R_Visualizer Manual

Christian Neuberger

November 24, 2016

Contents

1 About R_Visualizer

R_Visualizer is a graphical tool used to observe and control the state of a compatible system. Said system is connected to the host pc running R_Visualizer via the CAN-Analyser USB interface. This interface will continually receive CAN messages from the system and propagate these messages to R_Visualizer and vice versa. Received messages will subsequently be visualized by R_Visualizer. The application itself is capable of sending messages to the system via the CAN-Analyser USB interface to control the system state or behavior.

2 Installing

There is no installation needed. Simply unzip the provided zip file to any location you desire that does not require administrative priviledges and start the application via the *R_Visualizer.exe* file. Do not delete any file or folder from the unzipped folder.

3 Getting Started

3.1 Overview

The general user interface is designed with simplicity in mind. The main goal was to expose as much information as possible without distracting the user with cluttered configuration dialogs. Therefore, the interface is divided into two main parts: the message stream on the left-hand side and the overview/configuration dialogs on the right-hand side.

Most of the application is tailored to the current user. A user can either be a regular user or an *Admin*. The regular user is only exposed to basic functionality that is needed to monitor and control the observed system. On the other hand, the *Admin* user has advanced functionality like storing or editing configuration entries. The user roles can be switched with the respective symbol in the shortcut panel or from the *User* menu at the top of the application.

3.1.1 Menu

The *Menu* is comprised of the items:

- File (see Fig. 2)
  can be used to load a recent *Message Stream* or store the current *Message Stream*. Additionally, the *Error Log* is accessible from the *File* menu.[1]

  - New - TESTING ONLY

---

[1]Currently the *New* option is used for testing. It does not flush the *Message Stream* to create a new stream but rather adds example messages to the stream...
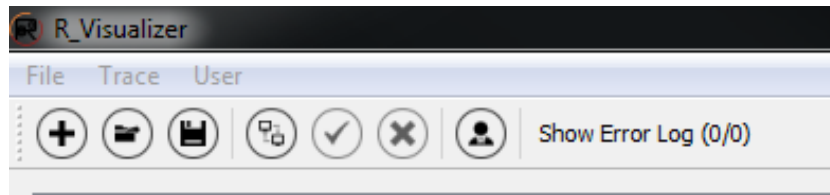
Figure 1:

- Open - Open a previously saved *Message Stream*
- Save - Save the current *Message Stream*
- Show Error Log - Displays the current *Error Log*[2]

- Trace (see Fig. 3)
  can be used to establish a connection to the CAN-Analyser USB interface and to disconnect R_Visualizer from the CAN-Anylser USB interface, as well as starting and stopping the recording of messages.[3]

  - (Dis)Connect - Connect or disconnect to the CAN-Analyser USB interface[4]
  - Start - Start recording incoming messages to the *Message Stream*(Connect first!)
  - Stop - Stop recording incoming messages to the *Message Stream*

- User (see Fig. 4)
  can be used to switch between *Admin* and *User* mode.

  - Admin/User - Switch to the currently displayed user role

All items from all sub-menus can also be accessed through the respective shortcut buttons in the menu panel.

3.1.2 Message Stream

To achieve maximum exposure of the system behavior, the *Message Stream* is visible in all configuration dialogs and during the system overview dialog. The incoming messages are continuously updated and displayed.

The *Message Stream* displays messages in the order they are picked up by the CAN-Anylser USB interface: in a chronological order. The messages visualization is customizable via the configuration dialogs in the *Message Config* tab (see Fig. 5). Per default messages are displayed with their respective timestamp, id and code plus data.

The user can use scrolling to view any messages that were picked up by the *Messages Stream*, there is no limitation on the history.

---

[2] ONLNY CAN errors are displayed here.

[3] Keep in mind that stopping the message recording will stop the Message Stream from storing incoming messages. Messages that are not stored by the Message Stream are lost and cannot be retrieved by any means.

