

# CANblue II

Extended User Manual

## USER MANUAL

4.01.0126.20000 3.4 en-US ENGLISH



---

# Important User Information

## Disclaimer

The information in this document is for informational purposes only. Please inform HMS Industrial Networks of any inaccuracies or omissions found in this document. HMS Industrial Networks disclaims any responsibility or liability for any errors that may appear in this document.

HMS Industrial Networks reserves the right to modify its products in line with its policy of continuous product development. The information in this document shall therefore not be construed as a commitment on the part of HMS Industrial Networks and is subject to change without notice. HMS Industrial Networks makes no commitment to update or keep current the information in this document.

The data, examples and illustrations found in this document are included for illustrative purposes and are only intended to help improve understanding of the functionality and handling of the product. In view of the wide range of possible applications of the product, and because of the many variables and requirements associated with any particular implementation, HMS Industrial Networks cannot assume responsibility or liability for actual use based on the data, examples or illustrations included in this document nor for any damages incurred during installation of the product. Those responsible for the use of the product must acquire sufficient knowledge in order to ensure that the product is used correctly in their specific application and that the application meets all performance and safety requirements including any applicable laws, regulations, codes and standards. Further, HMS Industrial Networks will under no circumstances assume liability or responsibility for any problems that may arise as a result from the use of undocumented features or functional side effects found outside the documented scope of the product. The effects caused by any direct or indirect use of such aspects of the product are undefined and may include e.g. compatibility issues and stability issues.

---

# Table of Contents

Page

<b>1</b>	<b>User Guide .....</b>	<b>5</b>
1.1	Target Audience.....	5
1.2	Related Documents .....	5
1.3	Document History .....	5
1.4	Conventions.....	6
<b>2</b>	<b>Safety Instructions .....</b>	<b>7</b>
2.1	Information on EMC .....	7
2.2	General Safety Instructions .....	7
2.3	Bluetooth® Connection.....	8
2.4	Intended Use.....	8
<b>3</b>	<b>Scope of Delivery .....</b>	<b>8</b>
<b>4</b>	<b>Product Description .....</b>	<b>9</b>
4.1	Operation Modes.....	9
4.2	Features .....	9
<b>5</b>	<b>Installation.....</b>	<b>10</b>
5.1	Installing the Software .....	10
5.1.1	Installing the Driver .....	10
5.1.2	Installing the CANblue II Software Package.....	10
5.2	Connectors .....	10
5.2.1	Power Connector .....	10
5.2.2	External Antenna .....	11
5.2.3	CAN Connector.....	11
5.3	Installing the Virtual COM Port .....	12
<b>6</b>	<b>Configuration as PC Interface with VCI Driver.....</b>	<b>15</b>
<b>7</b>	<b>Configuration as Generic PC Interface or as Bridge .....</b>	<b>16</b>
7.1	Configuration Tools .....	16
7.1.1	Terminal Program.....	16
7.1.2	CANblueCon Configuration Tool.....	16
7.1.3	Examples.....	18
7.2	Configuring an Interface.....	18
7.3	Configuring a Bridge .....	20
7.3.1	Bridge Chain .....	21
7.4	Settings in Generic Mode .....	22
7.4.1	Configuring the Filter.....	22
7.4.2	Autostart .....	22

7.4.3	Changing the Message Format .....	22
7.4.4	Setting the Transmitting Time .....	23
7.4.5	Reset to Factory Settings .....	23
7.4.6	Changing the Bluetooth Passkey .....	24
7.4.7	Visibility .....	24
7.4.8	Connection Security in Bridge Setup.....	24
<b>8</b>	<b>Operation.....</b>	<b>25</b>
8.1	Overview .....	25
8.2	Indicators .....	25
8.2.1	Mode LED .....	25
8.2.2	CAN LED .....	25
8.2.3	Bluetooth LED .....	25
8.3	Connection Behavior .....	25
<b>9</b>	<b>Errors and Troubleshooting .....</b>	<b>26</b>
<b>10</b>	<b>PC Interface with VCI Driver Network and Device Communication.....</b>	<b>28</b>
<b>11</b>	<b>Generic Mode Network and Device Communication.....</b>	<b>28</b>
11.1	ASCII Protocol.....	28
11.2	CAN Commands.....	29
11.2.1	Configuring the Communication Behavior.....	29
11.2.2	Initializing the CAN Controller .....	32
11.2.3	Configuring the Filter.....	33
11.2.4	Starting the CAN Controller .....	36
11.2.5	Stopping the CAN Controller.....	36
11.2.6	Reset the CAN Controller.....	37
11.3	Device Commands.....	38
11.3.1	Requesting Device Information.....	38
11.3.2	MAC Commands for Connecting Devices .....	40
11.3.3	MAC Commands Security.....	42
11.3.4	Configuring the Device.....	45
11.3.5	Reset the Device .....	49
11.4	CAN Messages in ASCII format.....	50
11.5	CAN Messages in Binary Format .....	51
11.6	Error Messages.....	52
<b>12</b>	<b>Technical Data .....</b>	<b>53</b>
<b>13</b>	<b>Default Settings .....</b>	<b>53</b>
<b>14</b>	<b>Support/Return Hardware.....</b>	<b>54</b>
14.1	Support .....	54
14.2	Return Hardware .....	54

---

<b>15 Disposal</b>	<b>54</b>
<b>A Regulatory Compliance</b>	<b>55</b>
A.1 EMC Compliance (CE)	55
A.2 FCC Compliance Statement	55
A.3 RoHs Directive	56
A.4 Japan Radio Equipment Compliance (TELEC)	56
<b>B Disposal and recycling</b>	<b>57</b>
<b>C Measurements</b>	<b>58</b>
<b>D Configuration Examples</b>	<b>59</b>
D.1 Example 1: Connecting a CAN Network With a Computer	59
D.2 Example 2: Configuring a CAN Bridge	60
D.3 Example 3: Configuring a Bridge Chain	62

**This page intentionally left blank**

# 1 User Guide

Please read the manual carefully. Make sure you fully understand the manual before using the product.

## 1.1 Target Audience

This manual addresses trained personnel who are familiar with CAN technology, Bluetooth® wireless technology and the applicable national standards. The contents of the manual must be made available to any person authorized to use or operate the product.

## 1.2 Related Documents

Document	Author
Installation Guide <i>VCI Driver</i>	HMS
User Manual of bus monitor in use	HMS
VCI Software Design Guides (.NET, C, C++)	HMS

## 1.3 Document History

Version	Date	Description
3.0	March 2017	Edited and revised in new design.
3.1	September 2018	Removed instructions for Windows XP, added bus off information, structural changes in configuration chapters, added target audience and intended use
3.2	October 2018	Added commands for Slave MAC address list
3.3	April 2019	Corrections appendix C, layout changes
3.4	August 2019	New disclaimer, minor changes

## 1.4 Conventions

Instructions and results are structured as follows:

- ▶ instruction 1
- ▶ instruction 2
  - result 1
  - result 2

Lists are structured as follows:

- item 1
- item 2


**Bold typeface** indicates interactive parts such as connectors and switches on the hardware, or menus and buttons in a graphical user interface.

```
This font is used to indicate program code and other kinds of data input/output such as configuration scripts.
```

This is a cross-reference within this document: [Conventions, p. 6](#)


This is an external link (URL): [www.hms-networks.com](http://www.hms-networks.com)

Safety advice is structured as follows:


	<p>Cause of the hazard!</p> <p>Consequences of not taking remediate action.</p> <p>How to avoid the hazard.</p>
---	---


Safety signs and signalwords are used dependent on the level of the hazard.


---

 *This is additional information which may facilitate installation and/or operation.*

---

	<p>This instruction must be followed to avoid a risk of reduced functionality and/or damage to the equipment, or to avoid a network security risk.</p>
---	--

	<p><b>Caution</b></p> <p>This instruction must be followed to avoid a risk of personal injury.</p>
---	--

	<p><b>WARNING</b></p> <p>This instruction must be followed to avoid a risk of death or serious injury.</p>
---	--



## 2 Safety Instructions



Risk of disturbances and interferences if used with WLAN at the same time!  
Bluetooth® wireless technology and WLAN both work with the frequency of 2.4 GHz.



### Caution

This equipment emits RF energy in the ISM (Industrial, Scientific, Medical) band. Make sure that all medical devices used in proximity to this device meet appropriate susceptibility specifications for this type of RF energy.

The CANblue II contains a small radio transmitter and receiver. During communication with other Bluetooth products the CANblue II receives and transmits electromagnetic fields (microwaves) in the frequency range 2.4 to 2.5 GHz. The output power of the radio transmitter is very low. The exposure to transmitted RF energy while using the device is well below the prescribed limits in all national and international RF safety standards and regulations.

### 2.1 Information on EMC



Risk of interference to radio and television if used in office or home environment!  
Use exclusively included accessories. Use exclusively shielded cables.  
Make sure that the shield of the interface is connected with the device plug and the plug on the other side.

### 2.2 General Safety Instructions

- ▶ Protect product from moisture and humidity.
- ▶ Protect product from too high or too low temperature (see [Technical Data, p. 53](#)).
- ▶ Protect product from fire.
- ▶ Do not paint the product.
- ▶ Do not modify or disassemble the product. Service must be carried out by HMS Industrial Networks.
- ▶ Store products in dry and dust-free place.

## 2.3 Bluetooth® Connection

Make sure that the following conditions are met:

- preferably unobstructed line of sight between the antennas of the devices
- minimum distance of 50 cm between the devices (to avoid interference)
- minimum distance of 10 m to WLAN recommended

Data transmission rate depends on:

- distance between the communicating devices
- obstacles between the devices
- environment (texture of walls etc.)
- device configuration
- signal conditions

## 2.4 Intended Use

The CANblue II is used to connect computer systems (like PC, notebook, tablet or smartphone) to CAN networks via Bluetooth® wireless technology.

# 3 Scope of Delivery

Included in the scope of delivery:

- CANblue II
- CANanalyser Mini
- Installation Guide *VCI Driver*
- User Manual *CANblue II*
- CD with VCI driver and extended User Manual

## 4 Product Description

With the CANblue II multiple CAN networks can be connected wireless via **Bluetooth®** wireless technology. The CANblue II forwards from the CAN network received messages to the Bluetooth connection. The messages that are received via a Bluetooth connection are transmitted to the CAN network and other existing Bluetooth connections.

The CANblue II provides an additional server. This connection can be used to configure the CANblue II. Various operation modes are supported.

### 4.1 Operation Modes

#### PC interface

- **VCI driver for Windows**
  - supported by the VCI driver
  - operation with all Ixxat tools possible
  - operation with other VCI-based application programs and tools possible
- **Generic mode (ASCII/binary protocol )**
  - communication based on ASCII commands and optimized binary data transfer
  - usable in all systems, for example embedded computer systems
  - low latency

#### Bridge mode

- several CANblue II can be connected
- CANblue II can serve as Master and Slave
- transparent message exchange on layer 2
- can be used in DeviceNet, CANopen, J1939 and customer specific protocols
- use of CAN ID filters possible

### 4.2 Features

- Bluetooth® specification Bluetooth v4.0
- power supply 9 to 30 V DC
- ISO 11898-2 CAN bus coupling (9 pin D-Sub 9)
- available with internal or external antenna
- different external antennas available
- CAN controller initialization with automatic baud rate detection
- CAN message filtering

## 5 Installation



Connection issues if the computer turns into sleep mode!

Deactivate the sleep mode of the computer the CANblue II is connected to. In case of reconnecting problems see [Errors and Troubleshooting, p. 26](#).

### 5.1 Installing the Software

#### 5.1.1 Installing the Driver

For the operation of the CANblue II as VCI PC interface for Windows the VCI driver is needed.

- ▶ Install the VCI driver (see Installation Guide *VCI Driver*).

#### 5.1.2 Installing the CANblue II Software Package

- ▶ Close all open applications.
- ▶ Make sure that all prior versions of the CANblue II software package are uninstalled.
- ▶ Insert the CD-ROM in the CD drive.
- ▶ Run *CANblue\_II\_Generic\_Setup.exe*.
- ▶ Follow the instructions in the installation program.

### 5.2 Connectors

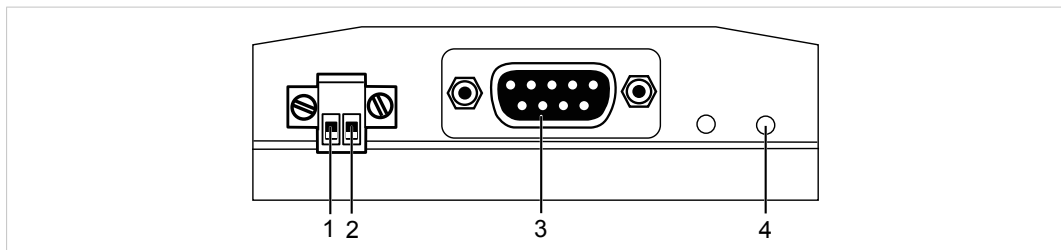


Fig. 1 Connectors

1	Power connector +
2	Power connector -
3	CAN connector
4	Button <b>Reset to factory settings</b>

#### 5.2.1 Power Connector

The device is protected against polarity reversal.

