1000;0000
CAN_2;;0;lowspeed;1000;0000
ACTION;SET_TEXT;clear;;0;0;1;1;          ;
ACTION;SET_TEXT;hello;;0;0;1;1;Hallo
ACTION;SET_TEXT;world;;0;0;7;1;Welt
INITIALIZATION;hello
KEY_PRESSED;F1;world
KEY_PRESSED;F2;clear
```

Chapter 7.3 – Sending a CAN message:

```
SimBoxClientConfiguration;send CAN
SPECVERSION;0.0.0.3
CAN_1;CAN1;1;highspeed;500000;0000
CAN_2;CAN2;1;highspeed;500000;0000
ACTION;MODIFY_CAN_SIG;modifyCan100;;0;CAN_1;0;100;ADD;0;8;1;0;0;0
ACTION;SEND_CAN;sendCan100;;0;CAN_1;0;100;;;0;0;0;;;;;
ACTION;SET_CAN;setCan100;;0;CAN_1;0;100;2;00 00      ;FF FF
INITIALIZATION;setCan100
KEY_PRESSED;A;sendCan100
KEY_PRESSED;B;modifyCan100
```

Chapter 7.4 – Timer-controlled sending and displaying of value:

```
SimBoxClientConfiguration;send CAN
SPECVERSION;0.0.0.3
CAN_1;CAN1;1;highspeed;500000;0000
CAN_2;CAN2;1;highspeed;500000;0000
ACTION;COPY_CAN_SIGNAL;copySignal100;;0;CAN_1;0;100;READCANSIGNAL;0;8;variable100;0
ACTION;MODIFY_CAN_SIG;modifyCan100;;0;CAN_1;0;100;ADD;0;8;1;0;0;0
ACTION;SEND_CAN;sendCan100;;0;CAN_1;0;100;;;0;0;0;;;;;
ACTION;SET_CAN;setCan100;;0;CAN_1;0;100;2;00 00      ;FF FF
ACTION;SET_TIMER;setTimer100;;0;Timer100;set_cyclic;500
ACTION;SHOW_VALUE;show100;;0;0;1;2;variable100;0;0;3;unsigned;right
ACTION;SET_TIMER;startTimer100;;0;Timer100;start;0
INITIALIZATION;setCan100:setTimer100
```

```
TIMER;Timer100;modifyCan100:copySignal100:sendCan100:show100;;1
KEY_PRESSED;A;sendCan100
KEY_PRESSED;B;modifyCan100
KEY_PRESSED;C;startTimer100
```

Chapter 7.5.1 – Timer-controlled sending with keypress (cyclic timer):

```
SimBoxClientConfiguration;Pushbutton test
SPECVERSION;0.0.0.3
CAN_1;CAN1;1;highspeed;500000;0000
CAN_2;CAN2;1;highspeed;500000;0000
ACTION;SET_TEXT;clearF2;;0;0;4;4;  ;
ACTION;SEND_CAN;sendCAN2;;0;CAN_2;0;222;;;0;0;0;;;;;
ACTION;SET_CAN;setCAN2;;0;CAN_2;0;222;2;22 22      ;ff ff
ACTION;SET_TIMER;setTimer2;;0;Timer2;set_cyclic;100
ACTION;SET_TIMER;startTimer2;;0;Timer2;start;0
ACTION;SET_TIMER;stopTimer2;;0;Timer2;stop;0
ACTION;SET_TEXT;textF2;;0;0;4;4;F2
ACTION;SET_TEXT;textHello;;0;0;1;1;Pushbutton test
INITIALIZATION;textHello:setCAN2:setTimer2
TIMER;Timer2;sendCAN2;;1
KEY_PRESSED;F2;textF2:startTimer2:sendCAN2
KEY_RELEASED;F2;clearF2:stopTimer2
```

Chapter 7.5.2 – Timer-controlled sending with keypress (single-shot timer):

```
SimBoxClientConfiguration;Pushbutton test
SPECVERSION;0.0.0.3
CAN_1;CAN1;1;highspeed;500000;0000
CAN_2;CAN2;1;highspeed;500000;0000
ACTION;SET_TEXT;clearF2;;0;0;4;4;  ;
ACTION;SEND_CAN;sendCAN2;;0;CAN_2;0;222;;;0;0;0;;;;;
ACTION;SET_CAN;setCAN2;;0;CAN_2;0;222;2;22 22      ;ff ff
ACTION;SET_TIMER;setTimer2;;0;Timer2;set_single;100
ACTION;SET_TIMER;startTimer2;;0;Timer2;start;0
ACTION;SET_TIMER;stopTimer2;;0;Timer2;stop;0
ACTION;SET_TEXT;textF2;;0;0;4;4;F2
ACTION;SET_TEXT;textHello;;0;0;1;1;Pushbutton test
INITIALIZATION;textHello:setCAN2:setTimer2
TIMER;Timer2;startTimer2:sendCAN2;;1
KEY_PRESSED;F2;textF2:startTimer2:sendCAN2
KEY_RELEASED;F2;clearF2:stopTimer2
```

Chapter 7.6 – Timer-controlled sending with keypress and delay:

```
SimBoxClientConfiguration;Pushbutton test
SPECVERSION;0.0.0.3
CAN_1;CAN1;1;highspeed;500000;0000
CAN_2;CAN2;1;highspeed;500000;0000
ACTION;SET_TEXT;clearF2;;0;0;4;4;  ;
ACTION;SEND_CAN;sendCAN2;;0;CAN_2;0;222;;;0;0;0;;;;;
ACTION;SET_CAN;setCAN2;;0;CAN_2;0;222;2;22 22      ;ff ff
ACTION;SET_TIMER;setTimer2;;0;Timer2;set_cyclic;100
ACTION;SET_TIMER;setTimer-Delay;;0;Timer-Delay;set_single;1000
```

```
ACTION;SET_TIMER;startTimer2;;0;Timer2;start;0
ACTION;SET_TIMER;startTimer-Delay;;0;Timer-Delay;start;0
ACTION;SET_TIMER;stopTimer2;;0;Timer2;stop;0
ACTION;SET_TIMER;stopTimer-Delay;;0;Timer-Delay;stop;0
ACTION;SET_TEXT;textF2;;0;0;4;4;F2
ACTION;SET_TEXT;textHello;;0;0;1;1;Pushbutton test
INITIALIZATION;textHello:setCAN2:setTimer2:setTimer-Delay
TIMER;Timer2;sendCAN2;;1
TIMER;Timer-Delay;startTimer2:sendCAN2;;1
KEY_PRESSED;F2;textF2:startTimer-Delay:sendCAN2
KEY_RELEASED;F2;clearF2:stopTimer2:stopTimer-Delay
```

Chapter 7.7 – Simulating a speed indicator:

```
SimBoxClientConfiguration;speed
SPECVERSION;0.0.0.3
CAN_1;CAN1;1;highspeed;500000;0000
CAN_2;CAN2;1;highspeed;500000;0000
ACTION;COPY_CAN_SIGNAL;copySignal100;;0;CAN_1;0;100;READCANSIGNAL;0;8;variable100;0
ACTION;COPY_CAN_SIGNAL;copySpeed;;0;CAN_1;0;100;READCANSIGNAL;16;16;varSpeed;0
ACTION;MODIFY_CAN_SIG;decreaseSpeed;;0;CAN_1;0;100;SUBTRACTLIMIT;16;16;32;0;0;3e8
ACTION;MODIFY_CAN_SIG;increaseSpeed;;0;CAN_1;0;100;ADDLIMIT;16;16;32;0;0;3e8
ACTION;MODIFY_CAN_SIG;modifyCan100;;0;CAN_1;0;100;ADD;0;8;1;0;0;0
ACTION;SEND_CAN;sendCan100;;0;CAN_1;0;100;;;0;0;0;;;;;
ACTION;SET_CAN;setCan100;;0;CAN_1;0;100;4;00 00 00 00    ;FF FF ff ff
ACTION;SET_TIMER;setTimer100;;0;Timer100;set_cyclic;500
ACTION;SHOW_VALUE;show100;;0;0;1;2;variable100;0;0;3;unsigned;right
ACTION;SHOW_VALUE;showSpeed;;0;0;7;3;varSpeed;0;-10;4;unsigned;right
ACTION;SET_TIMER;startTimer100;;0;Timer100;start;0
ACTION;SET_TEXT;textSpeed;;0;0;1;3;speed:
INITIALIZATION;setCan100:setTimer100:textSpeed:copySpeed:showSpeed
TIMER;Timer100;modifyCan100:copySignal100:sendCan100:show100;;1
KEY_PRESSED;A;sendCan100
KEY_PRESSED;B;modifyCan100
KEY_PRESSED;C;startTimer100
KEY_AUTOREPEAT;LEFT_DOWN;decreaseSpeed:copySpeed:showSpeed
KEY_PRESSED;LEFT_DOWN;decreaseSpeed:copySpeed:showSpeed
KEY_AUTOREPEAT;LEFT_UP;increaseSpeed:copySpeed:showSpeed
KEY_PRESSED;LEFT_UP;increaseSpeed:copySpeed:showSpeed
```

9 SimBox Bootloader

The firmware of the SimBox is roughly divided into three parts each having individual tasks:

- Bootloader
- Firmware
- Configuration

The firmware and especially the configuration are those parts the user normally interacts with. They determine the functionality of the SimBox. Every time the user creates a new configuration with the SimBox client and sends it to the SimBox, the area of the flash memory, the SimBox uses for storage of that configuration, is rewritten. Moreover, the firmware may be updated in case new versions with additional functions or bugfixes are available.

9.1 Bootloader

The bootloader of the SimBox is only involved if a new configuration or a new firmware is to be written to the flash memory. In the event of an update, the bootloader receives the new data from a client application and, in turn, writes them to the memory. Although the bootloader represents the first part being activated as soon as the SimBox is switched on, it stays invisibly in the background during normal operation and appears only upon request. In exceptional cases, however, it is able to refuse a start of the firmware and request an update instead.

If the bootloader is in update mode, it informs about its current state by showing different messages on the SimBox display – these will be described in the following paragraphs. In all cases, the right lower corner of the display shows a rotating bar whenever the bootloader communicates with the SimBox client on the computer. There is no data exchange, however, when this bar stops rotating for a longer time (>10 s) or a letter appears.

Having successfully updated the firmware or a configuration, the bootloader tries to switch the SimBox to standby mode. Depending on external circumstances, the SimBox may immediately switch on again, e. g. if it is connected to a running test setup with active CAN buses and, thus, receives an immediate wake-up signal.

Please note:

The firmware or a configuration should never be updated as long as the SimBox is connected to a running system. Depending on the imported configuration, the SimBox might affect the operation of other devices connected to a CAN bus. This is why the function of a configuration should be verified after having installed an update and before connecting the SimBox to a running system.

The bootloader has a timeout function. If there is no communication with a client for a specific period (approximately 40 s), the SimBox is also set to standby mode.

Please note:

If an update process is cancelled, the SimBox might get inoperable. Please ensure that the SimBox is connected to a stable power supply and that the client may run on the computer without any interruptions.

9.2 Activation by Firmware

Having switched on the SimBox, the firmware usually starts, reads the active configuration and executes the instructions given there. In case a new firmware or configuration shall be imported into the SimBox, the latter has to be in update mode and the corresponding function is to be started in the SimBox client – both conditions need to be fulfilled before starting the update process. The update mode has to be activated by the firmware and, thus, the necessary operation step depends on the firmware version currently available on the SimBox before the respective update process is started.

For example, the keys “S” and “Down” – located on the right side of the keyboard – are simultaneously pressed in order to set a SimBox with firmware version 1.0.5 in update mode:

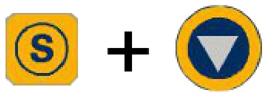
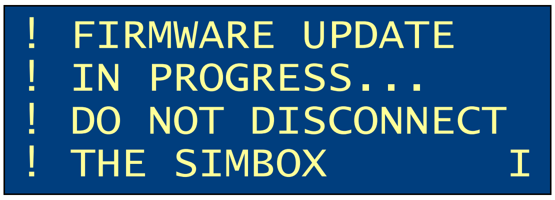


Figure 26: Activation of bootloader

Afterwards, the SimBox display shows the following message of the bootloader. The right lower corner of the display presents a letter which turns into a rotating bar as soon as a client tries to contact the SimBox.



```
! FIRMWARE UPDATE
! IN PROGRESS...
! DO NOT DISCONNECT
! THE SIMBOX         I
```

Figure 27: SimBox display during firmware update

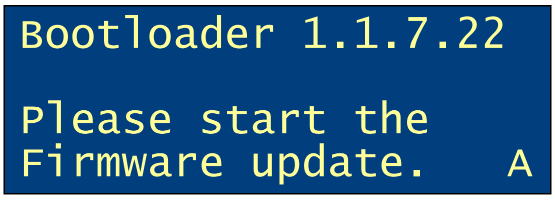
As regards older versions of the SimBox firmware (e. g. v1.0.1), a different key combination is valid: “F9”, “F10” and “Down” – located in the right control field – need to be pressed simultaneously.

9.3 Start after Cancelled Update Process

In some cases, the bootloader is able to recognize an update failure of the firmware or configuration. It assumes that the firmware is incomplete and cannot be executed. Being in this state, the user would not be able to use it, i. e. activating the bootloader for a new update process would not be possible either.

A further error case is that the SimBox does not have any firmware the bootloader may start. As a consequence, the bootloader cannot be activated by the firmware.

This is why the bootloader directly switches to update mode in these cases and, thus, the user is able to start immediately a new update process. The SimBox bootloader indicates this operating mode with an individual message on the display (see Figure 28). Please try to repeat the last update process as soon as this message appears (firmware or configuration).