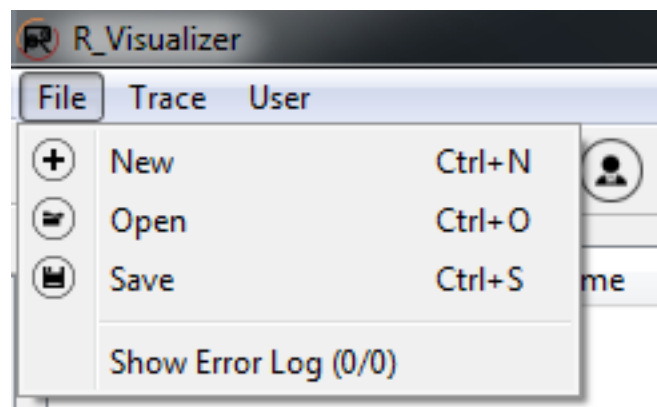[4] The functionality of the menu entry depends on the current connection state.
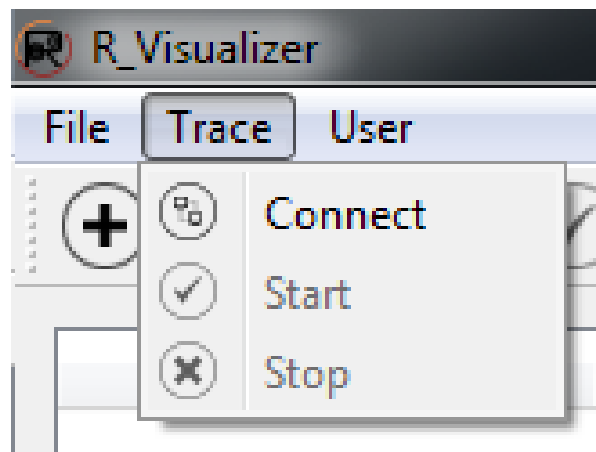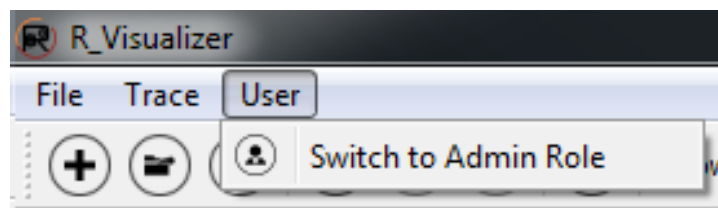
Figure 2:

Figure 3:

Figure 4:

Timestamp	Name	Message
16.11.2016 - 20:03:21.404	Master	Code: SYS_TICK_code Tick nr. 6
16.11.2016 - 20:03:21.557	Pump	Code: PUMP_STAT_OFF PUMP is OFF
16.11.2016 - 20:03:21.719	Pump	Code: PUMP_STAT_ON_code PUMP is ON
16.11.2016 - 20:03:21.881	Pump	Code: PUMP_STAT_OFF PUMP is OFF
16.11.2016 - 20:03:22.042	Pump	Code: PUMP_STAT_ON_code PUMP is ON
16.11.2016 - 20:03:22.064	Temp	Code: TEMP_code TEMP: 0°C
16.11.2016 - 20:03:22.066	Temp	Code: TEMP_code TEMP: 1°C
16.11.2016 - 20:03:22.068	Temp	Code: TEMP_code TEMP: 2°C
16.11.2016 - 20:03:22.069	Temp	Code: TEMP_code TEMP: 3°C
16.11.2016 - 20:03:22.071	Temp	Code: TEMP_code TEMP: 4°C
16.11.2016 - 20:03:22.073	Temp	Code: TEMP_code TEMP: 5°C
16.11.2016 - 20:03:22.074	Temp	Code: TEMP_code TEMP: 6°C
16.11.2016 - 20:03:22.076	Temp	Code: TEMP_code TEMP: 7°C
16.11.2016 - 20:03:27.380	Master	Code: SYS_TICK_code Tick nr. 7
16.11.2016 - 20:03:27.534	Pump	Code: PUMP_STAT_ON_code PUMP is ON
16.11.2016 - 20:03:27.695	Pump	Code: PUMP_STAT_OFF PUMP is OFF
16.11.2016 - 20:03:27.857	Pump	Code: PUMP_STAT_ON_code PUMP is ON
16.11.2016 - 20:03:28.019	Pump	Code: PUMP_STAT_OFF PUMP is OFF
16.11.2016 - 20:03:33.358	Master	Code: SYS_TICK_code Tick nr. 8
16.11.2016 - 20:03:33.512	Pump	Code: PUMP_STAT_OFF PUMP is OFF
16.11.2016 - 20:03:39.334	Master	Code: SYS_TICK_code Tick nr. 9
16.11.2016 - 20:03:39.487	Pump	Code: PUMP_STAT_OFF PUMP is OFF
16.11.2016 - 20:03:45.311	Master	Code: SYS_TICK_code Tick nr. 10
16.11.2016 - 20:03:51.287	Master	Code: SYS_TICK_code Tick nr. 11
16.11.2016 - 20:03:57.264	Master	Code: SYS_TICK_code Tick nr. 12
16.11.2016 - 20:03:57.417	Pump	Code: PUMP_STAT_ON_code PUMP is ON
16.11.2016 - 20:03:57.579	Pump	Code: PUMP_STAT_ON_code PUMP is ON
16.11.2016 - 20:04:03.241	Master	Code: SYNC_code Message: 0x00 0x00 0x70 0x3e 0x00 0x00 0x00
16.11.2016 - 20:04:03.242	Master	Code: SYNC_code Message: 0x00 0x00 0x70 0x3e 0x00 0x00 0x00
16.11.2016 - 20:04:03.243	Master	Code: SYS_TICK_code Tick nr. 1
16.11.2016 - 20:04:03.395	Pump	Code: PUMP_STAT_OFF PUMP is OFF
16.11.2016 - 20:04:09.217	Master	Code: SYS_TICK_code Tick nr. 2