Pin Allocation		
Number	Pin allocation	Signal
1	+	9 to 30 V DC
2	-	GND

## 5.2.2 External Antenna

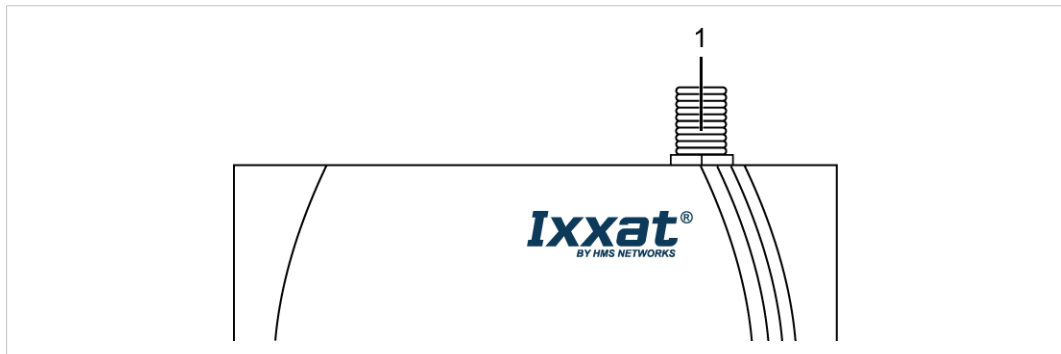


Fig. 2 Connector for external antenna

- ▶ Screw the external antenna on connector (1).
- ▶ Use exclusively antennas that are approved by HMS Industrial Networks (by reason of radio certification).
- ▶ For further information about different antennas see [www.ixxat.com](http://www.ixxat.com).

## 5.2.3 CAN Connector

Pin Allocation of D-Sub 9 Connector

Pin no.	Signal
1	–
2	CAN low
3	GND
4	–
5	–
6	–
7	CAN high
8	–
9	–

## 5.3 Installing the Virtual COM Port

The CANblue II provides two virtual servers: Config and SPP. For the configuration of the CANblue II a Bluetooth-capable device that supports the serial port profile (SPP) must be connected to the Config server via a virtual COM port.

### Windows 7, 8 and 10

- ▶ In Windows task bar right-click on the Bluetooth icon and select **Add a device**.
  - All available devices are displayed.
  - CANblue II devices are named *Ixxat CANblue II ([MAC address])*.

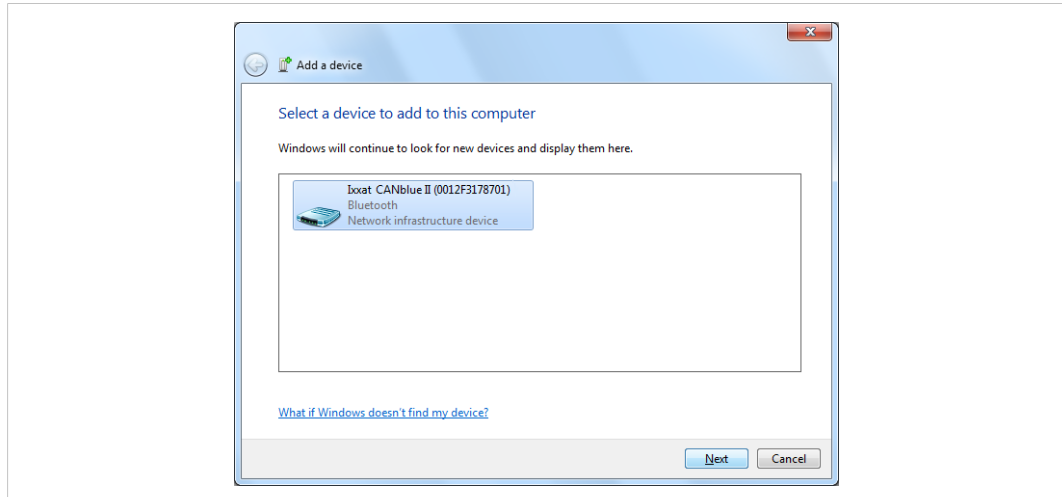


Fig. 3 Add a device

- ▶ Check the MAC address of the CANblue II that is printed on the back of the device.
- ▶ Select the device to connect and click button **Next**.

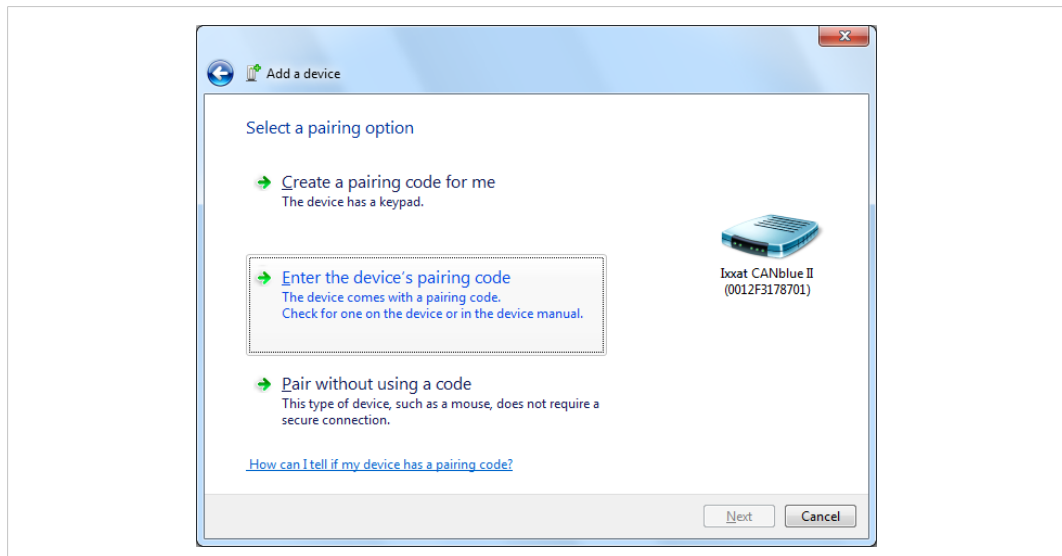


Fig. 4 Add a device

- ▶ Select **Enter the device's pairing code** and click button **Next**.

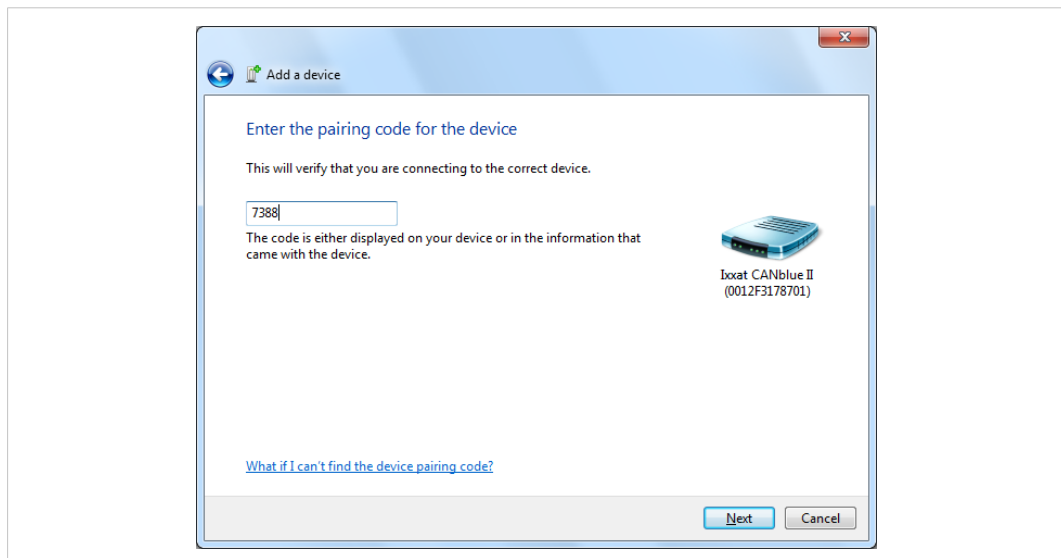


Fig. 5 Pairing code

- ▶ Enter the default pairing code **7388** and click button **Next**.
  - Added device is displayed in window **Devices and Printers**.



*Some Bluetooth drivers do not ask for a pairing code. In this case pairing is possible without code.*

#### Determine the correct COM port:

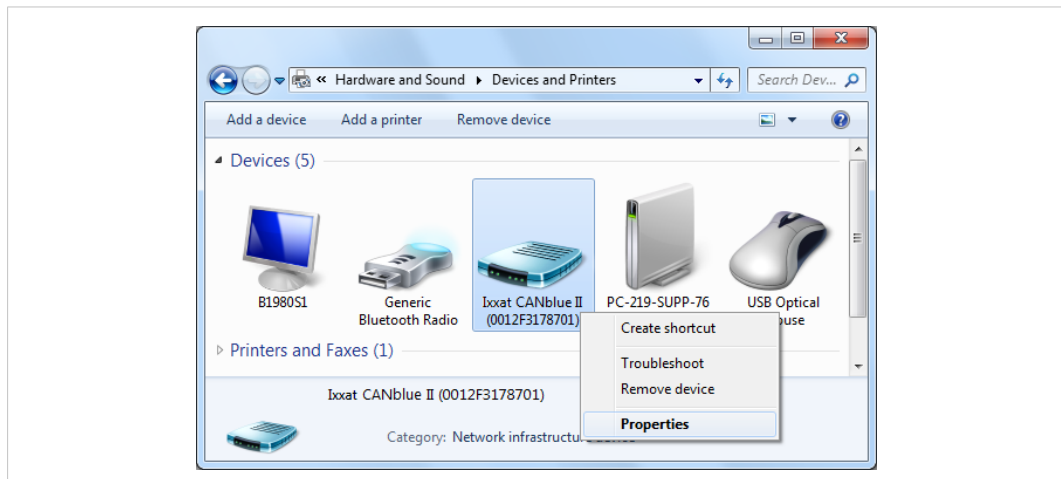
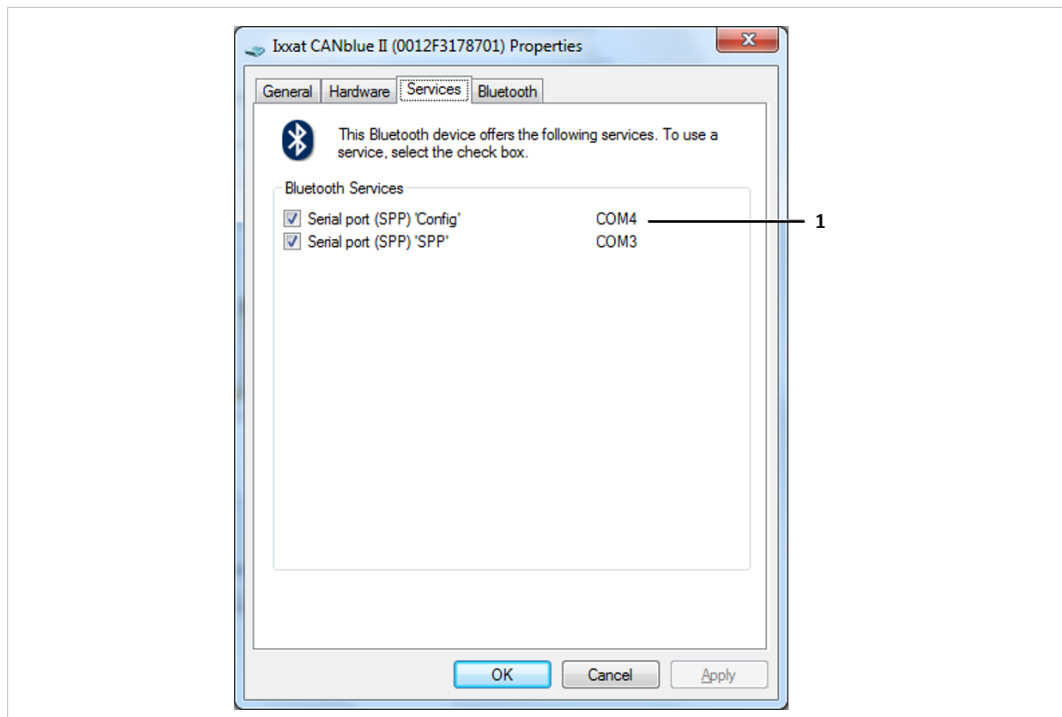


Fig. 6 Devices and printers

- ▶ In window **Devices and Printers** right-click on the newly added CANblue II and in the context menu select **Properties**.

→ Window **CANblue II Properties** is opened.



**Fig. 7** CANblue II properties

→ Two SPP server of the device are displayed.

→ With the displayed COM port of **Serial port (SPP) 'Config'** (1) a connection to the CANblue II can be established.

→ The COM port of **Serial port (SPP) 'SPP'** is reserved for a connection between two CANblue II devices.