```
Bootloader 1.1.7.22

Please start the
Firmware update.   A
```

Figure 28: SimBox display after cancelled update process

As soon as the SimBox client requests to switch the SimBox to update mode, no further operating steps need to be carried out on the SimBox – the update mode has already been activated. Please note that the timeout function of the bootloader is active and, thus, the SimBox might switch off before the client starts to exchange data – in this case, please repeat the update process.

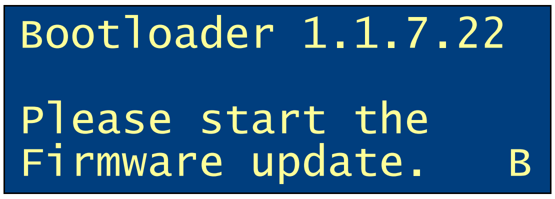
The bootloader does not distinguish between memory areas for firmware or configuration when repeating cancelled update processes. It might happen that the bootloader tries to start an incomplete firmware if its faulty update is followed by a successful update of a configuration. In this case, the update mode cannot be started with the help of the firmware – please follow then the instructions given in the next chapter.

9.4 Start with Key Combination

It might happen that the SimBox reaches a status where neither the firmware can be instructed to activate the update mode nor the bootloader is able to recognize this error status and switch to the update mode automatically. As long as the bootloader is still intact, the update mode can be activated with the help of a key combination when switching on the SimBox – a new firmware can then be loaded.

In order to activate this mode, please ensure first that the SimBox has been switched off or set to standby mode. The latter is characterized by low power consumption and, thus, the display is switched off. In the event of defective firmware, the SimBox might also leave the display inactive. If the operating mode of the SimBox cannot be identified, simply disconnect it from any power supply.

Please press simultaneously the keys “F9”, “F10” and “Down” on the right control field. Being in this state, the SimBox can be started by connecting it to a power supply or by pressing the key “S” additionally. As soon as the bootloader has been successfully started, the SimBox display shows the message of Figure 29.



```
Bootloader 1.1.7.22

Please start the
Firmware update.   B
```

Figure 29: SimBox display after key combination

Please note:

The best way is to prepare first the SimBox client for a new update process and to switch on the SimBox afterwards, so that the process can be started immediately. Due to the fact that the time-out function of the bootloader is also active in this case, the SimBox might switch off itself while the client is being prepared.

Appendix A: Technical Data and Support

General Data	
Power supply voltage	8,5V..16V, 12V (typ.)
Operating current (ca.)	200mA...250mA (depending on display backlight)
Standby current (ca.)	0,1mA
Operating temperature	0 °C to 50 °C (32 °F to 122 °F)
Storage temperature	-20 °C to 70 °C (-4 °F to 158 °F)
Weight (ca.)	370g (0,81 pounds)
Case	
Size (ca.)	180mm x 85mm x 25mm (7.09" x 3.35" x 0.98")
Operating controls	Keyboard with 25 keys
Displays	Character display, 20 characters x 4 lines
Connectors	26-pin D-SUB-connector (power supply and CAN buses)
	9-pin D-SUB-connector (RS232, firmware update)

Support:
http://www.telemotive.de/
mailto:produktsupport@telemotive.de

Appendix B: Pin Assignment

The SimBox has two connectors with D-SUB-format. The 26-pin connector on the right side is used for its power supply and all signals necessary for a simulation; the 9-pin connector on the left side serves to connect it with a computer in order to transfer configurations and updates to the SimBox. The 9-pin connector corresponds to a usual RS232-assignment and may be directly linked to a COM-port of a computer.

26-pin D-SUB-connector (DA26, male), located on the right side:

Pin	Description
1, 2	Clamp 30 (positive supply voltage)
3	Digital out 0
4	CAN channel 2: CAN_H, low speed
5	CAN channel 1: CAN_H, low speed
6	Analog input 3: ground
7	Analog input 2: ground
8	Analog input 1: ground
9	Analog input 0: ground
10, 11	Clamp 31 (negative supply voltage)
12	Digital out 1
13	CAN channel 2: CAN_L, low speed
14	CAN channel 1: CAN_L, low speed
15	Analog input 3: signal
16	Analog input 2: signal
17	Analog input 1: signal
18	Analog input 0: signal
19	CAN channel 2: CAN_L, high speed
20	CAN channel 2: CAN_H, high speed
21	CAN channel 1: CAN_L, high speed
22	CAN channel 1: CAN_H, high speed
23	Digital input 3
24	Digital input 2
25	Digital input 1
26	Digital input 0

9-pin D-SUB-connector (DE9, female), located on the left side:

Pin	Description
1	n.c.
2	TxD (SimBox, connects to RxD of computer)
3	RxD (SimBox, connects to TxD of computer)
4	n.c.
5	GND
6	n.c.
7	RTS
8	CTS
9	n.c.

Appendix C: History Of Modifications

Version	**Date**	**Description**	**Author**
1.0	02/05/2013	File created	Helmut Sochor