Figure 5: The *Message Stream* displaying incoming messages

3.1.3 System Overview

CURRENTLY NOT IN A PRODUCTION STATE. USE AT OWN RISK.

The System Overview tab can be used to visually recreate the connected system and observe the system's current state.

3.1.4 Send Messages

In order to communicate and control the system, the *Send Messages* tab provides the means to send single messages or whole packages of messages to the system (see Fig. 6)

Sending single messages can be achieved with the Send Single Message sub-dialog within the *Send Messages* tab. Type in an ID that shall be exposed to the system as the sender of the message, the desired code to send and additionally the data you want to send with the message.

The ID/Name field can be filled with a numerical value in decimal or hexademical format[5]. If the ID to name mapping has already been loaded or created in the *Message Config* tab, it is also possible to type in the specified name or choose one from the auto-completion list.

The Code field can be filled with a numerical value in decimal or hexademical format[6]. If the Code to name mapping has already been loaded or created in the *Message Config* tab, it is also possible to type in the specified name or choose one from the auto-completion list.

The Data field can be filled with numerical values according to your selection in the the combo box right next to the data field. If Hex is specified you can enter hex values in portions of bytes, else if Dec Data is specified you can enter decimal values in portions of bytes[7], else if Dec Value is specified you can enter a decimal value which will automatically be translated to bytes by the application[8], else if Bin is selected you can enter the data byte-wise in binary notation. When switching the selection of the combo box the value is automatically translated from the last selection to the new selection[9]

3.1.5 Message Config

The *Message Config* tab directly interferes with the *Message Stream*. It consists of three separate sub-dialogs: Filter, IDs and Message Types (see Fig. 7)

The Filter section can be used to filter the *Message Stream* for certain properties (see Fig. 8). The filters are chained. To be more specific, the filters interact with each other to display only messages that suffice all enabled filter criteria at onces. Use multiple