- ▶ Make sure that both checkboxes of **Serial port (SPP) 'Config'** and of **Serial port (SPP) 'SPP'** are activated.



*If the checkboxes are not activated the driver may not be correctly installed. To download the driver, make sure that an internet connection is established.*

---

- ▶ Click button **Apply**.

→ The COM port of **Serial port (SPP) 'Config'** can be used to connect to the CANblue II



## 6 Configuration as PC Interface with VCI Driver

The CANblue II can be configured as a PC interface with the VCI driver for Windows.



*HMS recommends to reset the device to factory settings for optimal performance.*

---



*Parallel usage with bridge mode is possible with reduced receive and transmit performance. Existing CAN filters are cleared in VCI mode and restored when the VCI mode is closed.*

---

- ▶ Make sure that the VCI driver and the CANblue II software package are installed (see [Installing the Software, p. 10](#)).
- ▶ Make sure that the virtual Config COM port is installed (see [Installing the Virtual COM Port, p. 12](#)).
- ▶ Install the hardware according to the instructions in the Installation Guide *VCI Driver*.
- ▶ Make sure to access the **Device Server Control** with administrator rights.
- ▶ Configure the device with a VCI based tool, for example with the canAnalyser Mini (included on delivery CD).
- ▶ To test if the device is connected, check the list of available devices in the canAnalyser Mini.

## 7 Configuration as Generic PC Interface or as Bridge

The CANblue II can be configured as Generic PC interface or as Bridge both with two different configuration tools.

### 7.1 Configuration Tools

To configure the CANblue II a terminal program or the CANblueCon Configuration Tool can be used. Loading of existing configurations (txt- and bat-files) and the use of local commands is only possible with the CANblueCon Configuration Tool.

#### 7.1.1 Terminal Program

- ▶ Make sure that the CANblue II Software package is installed (see [Installing the CANblue II Software Package, p. 10](#)).
- ▶ Make sure that the virtual Config COM port is installed (see [Installing the Virtual COM Port, p. 12](#)).
- ▶ Select the setting **serial** and the correct COM port.
- ▶ Start the terminal program.
- ▶ Activate the local echo.
- ▶ Activate **transmitting of carriage return and linefeed with Enter key** at the end of an entered command.
- ▶ Enter the virtual Config COM port.
  - Device is connected.
- ▶ Configure a generic interface (see [Configuring an Interface, p. 18](#)) or a Bridge (see [Configuring a Bridge, p. 20](#)).
- ▶ Use ASCII commands to configure the device and observe the following:
  - Enter the commands in capital letters.
  - Execute the commands with key **Enter**.
  - See [Generic Mode Network and Device Communication, p. 28](#) for further information about the commands.

#### 7.1.2 CANblueCon Configuration Tool



*Configuration examples for a generic interface and a bridge are included on the delivery CD in the folder CANblueCon Examples. The examples can be loaded with the CANblueCon Configuration Tool.*

---



*bat-files can be started directly from the file.*

*Adjust the COM port in the bat-file with an editor and in bridge configurations adjust the MAC address in the txt-file. To start the bat-file in the CANblueCon Configuration Tool, double-click the bat-file.*

---

- ▶ Make sure that the CANblue II Software package is installed (see [Installing the CANblue II Software Package, p. 10](#)).
- ▶ Make sure, that the virtual Config COM port is installed (see [Installing the Virtual COM Port, p. 12](#)).
- ▶ Start the command line.

- ▶ Enter the path to the *CanBlueCon.exe*.

#### To load an existing configuration:

- ▶ In bridge configurations adjust the MAC address in the txt-file.
- ▶ Enter `CanBlueCon.exe <CONFIG_COM_PORT_NUMBER> <FILE_NAME>` in the command line.
  - Batch mode is started.
  - Commands are read from the configuration file.

#### To define a new configuration:

- ▶ Enter `CanBlueCon.exe <CONFIG_COM_PORT_NUMBER>`.
  - Interactive mode is started.
- ▶ Configure a generic interface (see [Configuring an Interface, p. 18](#)) or a Bridge (see [Configuring a Bridge, p. 20](#)).
- ▶ Use ASCII commands to configure the device and observe the following:
  - Execute the commands with key **Enter**.
  - See [Generic Mode Network and Device Communication, p. 28](#) for further information about the commands.



The CANblueCon configuration tool supports a command history. Scrolling through former commands is possible with the keys **Up** and **Down**.

#### Local Commands

Additionally to the ASCII commands local commands are supported by the CANblueCon. The commands are interpreted locally and allow for example the implementation of cyclic transmission. Local commands are useful if a configuration is planned to be used in Batch mode of the CANblueCon, for example to implement loops or prints on screen.

**Additional Commands with CANblueCon Configuration Tool**

Command	Parameter	Description
#delay	<DELAY_TIME>	Delays the execution for the specified time in seconds.
#goto	<LABEL_NAME>	Continues the execution from the string where the label is defined.
#help	-	Shows a help screen.
#label	<LABEL_NAME>	Defines a label.
#pause	-	Waits until any key is pressed.
#print	<TEXT>	Prints <TEXT> on the display.
#exit	-	Closes the CANblueCon.

### 7.1.3 Examples

CANblue II command and CANblue II reply:

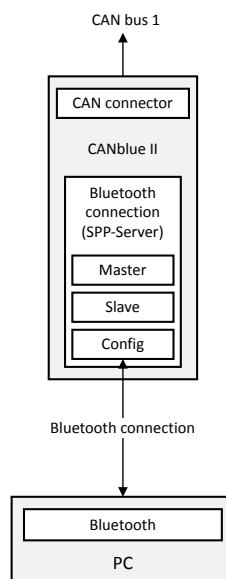
```
>c can_init 1000
I OK: CAN_INIT
```

Local command and local output:

```
>#print CANblue Generic
# CANblue Generic
```

## 7.2 Configuring an Interface

The installed virtual Config COM port is used to configure the CANblue II to exchange data with a CAN network connected to the CANblue II.



- ▶ Make sure, that the software package and the virtual COM port are installed.
- ▶ Start and configure the desired configuration tool (see [Configuration Tools, p. 16](#)).
- ▶ Reset the device to factory settings with command `D SETTINGS_DEFAULT`.
  - Existing filters and settings are deleted.
  - Factory settings are set (information about settings see [Reset to Factory Settings, p. 23](#)).
- ▶ Initialize the CAN controller with the desired baud rate with command `C CAN_INIT <baudrate>`.
- ▶ Set the filter (see [Configuring the Filter, p. 22](#)).
- ▶ Specify further settings (see [Settings in Generic Mode, p. 22](#)).
- ▶ Check the configuration with command `C CONFIG SHOW`.
- ▶ Save the configuration with command `C CONFIG SAVE`.
- ▶ Start the CAN controller with command `C CAN_START`.
  - If the CAN controller receives a message from the CAN network that matches one of the filters, the message is transmitted on the Bluetooth connection in ASCII format.

- ▶ To transmit CAN messages to the CANblue II or into the connected CAN network, use ASCII or binary format (see [Generic Mode Network and Device Communication, p. 28](#)).
  - Transmission format of CAN messages is automatically matched to the received format.

**Example Message**

- ▶ To transmit a CAN data frame with the Standard identifier 7FF and the data bytes *1A 2B 3C 4D 5E 6F 70* to the CAN bus, use command `M SD7 7FF 1A 2B 3C 4D 5E 6F 70`.

## 7.3 Configuring a Bridge

Several Bluetooth devices can be connected as Master and Slave.



Use only CANblue II devices with the same firmware version for a bridge. If CANblue II devices with new firmware version (V2.01.07 and higher) and CANblue II devices with an older firmware version are used in a Bridge contact Ixxat support for information about the compatibility.

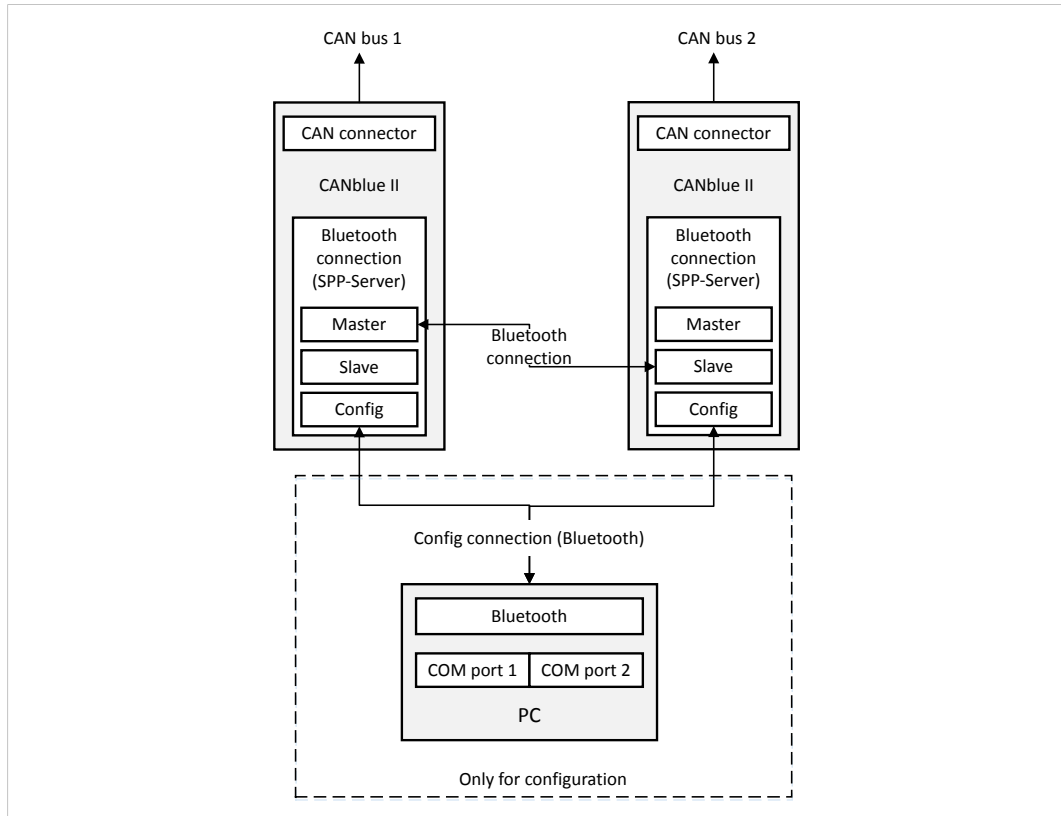


Fig. 8 Configuring a bridge



To simplify the configuration, deactivate the transmission of CAN messages by the Master:

Stop the CAN controller with command `C CAN_STOP` or disable the transmission of CAN messages on the connection with command `C SEND_CAN_FRAMES OFF`.

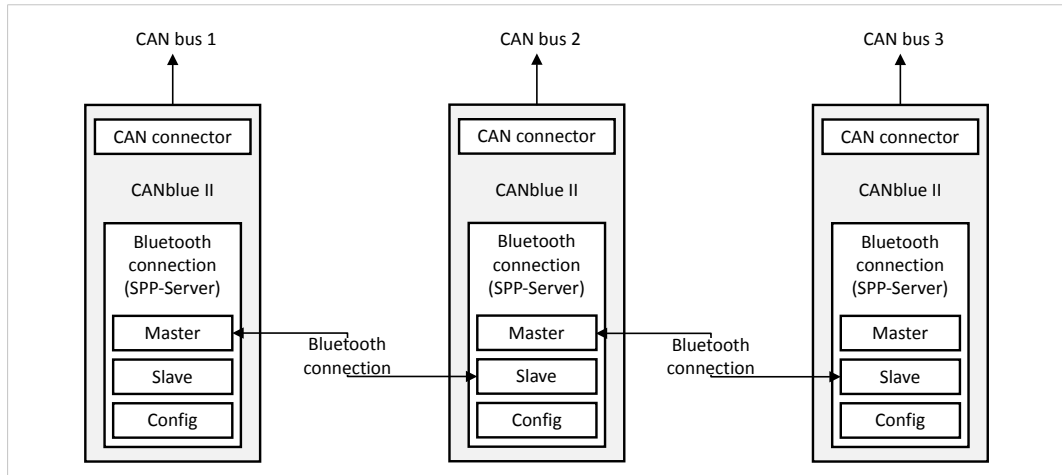
- ▶ Make sure, that the software package and the virtual COM port are installed.
- ▶ Make sure, that the virtual COM ports are installed for all devices and that the connection is established.
- ▶ Start and configure the desired configuration tool (see [Configuration Tools, p. 16](#)).
- ▶ Configure the devices like an interface (see [Configuring an Interface, p. 18](#)).
- ▶ Enable the autostart mode with command `C AUTOSTART ON` on both devices (for further information see [Autostart, p. 22](#)).
- ▶ With the desired Master device enter command `D MAC_ADD <address of slave>`.
  - Device acts as Master and connects to the Slave.
  - Devices start automatically.
  - Devices function as a Bridge between the two CAN networks.
- ▶ Save the configurations on both devices with the commands `C CONFIG SAVE`.