---

[5] for hexadecimal prepend 0x to the hexadecimal number
[6] for hexadecimal prepend 0x to the hexadecimal number
[7] Keep in mind that one byte must not exceed the number 255
[8] Keep in mind not to exceed the 56bit limitation, which is 72057594037927935
[9] Keep in mind that translating from a value-based notation to a byte-wise notation or vice versa might mess up the byte order.
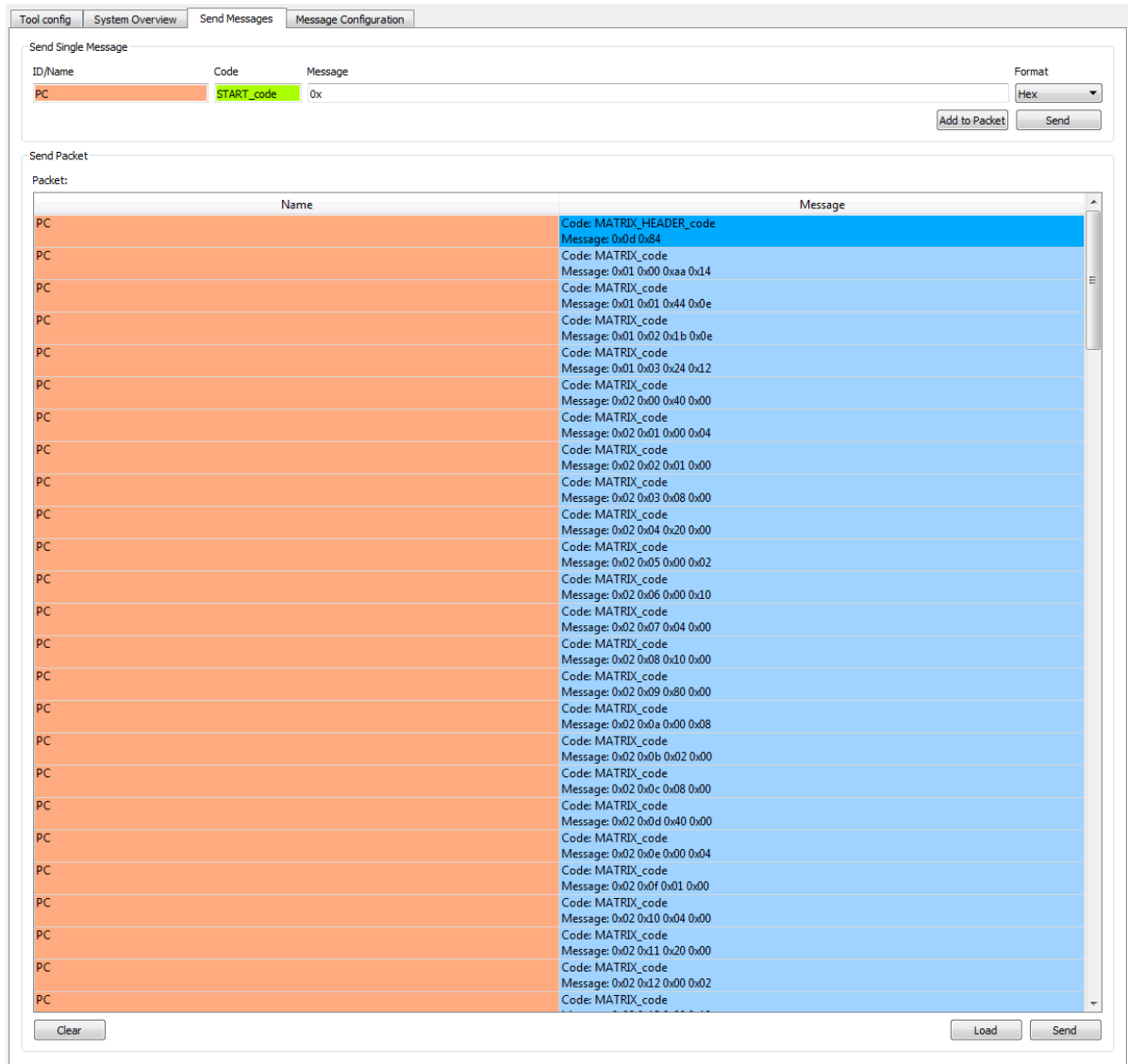
Figure 6: The *Send Messages* dialog that can be used to send single messages or whole packages of messages in the specified order

Tool config | System Overview | Send Messages | Message Configuration

Filter:

ID/Name

Master
PC

Disable   Add   Remove

Message Code

SYS_TICK_code
MATRIX_code
OK_code

Enable   Add   Remove

Timestamp

From 16.11.2016 12:06:00.000   Disable
To   16.11.2016 12:07:00.000   Disable

IDs

ID	Name	Color	
0x00ff	PC	#ffaa7f	Load
0x00f0	Master	#abd0ff	
0x0100	Pump	#d7b4ff	
0x0110	Temp	#ff9c9c	
0x0120	Valve0	#7a5d7f	
0x0121	Valve1	#7f5b71	

Message Types

Code	Name	Message Format	Color	
0x0b	MATRIX_code		#a0d3ff	Load
0x10	SYS_TICK_code	Tick nr. #Data0#	#00ff95	
0x20	PROGRAM_code		#ffff7f	
0x40	SYNC_code		#86c5ff	
0x4f	OK_code		#00ce00	
0x50	PUMP_SENT_code		#d778ff	
0x51	PUMP_ON_code		#bfa6ff	
0x52	PUMP_OFF_code		#ffa2ff	

Figure 7: The *Message Config* dialog that can be used to enhance the visual represen-
tation of messages in the *Message Stream* and to filter for specific messages in
the *Message Stream*

9

Figure 8: The *Message Config* Filters that can be used to filter the *Message Stream* for specific messages

filters to get a fine-grained filtering of the *Message Stream*. Since the *Messages Stream* is updated live, it is advised to first setup all of the desired filter criteria and then enable all filters, starting from the most general one.

The ID/Name subsection can be used to filter for certain IDs or -provided the ID to name mapping has been setup appropriately- names. To create an ID filter, simply click the *Add* button and type in the desired ID in either decimal or hexadecimal representation[10]. If and only if an ID to name mapping has been set up, the respective names can be entered directly or chosen from the auto-completion dialog. To remove a filter, select the according entry from the list and click on the *Remove* button. Per default the ID/Name filter is disabled. To enable the filter click on the *Enable* button.[11].