- ▶ To achieve the highest possible data rate between the devices, disconnect the Config connection to the computer.

Since the connection is stored on both devices, devices reconnect automatically after turning off and on and resume to forward CAN messages.

### 7.3.1 Bridge Chain

Configuring a Bridge chain is possible because every Slave can serve as Master for another Slave.



**Fig. 9** Bridge configuration



*Each additional CAN bus increases the rate of CAN messages on the Bluetooth connections and reduces the maximum possible data rate of all connections.*

## 7.4 Settings in Generic Mode

### 7.4.1 Configuring the Filter

Filtering of received messages is possible with the following criteria:

- identifier
- frame format (Extended, Standard)
- frame type (data, remote)

The filter works as a positive filter. CAN messages, with defined criteria in the filter list, that are received by the CAN controller are forwarded to the Bluetooth connection.

Up to 4096 Standard filter entries (includes all possible identifiers of Standard frame format) are supported.

For the Extended filter 300 byte memory are provided. An Extended filter entry occupies 8, 16, 24 or 32 bit, dependent on the number of CAN ID digits. 75 to 300 Extended messages can be filtered.

CAN ID range	Memory consumption in bytes
0–7F	1
80–7FFF	2
8000–7FFFFFF	3
800000–7FFFFFFF	4

For information about the available commands to configure the filter see ASCII commands in [Configuring the Filter, p. 33](#).

### 7.4.2 Autostart

If the autostart mode of the device is enabled and a Bluetooth connection is established, the device attempts to carry out a handshake to start the CAN controller.

- ▶ To enable the autostart mode, use command `C AUTOSTART ON`.
- ▶ Make sure, that the autostart mode is enabled with both devices.

If the Config connection is established:

- ▶ Transmit the response to handshake messages manually.
  - Handshake is concluded.
  - Devices exchange CAN messages in binary format.

### 7.4.3 Changing the Message Format

The message format changes automatically in the following situations:

- With command `C CAN_START` the transmission format is switched to ASCII.
- When the Config connection is used to transmit a CAN message to the device in ASCII format or binary format the device switches to the same format.
- If the device is in autostart mode and a handshake is carried out on the Config connection, the device switches to the binary format.
- ▶ To switch from ASCII format to binary format or to disable the reception of CAN messages, use command `C SEND_CAN_FRAMES` with Config connection.



### 7.4.4 Setting the Transmitting Time

With the standard configuration, messages from the device are collected for up to 4 ms before transmission. The minimum time between the transmission of two consecutive transmission packets can be adjusted.

- ▶ Adjust the time between two transmission packets with command `D BUFF_TIMEOUT`.
  - Transmitting is possible before a Bluetooth SPP packet is filled completely.
  - With timeout 0 the data is transmitted immediately. Protocol overhead is increased.

The size of a packet depends on the other node in the connection. CANblue II devices use data packets of up to 669 bytes between themselves.

### 7.4.5 Reset to Factory Settings

Factory settings:

- controller stopped
- filters deleted
- configuration deleted
- master table deleted
- transmitting time set to 4 ms
- passkey set to 7388
- visibility timeout set to 0



Observe, that preconfigured devices have different factory settings (preconfigured default settings).

#### With Bluetooth Connection (Config)

- ▶ To reset the device to factory settings, use command `D SETTINGS_DEFAULT`.

#### Without Bluetooth Connection

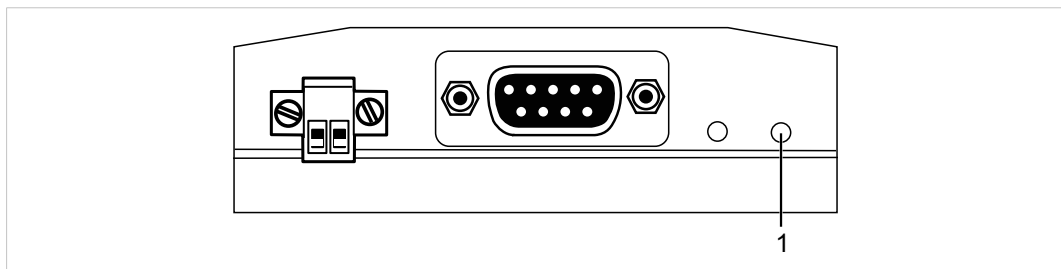


Fig. 10 Button Reset to factory settings

- ▶ Disconnect the device from the power supply.
- ▶ Press and hold button **Reset to factory Settings (1)**.
- ▶ Connect the device to the power supply.
  - CAN LED is flashing red and green.
- ▶ Release button **Reset to factory Settings (1)**.
  - If Mode LED flashes several times, the configuration is reset to factory settings.

### 7.4.6 Changing the Bluetooth Passkey

The default Bluetooth passkey is 7388.

- ▶ Change the passkey with command *D PASSKEY\_SET*.
- ▶ Use character strings with maximally 16 digits.

### 7.4.7 Visibility

It is possible to adjust if the CANblue II is visible and how long it stays visible after connecting to another device.

- ▶ To specify the visibility, use command *D VISIBILITY\_TIMEOUT*.
  - timeout: time the device stays visible after connected to other device
  - timeout 0: always visible

### 7.4.8 Connection Security in Bridge Setup

It is possible to configure the Slave devices to accept only Bluetooth connections from devices which MAC addresses are listed in the Master MAC address list.

- ▶ Add a MAC address to the Master MAC address list of the Slave device with command *D MAC\_MASTER\_ADD*.
  - 10 entries are available
  - 6 Byte hexadecimal

## 8 Operation

### 8.1 Overview

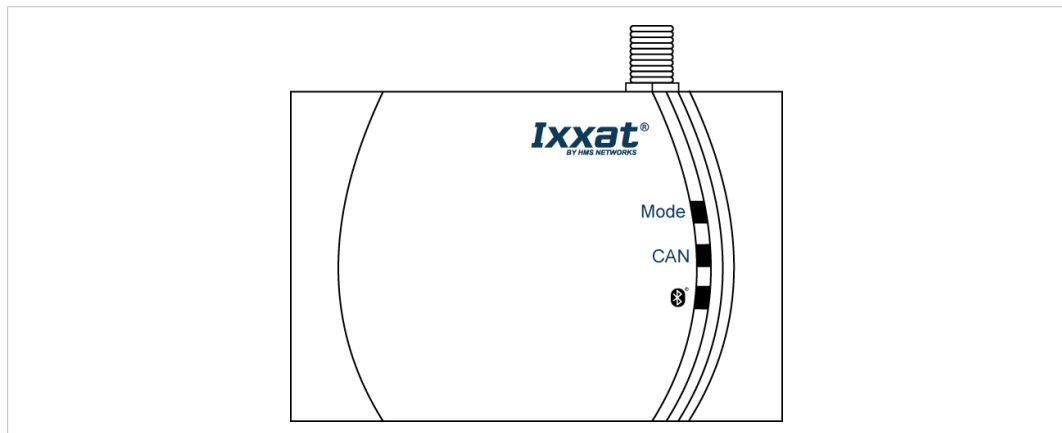


Fig. 11 LED array

### 8.2 Indicators

When the identity of device is requested, all LEDs are blinking.

#### 8.2.1 Mode LED

LED	Description
Red	No Bluetooth MAC address stored in the configuration, no connection to the Slave on the unit

#### 8.2.2 CAN LED

LED	Description
Green flashing	CAN message transmitted or received
Red flashing	CAN message transmitted or received, controller in <i>Warning</i> state
Red	CAN controller in <i>Bus off</i> state

#### 8.2.3 Bluetooth LED

LED	Description
Blue flashing (2 Hz)	Attempt to establish a Bluetooth SPP connection with another device or established connection to the device
Blue flashing (10 Hz)	Data is transmitted or received via Bluetooth SPP.
Blue	Bluetooth SPP connection to another device is established.

### 8.3 Connection Behavior

If a Bluetooth MAC address is stored, the device attempts to establish a Bluetooth connection to that address for 5 seconds. If the connection attempt fails, every 2 seconds a new attempt is started. The loss of a Bluetooth connection is detected after 3 seconds. The Master immediately attempts to establish a new connection. If the Bluetooth connection to the CANblue II is lost, the CAN controller is automatically stopped and the messages in the Tx queue and in the Rx queue are lost. In Bridge mode all messages of all CANblue II devices are lost.

## 9 Errors and Troubleshooting

### Terminal program replies **E 99 Unknown Error** to correctly entered commands.

Commands are not typed in capital letters.

- ▶ In the terminal program enter the commands in capital letters.

### Loss of messages

#### Messages are not forwarded

- Former configuration is stored on the device. Filters are not deleted with initialization.

- ▶ Before configuring the device, reset the device to factory settings or make sure that all filters are deleted or disabled.

#### CAN transmission buffer overflow, oldest buffer entries are overwritten

- CAN controller in *warning* or *bus off* state
- Buffer is full (check queue size with [C CAN\\_INFO](#))
- Entries are overwritten to avoid blocking the reception of data via the Bluetooth connection

- ▶ Reduce the traffic.

#### Bluetooth transmission buffer overflow, incoming messages are discarded

- Too many CAN messages

- ▶ Change the transmission type.  
or
- ▶ Reduce the traffic.

#### CAN receive buffer overflow, CAN messages are discarded, indicated by error message **E 84** (in Config connection)

- Config connection established during high traffic on the connected CAN network
- Connection attempt of CANblue II

- ▶ Before establishing a Config connection reduce the traffic or stop the CAN controller.

### Loss of device responses to commands transmitted on the Config connection

High data traffic between Bluetooth and Config connection, if a command is transmitted via the Config connection, lines of the device response can be lost.

### CAN controller in *warning* state

Multiply incorrectly received or transmitted messages

Stopping and starting the CAN controller does not reset the *warning* state.

- ▶ Reset the device.  
or
- ▶ Make sure that the device receives or transmits several valid CAN messages.

**CAN controller in *bus off* state**

If the controller is in *bus off* state in Bridge mode the messages in the Rx queue are transmitted via a Bluetooth connection to the connected CANblue. The messages in the Tx queue are transmitted via CAN when the CAN connection is restored.

If the Tx queue is full, because of new messages received via the Bluetooth connection during the bus off, the oldest messages are overwritten (Tx queue can store maximally 256 messages).

Bus off recovery is started automatically:

- 5 seconds after the bus off the CAN controller is stopped.
- After 1 second the CAN controller is started.
- If the CAN controller detects 128 occurrences of 11 consecutive bits (e.g. no CAN message transmitted until after 128 occurrences of 11 consecutive bits), all error flags are reset and the CAN controller is in operating state (according to ISO 11898-1).
- Automatic bus off recovery is done until the CAN controller is in *operating* state or stopped via the Config connection.

Manual bus off recovery:

- Stop the CAN controller via the Config connection and restart the CAN controller.

**Connection via the previously used COM port is not possible after restart.**

If the device is disconnected from power while still connected to Windows, the COM port stays occupied.

- ▶ Make sure that the device is disconnected from Windows before disconnecting the device from power.

**CANblue II can not be accessed after the computer was in sleep mode.**

If the computer turns to sleep mode, the Bluetooth connection may not be closed completely. The connection remains and blocks all applications from connecting to the CANblue II.

A restart of the CANblue II does not solve the issue.

If the computer was in sleep mode, and the connection is blocked:

- ▶ Reboot the computer.
- ▶ Stop CANblue II in **Device Server Control**.
- ▶ Stop CANblue II in canAnalyser by clicking the **Stop** button in the Control Panel.
- ▶ Stop all applications which access a CAN controller.
- ▶ In canAnalyser Control Panel remove the CANblue II via **Remove Device**.
- ▶ Reinstall the CANblue II (see Installation Guide *VCI Driver*).

---

## 10 PC Interface with VCI Driver Network and Device Communication

For information about network and device communication with the VCI driver see Software Design Guides (.NET, C, C++) of the VCI.

## 11 Generic Mode Network and Device Communication

To configure and transmit CAN messages via Bluetooth wireless technology an ASCII protocol is defined. To permit a better data rate a binary format is also available for the transmission of the messages. To transmit messages between themselves CANblue II devices use the binary format.

### 11.1 ASCII Protocol

**Structure of ASCII commands:**

Message type	Command	Parameter 1	...	Parameter n	LF or CR-LF
--------------	---------	-------------	-----	-------------	-------------

**Basic rules of the ASCII protocol:**

- Individual fields are separated by blanks.
- Multiple sequential blanks are considered as single blank.
- No distinction between capital and lower-case letters
- Messages are terminated with ASCII linefeed control code (LF or \n) or with carriage return and linefeed (CR LF or \r\n).
- ASCII messages that are transmitted by the device are terminated with the same ASCII control codes as ASCII messages that are transmitted by the user. If no ASCII message is transmitted by the user, the device uses CR LF as terminator.

The following message types (type defined by first byte) are supported:

- CAN commands (C)
- Device commands (D)
- CAN messages in ASCII format (M)
- CAN messages in binary format (X)
- Info messages (I)
- Error messages (E)

#### Examples

ASCII command	Response from the device
C CAN_INIT 250	I OK: CAN_INIT
C CAN_START	I OK: CAN_START
C FILTER_ADD EXT 7FA1 RTR	I OK: FILTER_ADD
C SETTINGS_DEFAULT	I OK: SETTINGS_DEFAULT

## 11.2 CAN Commands

The commands are used to control the CAN controller on the device and to modify the filter settings.

Valid order of usage:

- ▶ Initialize the CAN controller.
- ▶ Configure the filter.
- ▶ Start the CAN controller.
- ▶ Stop the CAN controller.

### 11.2.1 Configuring the Communication Behavior

#### C CAN\_INFO

Shows information about:

- current status of the CAN controller
- software queues (Overrun flags are cleared after response is transmitted.)
- TX queue size
- transmitted CAN messages since the last connection was established (TX counter is a WORD value and starts from zero when 65535+1, possible to implement TX handshake, if difference between local TX counter and CANblue II TX counter is calculated)

C CAN\_INFO

#### Possible Return Values

Return Value	Description
I CAN started I Tx queue size: 256 I Tx counter: 25 I OK: CAN_INFO	State of CAN controller, Tx queue size in number of messages, number of transmitted messages
I CAN stopped I Tx queue size: 256 I Tx counter: 0 I OK: CAN_INFO	State of CAN controller, Tx queue size in number of messages, number of transmitted messages
I CAN controller in WARNING LEVEL I Rx CAN controller OVERRUN I Rx SW queue OVERRUN I Tx SW queue OVERRUN I Tx pending I OK: CAN_INFO	CAN controller in <i>warning level</i> state
I CAN controller in BUS OFF I Rx CAN controller OVERRUN I Rx SW queue OVERRUN I Tx SW queue OVERRUN I Tx pending I OK: CAN_INFO	CAN controller in <i>bus off</i> state

## C CONFIG

It is possible to save, load and show the configuration.

```
C CONFIG <operation>
```

### Parameter

Parameter	Description
<i>operation</i>	Possible entries: SAVE: Saves the current configuration, can take several seconds. LOAD: Loads the existing configuration. SHOW: Shows the configuration.

### Example

```
C CONFIG SHOW
```

### Possible Return Values

#### SAVE

Return Value	Description
I OK: CONFIG SAVE	Function succeeded

#### LOAD

Return Value	Description
I OK: CONFIG LOAD	Function succeeded

#### SHOW

Return Value	Description
I BT0=0, BT1=14 (1000 kBaud)	Values of bus timing registers, name of configuration in brackets
I Bus coupling: HIGH	Bus coupling, exclusively HIGH supported
I Autostart: ON	Autostart mode ON/OFF
I STD filter list I CAN Id: 1 I CAN Id: 4, RTR bit set I STD filter enabled	Content of Standard filter list
I EXT filter list: I CAN Id: 4, RTR bit set I CAN Id: 7FFFF I EXT filter disabled	Content of Extended filter list
I MAC-Slave: 001122334455 Can-Bluet.- form.: binary, State: disconnected	Information about the connection: MAC address, format of CAN messages (ASCII, BINARY, OFF), connection status (connected, disconnected)
I MAC-Master: C44619F9813A Can-Bluet.- form.: off, State: connected	Information about the Master
I TX-Buff. timeout: 0	Timeout value of transmitting buffer
I Passkey: 7388	Bluetooth passkey
I Visibility: 0	Bluetooth visibility
I MAC-Master List:	List with Master MAC IDs
I MAC-Slave List:	List with Slave MAC IDs
I OK: CONFIG SHOW	Function succeeded

Error	Description
E 63 Error while saving config	Error occurred during saving of the configuration. Configuration is lost.
E 61 No valid config	No valid configuration to load.



**C SEND\_CAN\_FRAMES**

Enables or disables the transmission of CAN messages in the directions the command comes from and sets the message format.

```
C SEND_CAN_FRAMES <mode>
```

**Parameter**

Parameter	Description
<i>mode</i>	Message format for transmitting via Bluetooth wireless technology, possible entries: ASCII, BINARY, OFF

**Example**

```
C SEND_CAN_FRAMES ASCII
```

**Possible Return Values**

Return Value	Description
OK: SEND_CAN_FRAMES	Function succeeded

## 11.2.2 Initializing the CAN Controller

### C CAN\_INIT

Initializes the CAN controller with the specified baud rate. Exclusively CiA standard baud rates are supported (10, 20, 50, 100, 125, 250, 500, 800, 1000 kBaud). For custom baud rates see [C CAN\\_INIT\\_CUSTOM](#).

```
C CAN_INIT <baud-rate><buscop>
```

#### Parameter

Parameter	Description
<i>baud-rate</i>	Baud rate in kBaud. CAN controller is initialized with the specified baud rate. Possible values: 10–1000 decimal (exclusively CiA standard)
<i>buscop</i>	Mode of bus coupling, exclusively HIGH is supported, possible entry: HIGH

#### Example

```
C CAN_INIT 500 HIGH
```

#### Possible Return Values

Return Value	Description
I OK: CAN_INIT	Function succeeded
E 22 Baudrate not supported	Baud rate is not supported. Use CiA supported baud rate.
E 31 Error while initializing CAN	Internal error while initializing the CAN controller. CAN controller not initialized. Try to initialize again.
E 4 Unsupported parameter	Bus coupling LOW is not supported. Use bus coupling HIGH.

### C CAN\_INIT\_AUTO

Initializes the CAN controller with automatic baud rate detection. CAN controller is set into TX passive mode and all CiA baud rates are tested until a valid CAN message is received. The CAN controller is initialized with the detected baud rate and a response with the same baud rate is transmitted.

```
C CAN_INIT_AUTO <timeout><buscop>
```

#### Parameter

Parameter	Description
<i>timeout</i>	Time in seconds to test for the CiA baud rate, possible values: 1–1000 decimal (optional, default: 1)
<i>buscop</i>	Mode of bus coupling, exclusively HIGH is supported, possible entry: HIGH

#### Example

```
C CAN_INIT_AUTO 10 HIGH
```

#### Possible Return Values

Return Value	Description
I 100 I OK: CAN_INIT_AUTO	Recognized baud rate 100 kBaud
E 23 Baudrate not detected	No valid baud rate detected within the specified timeout. Maximum response time is 10 times of timeout value.
E 4 Unsupported parameter	Bus coupling LOW is not supported. Use bus coupling HIGH.

**C CAN\_INIT\_CUSTOM**

Initializes the CAN controller with custom baud rate. Parameters *bt0* and *bt1* correspond to the bus timing register of Phillips SJA 1000 CAN controller with a clock frequency of 16 MHz.

Bit 7 of parameter *bt1* is ignored, because the CANblue II CAN controller does not support different sample rates.

```
C CAN_INIT_CUSTOM <bt0><bt1><buscop><name>
```

**Parameter**

Parameter	Description
<i>bt0</i>	SJA1000, bit timing register 0, possible values: 0–FF hexadecimal
<i>bt1</i>	SJA1000, bit timing register 1, possible values: 0–FF hexadecimal
<i>buscop</i>	Mode of bus coupling, exclusively HIGH is supported, possible entry: HIGH
<i>name</i>	String enclosed in "", max. 30 characters. Name of bus timing configuration that is used for command C CONFIG SHOW. If no name is specified, the baud rate is used as name.

**Example**

```
C CAN_INIT_CUSTOM 0 1C HIGH 1000KBAUD CUSTOM
```

**Possible Return Values**

Return Value	Description
I OK: CAN_INIT_CUSTOM	Function succeeded
E 31 Error while initializing CAN	Internal error while initializing the CAN controller. CAN controller not initialized. Try to initialize again.
E 4 Unsupported parameter	Bus coupling LOW is not supported. Use bus coupling HIGH.

**11.2.3 Configuring the Filter****C FILTER\_ADD**

Adds a filter entry to the filter list. The filter works as a positive filter. Received messages which are in the list are forwarded. Messages received via the Bluetooth connection are not filtered.

For information about space and used memory see [Configuring the Filter, p. 22](#).

```
C FILTER_ADD <msg_typ><id><rtr>
```

**Parameter**

Parameter	Description
<i>msg_typ</i>	Message type of filter entry (Standard or Extended), possible values: STD/EXT
<i>id</i>	CAN ID of filter entry, possible values: 0-7FF (Standard), 0-1FFFFFF (Extended)
<i>rtr</i>	Data or remote frame, possible entries: DATA/RTR (optional, default: DATA)

**Example**

```
C FILTER_ADD STD 3A RTR
```

**Possible Return Values**

Return Value	Description
I OK: FILTER_ADD	Function succeeded
E 41 Error adding ID to filter	Out of memory for extended filter elements.

**C FILTER\_REMOVE**

Removes a filter entry from the filter list.

```
C FILTER_REMOVE <msg_typ><id><rtr>
```

**Parameter**

Parameter	Description
<i>msg_typ</i>	Message type of filter entry (Standard or Extended), possible entries: STD/EXT
<i>id</i>	CAN ID of filter entry, possible values: 0-7FF (Standard), 0-1FFFFFF (Extended)
<i>rtr</i>	Data or remote frame, possible entries: DATA/RTR (optional, default: DATA)

**Example**

```
C FILTER_REMOVE STD 3A RTR
```

**Possible Return Values**

Return Value	Description
I OK: FILTER_REMOVE	Function succeeded

**C FILTER\_CLEAR**

Clears the Standard filter list or the Extended filter list.

```
C FILTER_CLEAR <id-tyt>
```

**Parameter**

Parameter	Description
<i>id-tyt</i>	Message type of filter entry (Standard or Extended), possible entries: STD/EXT

**Example**

```
C FILTER_CLEAR EXT
```

**Possible Return Values**

Return Value	Description
I OK: FILTER_CLEAR	Function succeeded

**C FILTER\_ENABLE**

Enables a Standard filter list or a Extended filter list. Messages are forwarded if the ID is found in the filter list. Filter lists for Standard IDs and for Extended IDs must be enabled separately.

```
C FILTER_ENABLE <id-ty>
```

**Parameter**

Parameter	Description
<i>id-ty</i>	Message type of filter entry (Standard or Extended), possible entries: STD/EXT

**Example**

```
C FILTER_ENABLE EXT
```

**Possible Return Values**

Return Value	Description
OK: FILTER_ENABLE	Function succeeded

**C FILTER\_DISABLE**

Disables a Standard filter list or a Extended filter list. Filter lists for Standard IDs and for Extended IDs must be disabled separately.

```
C FILTER_DISABLE <id-ty>
```

**Parameter**

Parameter	Description
<i>id-ty</i>	Message type of filter entry (Standard or Extended), possible entries: STD/EXT

**Example**

```
C FILTER_DISABLE EXT
```

**Possible Return Values**

Return Value	Description
OK: FILTER_DISABLE	Function succeeded

## 11.2.4 Starting the CAN Controller

### C CAN\_START

Starts the CAN controller. Message format for transmitting CAN messages over the Bluetooth connection is set to ASCII mode.

```
C CAN_START
```

#### Possible Return Values

Return Value	Description
I OK: CAN_START	Function succeeded
E 32 Error starting CAN	Internal error while initializing the CAN controller. CAN controller not initialized. Try to initialize again.

### C AUTOSTART

Enables or disables the autostart mode (for further information see [Autostart, p. 22](#)).

```
C AUTOSTART <mode>
```

#### Parameter

Parameter	Description
<i>mode</i>	Enable or disable autostart mode, possible entries: ON/OFF

#### Example

```
C AUTOSTART ON
```

#### Possible Return Values

Return Value	Description
I AUTOSTART ON	Autostart activated
I AUTOSTART OFF	Autostart deactivated
I OK: AUTOSTART	Function succeeded

## 11.2.5 Stopping the CAN Controller

### C CAN\_STOP

Stops the CAN controller.

```
C CAN_STOP
```

#### Possible Return Values

Return Value	Description
I OK: CAN_STOP	Function succeeded
E 33 Error stop CAN	Internal error while initializing the CAN controller.

## 11.2.6 Reset the CAN Controller

### C CAN\_RESET

Resets the CAN controller.

```
C CAN_RESET
```

### Possible Return Values

Return Value	Description
I OK: CAN_RESET	Function succeeded

## 11.3 Device Commands

### 11.3.1 Requesting Device Information

#### D VERSION

Gets the firmware version of the CANblue II.

```
D VERSION
```

#### Possible Return Values

Return Value	Description
I CANblue Generic – Bridge v2.01.07 I OK: VERSION	Firmware version of the device

#### D PROTOCOL

Gets the ASCII protocol version.

```
D PROTOCOL
```

#### Possible Return Values

Return Value	Description
I ASCII Extended Protocol v1.2 I OK: PROTOCOL	ASCII protocol version

#### D IDENTIFY

Gets the hardware version number and the name of the CANblue II. The device name contains the Bluetooth MAC address. All LEDs of the CANblue II are flashing.

```
D IDENTIFY
```

#### Possible Return Values

Return Value	Description
I Name: IXXAT CANblue II (1A2B3C4D5E6F) I HW-Number: HW 999999 I OK: IDENTIFY	Name of the device, MAC address in brackets Hardware version number



**D INFO**

Shows information about the configured Bluetooth connection settings and the Bluetooth connections. Additional information like connection quality, receive signal strength or transmission power is shown for each connection.