The Code subsection can be used to filter for certain code or -provided the code to name mapping has been setup appropriately- names. To create a code filter, simply click the *Add* button and type in the desired Code in either decimal or hexadecimal representation[12]. If and only if a code to name mapping has been set up, the respective names can be entered directly or chosen from the auto-completion dialog. To remove a filter, select the according entry from the list and click on the *Remove* button. Per default the ID/Name filter is disabled. To enable the filter click on the *Enable* button.[13].

The Timestamp subsection can be used to filter for a certain period of time. Both, the *from* as well as the *to*, can be enabled or disabled independently from another.

---

[10] for hexadecimal prepend 0x to the hexadecimal number

[11] It is advised to first add all desired filter criteria before enabling the filter, because the *Message Stream* will be updated with each new entry which leads to increased workload

[12] for hexadecimal prepend 0x to the hexadecimal number

[13] It is advised to first add all desired filter criteria before enabling the filter, because the *Message Stream* will be updated with each new entry which leads to increased workload

Figure 9: The *Message Config* IDs that can be used to set alias name and coloring for the *Message Stream's* Name column

Setting the *from* parameter will automatically adjust the *to* parameter and vice versa to form a meaningful time span -even if disabled. Again, it is advised to adjust the parameters with the filters being disabled in order to gain performance.

The IDs section can be used to alter the visualization of the *Message Stream's Name* column (see Fig. 9). Each ID can be assigned to an alias and a color. If a message in the *Message Stream* matches against a configured ID, its name will be displayed instead of the numerical representation and the respective field will be colored in the configured color. This is especially useful to rapidly distinguish different message senders in the *Message Stream*. In addition, all fields in all other menus that require IDs are updated to auto-complete if a name to a configured ID is entered. Most of the fields are then also colored respectively.

The Message Types section can be used to alter the visualization of the *Message Stream's Message* column (see Fig. 10). Each Code can be assigned to an alias and a color. In addition, a message format can be set with the message formatter dialog that will be used to parse the data portion and code of an incoming message to a visually appealing format. If a message in the *Message Stream* matches against a configured Code, its name will be displayed instead of the numerical representation and the respective field will be colored in the configured color. In addition, the configured message format will be used to enhance the representation of the message. This is especially useful to rapidly distinguish different messages by code in the *Message Stream*. In addition, all fields in all other menus that require Codes are updated to auto-complete if a name to a configured Code is entered. Most of the fields are then also colored respectively.

3.2 Setting up R_Visualizer

3.3 Connecting the Host PC to the System

The CAN-Analyser USB interface has exactly two outputs: a standard USB connector and a four pin CAN connector. The CAN connector shall be connected to the observed

Figure 10: The *Message Config* Message Types that can be used to set alias name and coloring for the *Message Stream's* Message column and format the messages' data

System's CAN bus, whereas the USB connector shall be connected to the host PC.

3.3.1 Connection to the System

Make sure that the system is not powered[14]. Connect the CAN-Analyser USB interface to one of the free connectors on the system's CAN bus using a supplied CAN compatible cable.

3.3.2 Connection to the Host PC

Simply plug the CAN-Analyser USB interface into one of your host PC's USB ports. Windows should be able to deduct the needed driver and install the CAN-Analyser USB interface accordingly.

The drivers needed are the HID driver and a USB driver. When the host PC is equipped with either a keyboard, a mouse or any other kind of standard input device, the HID driver should be pre-installed. When the host PC is equipped with USB ports, the USB drivers should be pre-installed.

It is advised to use a USB extension cable. Since the CAN-Anylser USB interface is moderately sized and only held by the USB connector, the USB connector can easily be destroyed by force in either direction when applied to the plugged in CAN-Analyser USB interface. Moreover, the whole CAN-Analyser USB interface might take damage when exposed to forced.

Additionally it is advised to plug the CAN-Analyser USB interface into a self-powering USB hub in order not to damager the host PC's USB ports in case of unexpected system failure. This scenario is quite unrealistic, since the CAN-Analyser USB interface has been tested thoroughly, but worthing noting.

---

[14]Sidenote: It is possible to plug in the CAN-Analyser USB interface while the system is powered on (or even running), but it is not advised. System failure or damage could be the result.

3.4 Quickstart Guide

4 Observing a System

5 Inspecting the Aggregated Data

6 Controlling a System