D INFO

**Possible Return Values**

Return Value	Description
I Link-policy parameter :	Bluetooth connection settings
I Settingname: DEFAULT	Name of configured connection settings (see <a href="#">D LINK_POLICY</a> , p. 47)
I Packettype: CC18	Bluetooth packet types
I PagescanInterval: 800	Page scan interval
I PagescanWindow: 12	Page scan window
I PagescanType: 0	Page scan type
I Latency (wished) : 40	Max. Bluetooth latency in Bluetooth time slots of 625 $\mu$ s
I Tx-Power (max) : 14 dBm	Max. allowed Bluetooth transmission power
I MAC, Latency, Link quality, RSSI, Tx-Power, PacketType	Table of current Bluetooth connections
I 123456789ABC, 40*625us, 100%, 15 dB, 1 dBm, CC18	Table entry of one connection: MAC address, latency in $\mu$ s, connection quality in %, receive signal strength indication in dB (-127 dB to + 128 dB), transmission power in dBm (-18 dBm to +14 dBm), Bluetooth packet types in use
I OK: INFO	Function succeeded

### 11.3.2 MAC Commands for Connecting Devices

#### D MAC\_ADD

Adds a MAC address to the connection list of a CANblue II. The Master device tries to establish a connection to the Bluetooth device with the added MAC address.

```
D MAC_ADD <adr>
```

#### Parameter

Parameter	Description
<i>adr</i>	MAC address of the second CANblue II (Slave), value: 6 Byte hexadecimal

#### Example

```
D MAC_ADD 001122334455
```

#### Possible Return Values

Return Value	Description
I OK: MAC_ADD	Function succeeded
E 51 MAC-list is full	Only one MAC address is supported. Not possible to add another MAC address.
E 53 MAC address already exists	MAC address is already used for a connection to a server.

#### Remark

To add a range of MAC addresses see [D MAC\\_SLAVE\\_ADD](#).

#### D MAC\_REMOVE

Removes a MAC address from the connection list of a CANblue II. An active connection or attempt to establish a connection is closed, when the command is called. This can cause a delayed response up to 5 seconds.

```
D MAC_REMOVE <adr>
```

#### Parameter

Parameter	Description
<i>adr</i>	MAC address of the CANblue II (Slave) to be removed from the list, value: 6 Byte hexadecimal

#### Example

```
D MAC_REMOVE 001122334455
```

#### Possible Return Values

Return Value	Description
I OK: MAC_REMOVE	Function succeeded
E 52 Wrong MAC address	MAC address is invalid or not in the connection list.

**D MAC\_CLEAR**

Removes all MAC addresses from the connection list of a CANblue II. An active connection or attempt to establish a connection is closed, when the command is called. This can cause a delayed response up to 5 seconds.

```
D MAC_CLEAR
```

**Possible Return Values**

Return Value	Description
I OK: MAC_CLEAR	Function succeeded

**D MAC\_SCAN**

Starts a scan for other Bluetooth devices. After the scan time is expired the response lists all active devices with name and Bluetooth MAC address. Due to a device name query the response can be delayed up to 5 seconds for every device. Maximally 10 devices can be listed.

```
D MAC_SCAN <time>
```

**Parameter**

Parameter	Description
<i>time</i>	Scan time in seconds, possible values: 1–255 decimal (optional, default: 10)

**Example**

```
D MAC_SCAN 20
```

**Possible Return Values**

Return Value	Description
I MAC-Address, Name I 001122334455 Device 1 I 010203040506 IXXAT CANblue (010203040506) I 012345678901 Mobile Phone X I 010203040507 IXXAT CANblue II (010203040507) I OK: MAC_SCAN	Function succeeded, list of active devices, name and MAC address in brackets
E 52 Wrong MAC address	MAC address is invalid or not in connection list.

### 11.3.3 MAC Commands Security

#### D MAC\_MASTER\_ADD

Adds a MAC address or a MAC address range to the Master MAC address list. Slave devices then only accept Bluetooth connections from devices which MAC addresses are listed in the Master MAC address list. In the Master MAC address list 10 entries are available.

```
D MAC_MASTER_ADD <adr1> <adr2>
```

#### Parameter

Parameter	Description
<i>adr1</i>	MAC address, value: 6 Byte hexadecimal If <i>adr2</i> is also defined, this address is the start address of the range.
<i>adr2</i>	Optional, end address of the MAC address range, value: 6 Byte hexadecimal

#### Example

```
D MAC_MASTER_ADD 001122334455
```

#### Possible Return Values

Return Value	Description
I OK: MAC_MASTER_ADD	Function succeeded
E 51 MAC-list is full	Only 10 MAC address list entries are supported. Not possible to add another MAC address.
E 52 Wrong MAC address	MAC address is invalid. Valid MAC address consists of 12 digits.
E 53 MAC address already exists	MAC address is already used for a connection to a server.
E 54 Invalid MAC Address range	Address range is invalid. Make sure, that <i>adr2</i> is higher than <i>adr1</i> .

#### D MAC\_SLAVE\_ADD

Adds a MAC address or a MAC address range to the Slave MAC address list. If the Slave MAC address list is not empty, the Master device scans the network and searches for Bluetooth devices. The Master connects to the detected devices, that are in the Slave MAC address list. In the Slave MAC address list 10 entries are available.

```
D MAC_SLAVE_ADD <ADDR1> <ADDR2>
```

#### Parameter

Parameter	Description
<i>ADDR1</i>	MAC address, value: 6 Byte hexadecimal If <i>ADDR2</i> is also defined, this address is the start address of the range.
<i>ADDR2</i>	Optional, end address of the MAC address range, value: 6 Byte hexadecimal

#### Example

```
D MAC_SLAVE_ADD 001122334455
```

#### Possible Return Values

Return Value	Description
I OK: MAC_SLAVE_ADD	Function succeeded
E 51 MAC-list is full	Only 10 MAC address list entries are supported. Not possible to add another MAC address.
E 52 Wrong MAC address	MAC address is invalid. Valid MAC address consists of 12 digits.
E 53 MAC address already exists	MAC address is already used for a connection to a server.
E 54 Invalid MAC Address range	Address range is invalid. Make sure, that <i>ADDR2</i> is higher than <i>ADDR1</i> .

**D MAC\_MASTER\_REMOVE**

Removes a MAC address or a MAC address range from the Master MAC address list. Only entries that match an entry in the Master MAC address list can be removed (the same range as added via [D MAC\\_MASTER\\_ADD](#)).

```
D MAC_MASTER_REMOVE <adr1> <adr2>
```

**Parameter**

Parameter	Description
<i>adr1</i>	MAC address to be removed from list, value: 6 Byte hexadecimal If <i>adr2</i> is also defined, this address is the start address of the range.
<i>adr2</i>	Optional, end address of the MAC address range, value: 6 Byte hexadecimal

**Example**

```
D MAC_MASTER_REMOVE 001122334455
```

**Possible Return Values**

Return Value	Description
I OK: MAC_MASTER_REMOVE	Function succeeded
E 52 Wrong MAC address	MAC address is invalid or not in the Master MAC address list. Valid MAC address consists of 12 digits.

**D MAC\_SLAVE\_REMOVE**

Removes a MAC address or a MAC address range from the Slave MAC address list. Only entries that match an entry in the Slave MAC address list can be removed (the same range as added via [D MAC\\_SLAVE\\_ADD](#)).

```
D MAC_SLAVE_REMOVE <ADDR1> <ADDR2>
```

**Parameter**

Parameter	Description
<i>ADDR1</i>	MAC address to be removed from list, value: 6 Byte hexadecimal If <i>ADDR2</i> is also defined, this address is the start address of the range.
<i>ADDR2</i>	Optional, end address of the MAC address range, value: 6 Byte hexadecimal

**Example**

```
D MAC_SLAVE_REMOVE 001122334455
```

**Possible Return Values**

Return Value	Description
I OK: MAC_REMOVE	Function succeeded
E 52 Wrong MAC address	MAC address is invalid or not in the Slave MAC address list.

**D MAC\_MASTER\_CLEAR**

Removes all MAC addresses from the Master MAC address list. After clearing the list, a Slave device accepts a Bluetooth connection from all devices.

```
D MAC_MASTER_CLEAR
```

**Possible Return Values**

Return Value	Description
I OK: MAC_MASTER_CLEAR	Function succeeded

**D MAC\_SLAVE\_CLEAR**

Removes all MAC addresses from the Slave MAC address list. After clearing the list, a Master device does not automatically search for other Bluetooth devices.

```
D MAC_SLAVE_CLEAR
```

**Possible Return Values**

Return Value	Description
I OK: MAC_SLAVE_CLEAR	Function succeeded

## 11.3.4 Configuring the Device

### D CONFIG

It is possible to save, load and show the configuration.

```
D CONFIG <operation>
```

#### Parameter

Parameter	Description
<i>operation</i>	SAVE: Saves the current configuration, can take several seconds. LOAD: Loads the existing configuration. SHOW: Shows the configuration.

#### Example

```
D CONFIG SHOW
```

#### Possible Return Values

SAVE	
Return Value	Description
I OK: CONFIG SAVE	Function succeeded

LOAD	
Return Value	Description
I OK: CONFIG LOAD	Function succeeded

SHOW	
Return Value	Description
I BT0=0, BT1=14 (1000 kBaud)	Values of bus timing registers. Name of configuration is given in brackets.
I Bus coupling: HIGH	Bus coupling, exclusively HIGH supported
I Autostart: ON	Autostart mode ON/OFF
I STD filter list I CAN Id: 1 I CAN Id: 4, RTRbit set I STD filter enabled	Content of Standard ID filter list
I EXT filter list: I CAN Id: 4, RTRbit set I CAN Id: 7FFFF I EXT filter disabled	Content of Extended ID filter list
I MAC-Slave: 001122334455 Can-Bluet.- form.: binary, State: disconnected	Information about connection: MAC address, format of CAN messages (ASCII, BINARY, OFF), connection status (connected, disconnected)
I MAC-Master: C44619F9813A Can-Bluet.- form.: off, State: connected	Information about Master
I TX-Buff. timeout: 0	Timeout value of transmitting buffer
I Passkey: 7388	Bluetooth passkey
I Visibility: 0	Bluetooth visibility
I MAC-Master List:	List with Master MAC IDs
I MAC-Slave List:	List with Slave MAC IDs
I OK: CONFIG SHOW	Function succeeded

**D PASSKEY\_SET**

Changes the Bluetooth passkey.

```
D PASSKEY_SET <key>
```

**Parameter**

Parameter	Description
<i>key</i>	Bluetooth passkey, up to 16 digits, value: character string. In Bridge configurations use the same passkey for each device.

**Example**

```
D PASSKEY_SET 1234567890ABCD
```

**Possible Return Values**

Return Value	Description
I OK: PASSKEY_SET	Function succeeded
E 13 Wrong data length	Invalid data length is received. Passkey is invalid. Valid passkey consists of maximally 16 digits.

**D VISIBILITY\_TIMEOUT**

Changes the Bluetooth visibility.

```
D VISIBILITY <timeout>
```

**Parameter**

Parameter	Description
<i>timeout</i>	Time in seconds after which the device is invisible to other devices. If the timeout is 0, the device is always visible. If the timeout is unequal 0, the device is invisible after a connection to another device is established or after the timeout is exceeded. Possible values: 0–60000 decimal

**Example**

```
D VISIBILITY 60
```

**Possible Return Values**

Return Value	Description
I OK: VISIBILITY	Function succeeded
E 2 Wrong parameter	Timeout value is out of range.



**D BUFF\_TIMEOUT**

Sets the timeout for the transmitting buffer resp. the time between two consecutive Tx Bluetooth packets of the CANblue II. The timeout is applied to all Bluetooth connections of the device (for further information see [Setting the Transmitting Time, p. 23](#)).

```
D BUFF_TIMEOUT <time>
```

**Parameter**

Parameter	Description
<i>time</i>	Collecting time of Rx CAN message in milliseconds, possible values: 0–1000 decimal

**Example**

```
D BUFF_TIMEOUT 4
```

**Possible Return Values**

Return Value	Description
I OK: BUFF_TIMEOUT	Function succeeded

**D LINK\_POLICY**

Sets the properties of the Bluetooth connection.

```
D LINK-POLICY <conf>
```

**Parameter**

Parameter	Description
<i>conf</i>	<p>Predefined Bluetooth configurations, the selection is applied on all Bluetooth connections to get the best results.</p> <p><b>DEFAULT:</b> Balanced configuration, suitable for more than one connection in parallel and for <i>none</i>-CANblue II devices</p> <p><b>SHORTEST_LATENCY:</b> Reduced latency for Bluetooth messages. Settings also reduces the data rate to approx. 2000 CAN Msg/s per direction. With the setting only one connection per device is possible. If a connection between the devices is established they cannot be found by a Bluetooth scan.</p> <p><b>QUICKEST_CONNECTION:</b> Allows faster establishment of a Bluetooth Bridge. Setting increases power consumption of the device and reduces the data rate.</p> <p><b>MOST_ROBUST_CONNECTION:</b> Allows bridging a long distance and the Bluetooth connection is more insusceptible to disturbances. Setting reduces the data rate to approx. 3000 CAN msg/s per direction.</p>

**Example**

```
D LINK_POLICY SHORTEST_LATENCY
```

**Possible Return Values**

Return Value	Description
I OK: LINK_POLICY	Function succeeded

**D DEVICE\_NAME\_SET**

Changes the CANblue device name. The device name is transferred to the host device when a new configuration is established. Changes must be saved with command [D CONFIG SAVE](#). Observe that changes are applied after a reboot of the device.

```
D DEVICE_NAME_SET <name>
```

**Parameter**

Parameter	Description
<i>name</i>	Alphanumeric string, including special characters (but no blanks). Valid range is 10-16 characters. The MAC ID is automatically added to the device name. In Bridge configurations use the same name for each device (the devices are distinguished by the MAC address). Value * resets the name to the default name.

**Example**

D DEVICE\_NAME\_SET TEST sets the device name TEST(0012F331DA4D).

**Possible Return Values**

Return Value	Description
I OK: DEVICE_NAME_SET	Function succeeded

**Remark**

The factory reset does not reset the name. The name can be read with command [D IDENTIFY](#).

### 11.3.5 Reset the Device

#### D RESET

The device transmits the response and resets itself. Any established Bluetooth connections are lost.

```
D RESET
```

#### Possible Return Values

Return Value	Description
I OK: RESET	Function succeeded

#### D SETTINGS\_DEFAULT

The configuration is reset to the factory default settings. Stored configurations are deleted.

```
D SETTINGS_DEFAULT
```

#### Possible Return Values

Return Value	Description
I OK: SETTINGS_DEFAULT	Function succeeded



*Observe, that preconfigured devices are reset to the preconfigured default settings.*

## 11.4 CAN Messages in ASCII format

CAN Messages coded in ASCII format are called M-type messages.

M-type messages are used to transmit CAN messages over a Bluetooth connection to another device. The receiving device forwards the message to all established Bluetooth connections and if the local CAN controller is started the message is transmitted to the CAN network.



*Remote messages are transmitted without any data bytes, but the value of the data length (DLC) can be a value between 0 and 8.*

```
M FTD ID D0 D1 D2 D3 D4 D5 D6 D7
```

### Parameter

Parameter	Description
<i>FTD</i>	Three characters define the message format: 1. character: frame format ( S — Standard, E — Extended) 2. character: frame type ( D — Data, R — Remote) 3. character: DLC ("0–8" data length)
<i>ID</i>	CAN message identifier Standard: 0–7FF hexadecimal Extended: 0–7FFFFFFF hexadecimal
<i>D0–D7</i>	Data bytes of the message, messages consist of up to 8 data bytes, every byte is separated by a blank. Possible values: 0–FF hexadecimal

### Example

```
M SD4 1A2 11 22 33 4
```

### Possible Return Values

Return value	Description
E 85 Tx SW queue OVERRUN	Overrun of the transmitting queue, for example the CAN controller is in <i>error warning</i> or <i>bus off</i> state or data could not be transmitted fast enough due to slow baud rate.

## 11.5 CAN Messages in Binary Format

CAN Messages coded in binary format are called X-type messages.

Basic features of binary format:

- allows faster transmission of CAN messages
- data of the CAN message is transmitted uncoded in a binary value
- fields are not separated by blanks
- fields are without CR/LF characters

X-type messages are used to transmit CAN messages over a Bluetooth connection to another device. The receiving device forwards the message to all established Bluetooth connections and if the local CAN controller is started the message is transmitted to the CAN network.

### Standard CAN Message

```
X FI ID_HB ID_LB D0 D1 D2 D3 D4 D5 D6 D7
```

### Extended CAN Message

```
X FI ID_HW_HB ID_HW_LB ID_LW_HB ID_LW_LB D0 D1 D2 D3 D4 D5 D6 D7
```

### Parameter

Parameter	Description
<i>FI</i> (bit field)	FF (bit 7): Frame format ( 0 — Standard, 1 — Extended) RTR (bit 6): Frame type ( 0 — Data, 1 — Remote) DLC (bit 0–3): Data length 0–8
<i>ID_HB</i>	High byte of Standard CAN ID (0–FF)
<i>ID_LB</i>	Low byte of Standard CAN ID (0–FF)
<i>ID_HW_HB</i>	High word, high byte of Extended CAN ID (0–1F)
<i>ID_HW_LB</i>	High word, low byte of Extended CAN ID (0–FF)
<i>ID_LW_HB</i>	Low word, high byte of Extended CAN ID (0–FF)
<i>ID_LW_LB</i>	Low word, low byte of Extended CAN ID (0–FF)
<i>D0–D7</i>	Up to 8 data bytes (0–FF)

### Example

```
0x58, 0x85, 0x01, 0x02, 0x03, 0x04, 0x19, 0x2A, 0x3B, 0x4C, 0x5D
```

0x58	X (binary message type)
0x85	FF=1 (Ext); RTR=0 (Data); DLC = 5
0x01020304	ID
0x19, 0x2A, 0x3B, 0x4C, 0x5D	5 data bytes

### Possible Return Values

Error	Description
E 85 Tx SW queue OVERRUN	Overrun of the transmitting queue, for example CAN controller is in <i>error warning</i> or <i>bus off</i> state or data could not be transmitted fast enough due to slow baud rate.

## 11.6 Error Messages

Error message	Description
E 1 Unknown command	Invalid command or message type is received.
E 2 Wrong parameter	Parameter of a command is invalid.
E 3 Unsupported command	Received command is not supported.
E 4 Unsupported parameter	Parameter of a command is not supported.
E 11 Wrong message type	Invalid message type is received (valid: Standard or Extended).
E 12 Wrong frame type	Invalid frame type is received (valid: data or remote).
E 13 Wrong data length	Invalid data length is received (valid: 0–8).
E 14 Wrong message ID	Invalid ID is received (valid: 0–7FF or 0–1FFFFFF).
E 15 Wrong number of data bytes	Number of data bytes does not match the data length.
E 21 Unknown Bus Coupling value	Invalid bus coupling value (valid: high).
E 22 Baudrate not supported	Baud rate is not supported. Use a CiA supported baud rate.
E 23 Baudrate not detected	No valid baud rate detected by automatic baud rate detection within the specified timeout.
E 31 Error while initializing CAN	CAN controller not initialized. Try to initialize again.
E 32 Error starting CAN	CAN controller not started. Try to start again.
E 33 Error stop CAN	CAN controller not stopped. Try to stop again.
E 41 Error adding ID to filter	Out of memory for extended filter elements.
E 51 MAC-list is full	Not possible to add another MAC address.
E 52 Wrong MAC Address	MAC address is invalid (valid: 6 byte hexadecimal) or MAC address is not in MAC address list.
E 53 MAC Address already exists	MAC address is already used for a connection to a server.
E 54 Invalid MAC Address range	Address range is invalid. Make sure, that <i>adr2</i> is higher than <i>adr1</i> .
E 61 No valid config	No valid configuration to load.
E 63 Error while saving config	Error while saving the configuration. Configuration is lost.
E 81 CAN controller in BUS OFF	CAN controller is in <i>bus off</i> state.
E 82 CAN controller in WARNING LEVEL	CAN controller is in <i>error warning</i> state.
E 84 Rx SW queue OVERRUN	One or more consecutive CAN messages are lost because of a software overrun.
E 85 Tx SW queue OVERRUN	One or more consecutive CAN messages are lost before transmitting by the CAN controller because the CAN controller is in <i>bus off</i> or <i>error warning</i> state or because of slow baud rate.
E 91 Can't show more	Not all filter elements are shown. Depending on the free space of the transmitting buffer only a limited amount of filter elements can be shown with the command <code>C CONFIG SHOW</code> .
E 99 Unknown Error	Internal error occurred. No specific error message specified.

## 12 Technical Data

Bluetooth® qualification	v4.0 (Bluetooth® classic)
Output power	11 dBm, internal antenna 13 dBm, external antenna
Bluetooth® output frequency	2.402 to 2.480 GHz, ISM band
CAN transceiver	Texas Instruments SN65HVD251
Max. number of CAN bus nodes	120
Power supply	9 to 30 V DC
Power consumption	Typ. 50 mA at 12 V, max. 100 mA at 12 V
Dimensions	81 x 66 x 26 mm
Weight	Approx. 83 g
Operating temperature	-40 °C to 85 °C
Relative humidity	10 to 95%, non-condensing
CAN interface isolation working voltage	130 V AC/DC (continuous) 1000 V DC (1 second)
External antenna	RP-SMA connector, max. antenna gain 3.4 dBi
Bridge set-up time	Typ. 3 to 4 seconds
Bluetooth transfer delay	Approx. 4 ms (average), CAN — Bluetooth, or Bluetooth — CAN
CAN transmission rate	100% bus load at 1 MBit
Maximal distance between two devices in bridge mode	200 m/650 ft

## 13 Default Settings

Pairing Code	7388
MAC address	Printed on the back of the device

## 14 Support/Return Hardware

Observe the following information in the support area on [www.ixxat.com](http://www.ixxat.com):

- information about products
- FAQ lists
- installation notes
- updated product versions
- updates

### 14.1 Support

- ▶ For problems or support with the product request support at [www.ixxat.com/support](http://www.ixxat.com/support).
- ▶ If required use support phone contacts on [www.ixxat.com](http://www.ixxat.com).

### 14.2 Return Hardware

- ▶ Fill in the form for warranty claims and repair on [www.ixxat.com/support/product-returns](http://www.ixxat.com/support/product-returns).
- ▶ Print out the Product Return Number (PRN resp. RMA).
- ▶ Pack product in a physically- and ESD-safe way, use original packaging if possible.
- ▶ Enclose PRN number.
- ▶ Observe further notes on [www.ixxat.com](http://www.ixxat.com).
- ▶ Return hardware.

## 15 Disposal

- ▶ Dispose of product according to national laws and regulations.
- ▶ Observe further notes about disposal of products on [www.ixxat.com](http://www.ixxat.com).



## A Regulatory Compliance



The Ixxat CANblue II with external antenna port is for OEM integrations only. The end-user product will be professionally installed in such a manner that only the authorized antennas are used. A list of authorized antennas is available from [www.ixxat.com](http://www.ixxat.com).

### A.1 EMC Compliance (CE)



The product is in compliance with the Electromagnetic Compatibility Directive. More information and the Declaration of Conformity is found at [www.ixxat.com](http://www.ixxat.com).

### A.2 FCC Compliance Statement

This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions:

- This device may not cause harmful interference.
- This device must accept any interference received, including interference that may cause undesired operation.

<b>Product name</b>	CANblue
<b>Model</b>	II
<b>Responsible party</b>	HMS Industrial Networks Inc
<b>Address</b>	35 E. Wacker Dr, Suite 1700 Chicago , IL 60601
<b>Phone</b>	+1 312 829 0601



Any changes or modifications not expressly approved by HMS Industrial Networks could void the user's authority to operate the equipment.



This equipment has been tested and found to comply with the limits for a Class B digital device, pursuant to Part 15 of the FCC rules. These limits are designed to provide reasonable protection against harmful interference in a residential installation. This equipment generates, uses and can radiate radio frequency energy and, if not installed and used in accordance with the instructions, may cause harmful interference to radio communications. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause harmful interference to radio or television reception, which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

Reorient or relocate the receiving antenna.

Increase the separation between the equipment and the receiver.

Connect the equipment into an outlet on a circuit different from that to which the receiver is connected.

Consult the dealer or an experienced radio/TV technician for help.

### A.3 RoHs Directive

The product is in compliance with the RoHs Directive 2002/95/EC (Restriction of the use of certain hazardous substances in electrical and electronic equipment).

### A.4 Japan Radio Equipment Compliance (TELEC)

CANblue II uses the cB-0946 module which complies with the Japanese Technical Regulation Conformity Certification of Specified Radio Equipment (ordinance of MPT N°. 37, 1981), Article 2, Paragraph 1, Item 19, "2.4 GHz band wide band low power data communication system". The cB-0946 MIC certification number is 204-210003.



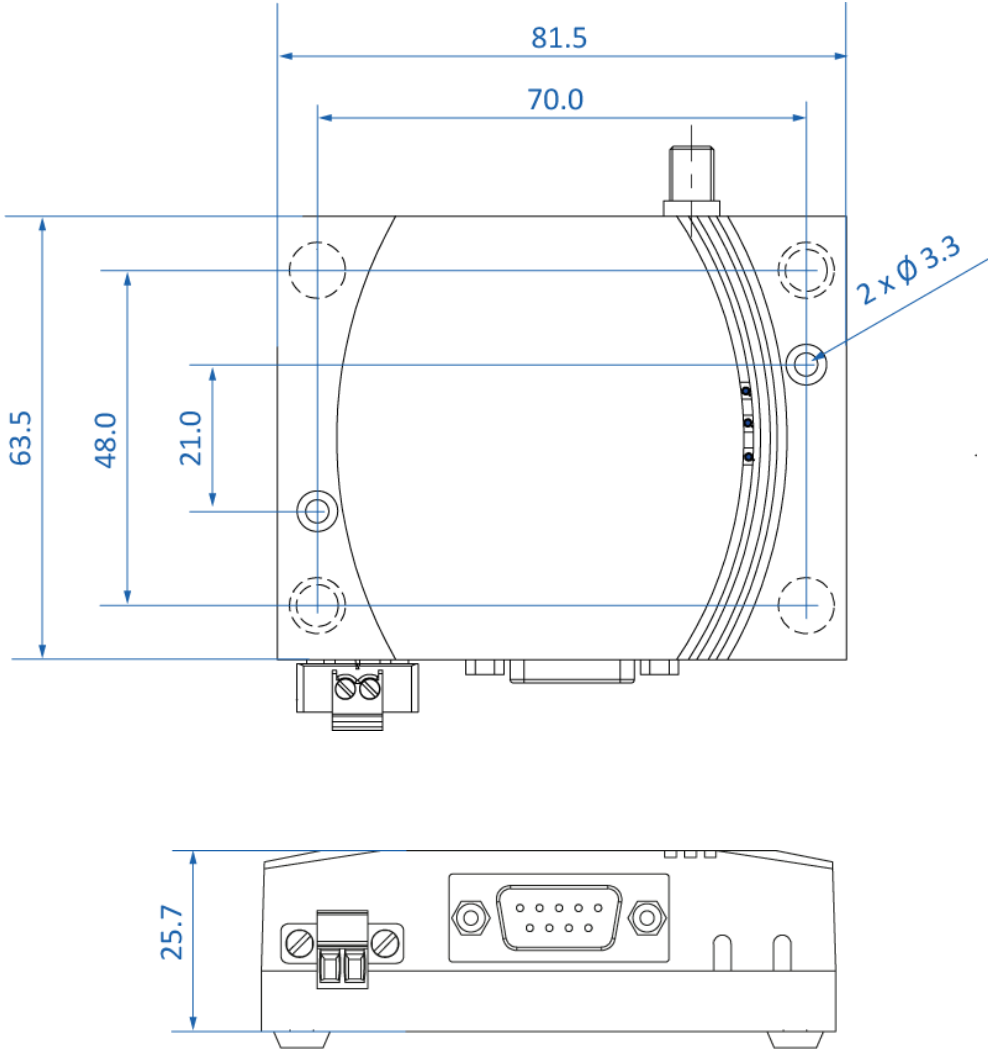
## B Disposal and recycling



You must dispose of this product properly according to local laws and regulations. Because this product contains electronic components, it must be disposed of separately from household waste. When this product reaches its end of life, contact local authorities to learn about disposal and recycling options, or simply drop it off at your local HMS office or return it to HMS.

For more information, see [www.hms-networks.com](http://www.hms-networks.com).

### C Measurements



## D Configuration Examples

### D.1 Example 1: Connecting a CAN Network With a Computer

The example shows how an installed virtual COM port can be used to configure the CANblue II to exchange data with a CAN network connected to the CANblue II.

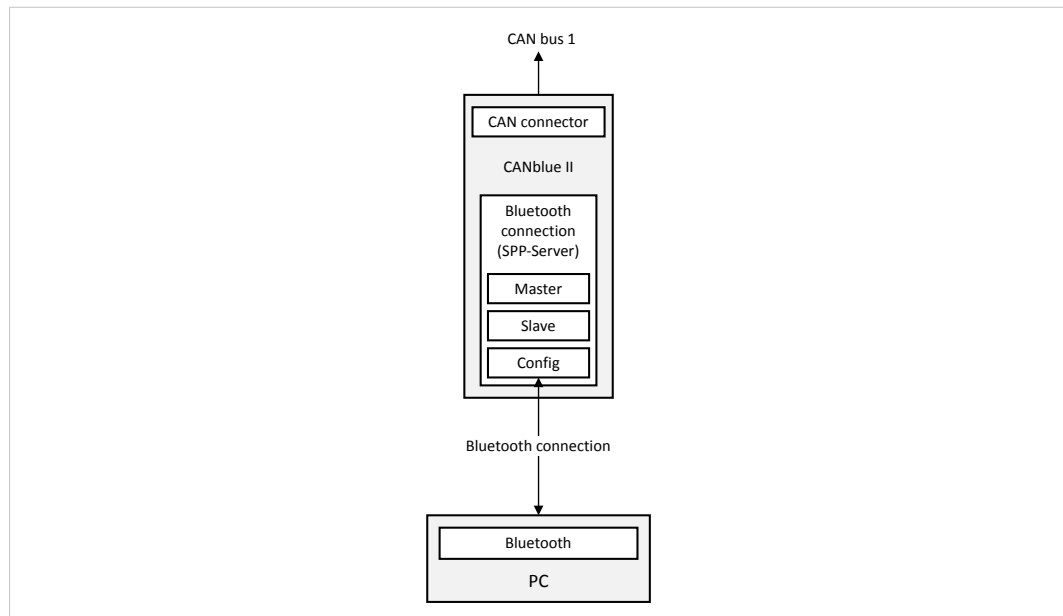


Fig. 12 PC Interface

The following specifications apply in the example:

- CAN network is operated at data rate of 500 kBaud.
- Exclusively the following messages are forwarded by the CANblue II:
  - data and remote frames with Standard identifier 5
  - remote frames with Standard identifier 1F
  - data frames with Extended identifier 1A2B3C
- ▶ Make sure that the virtual COM port is installed.
- ▶ Make sure that the software package (*CANblue\_II\_Generic\_Setup.exe*) is installed.
- ▶ Start the terminal program or the CANblue Configuration Tool (*CANblueCon.exe*).
- ▶ Reset the device to factory settings with command `D SETTINGS_DEFAULT`.
- ▶ Initialize the CAN controller to 500 kBaud with command `C CAN_INIT 500`.
- ▶ To set the filter, use the following commands:
  - `C FILTER_ADD 5`
  - `C FILTER_ADD STD 5 RTR`
  - `C FILTER_ADD STD 1F`
  - `C FILTER_ADD EXT 1A2B3C`
- ▶ Activate the Standard filter with command `C FILTER_ENABLE STD`.
- ▶ Activate the Extended filter with command `C FILTER_ENABLE EXT`.
- ▶ Check the configuration with command `C CONFIG SHOW`.
- ▶ Save the configuration with command `C CONFIG SAVE`.

- ▶ Start the CAN controller with command `C CAN_START`.
  - If the CAN controller receives a message from the CAN network that matches one of the filters, the message is transmitted on the Bluetooth connection in ASCII format.
- ▶ To transmit CAN messages to the CANblue II or into the connected CAN network use ASCII format or binary format (see [Generic Mode Network and Device Communication, p. 28](#)).
  - Transmission format of CAN messages is automatically matched to the received format.
- ▶ To transmit a CAN data frame with the Standard identifier `7FF` and the data bytes `1A 2B 3C 4D 5E 6F 70` to the CAN bus, use command `M SD7 7FF 1A 2B 3C 4D 5E 6F 70`.

## D.2 Example 2: Configuring a CAN Bridge

The example shows how a CANblue II (configured as in example 1) is connected to a second CANblue II.

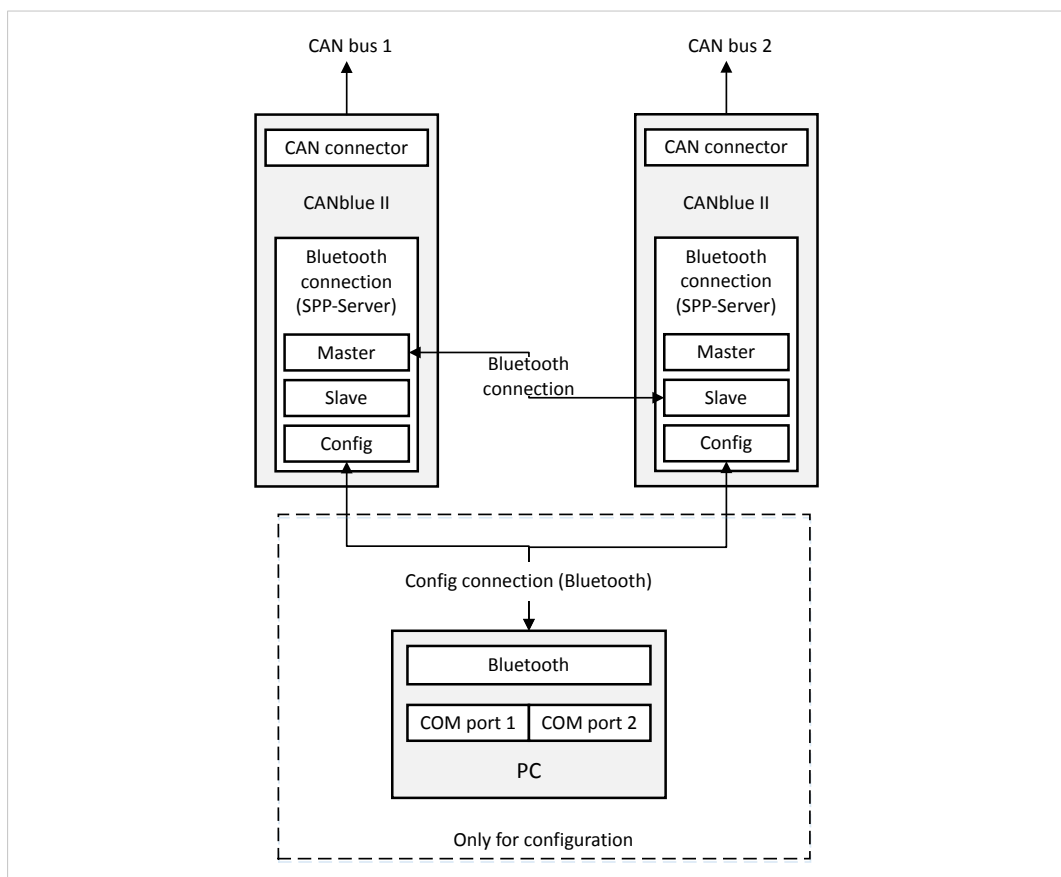


Fig. 13 Configuring a bridge

The following specifications apply in the example:

- The Slave is connected to a 1000 kBaud CAN network.
- The Master (configured as in example 1) forwards all Standard CAN messages and filters out all Extended CAN messages.
- The Slave forwards all CAN messages.

### Slave Device

- ▶ Make sure that the virtual COM port is installed and that a connection is established.
- ▶ Make sure that the software package (*CANblue\_II\_Generic\_Setup.exe*) is installed.

- ▶ Start the terminal program or the CANblue Configuration Tool (*CANblueCon.exe*).
- ▶ Reset the device to factory settings with command `D SETTINGS_DEFAULT`.
  - CAN controller is automatically initialized to 1000 kBaud (preset in factory settings).
- ▶ Enable the autostart mode with command `C AUTOSTART ON`.
- ▶ Save the configuration with command `C CONFIG SAVE`.

### Master Device

- ▶ Make sure the Master (configured as in example 1) is connected to the virtual COM port.

To simplify the configuration, turn off the transmission of CAN messages by the Master:

- ▶ Stop the CAN controller with command `C CAN_STOP`.
  - or
- ▶ Disable the transmission of CAN messages on the specific connection with command `C SEND_CAN_FRAMES OFF`.
- ▶ Disable the filter of the Master with command `C FILTER_DISABLE STD`.
  - All Standard CAN messages are forwarded by the Master.
- ▶ Delete all Extended filter entries with command `C FILTER_CLEAR EXT`.
  - Device filters out all Extended CAN messages (set by `C FILTER_ENABLE EXT` in example 1).
- ▶ Enable the autostart mode of the Master with command `C AUTOSTART ON`.
- ▶ With the Master use command `D MAC_ADD <address of Slave>`.
  - Devices connect as Master and Slave and start automatically.
  - Devices work as Bridge between the two CAN networks.
- ▶ Save the configuration with command `C CONFIG SAVE`.
- ▶ To achieve the highest possible data rate between the devices, disconnect the Config connection to the computer.

Since the connection is stored on both devices, devices reconnect automatically after turning off and on and resume to forward CAN messages.

### D.3 Example 3: Configuring a Bridge Chain

Configuring a bridge chain is possible because every Slave can serve as Master for another Slave.

To connect a third CAN bus by using an additional CANblue II to the CAN buses that are configured in example 1 and 2, two options are possible:

- connecting the Slave to the new device (Slave serves as Master for the new device)
- connecting the new device to the Master (Master serves as Slave for the new device)

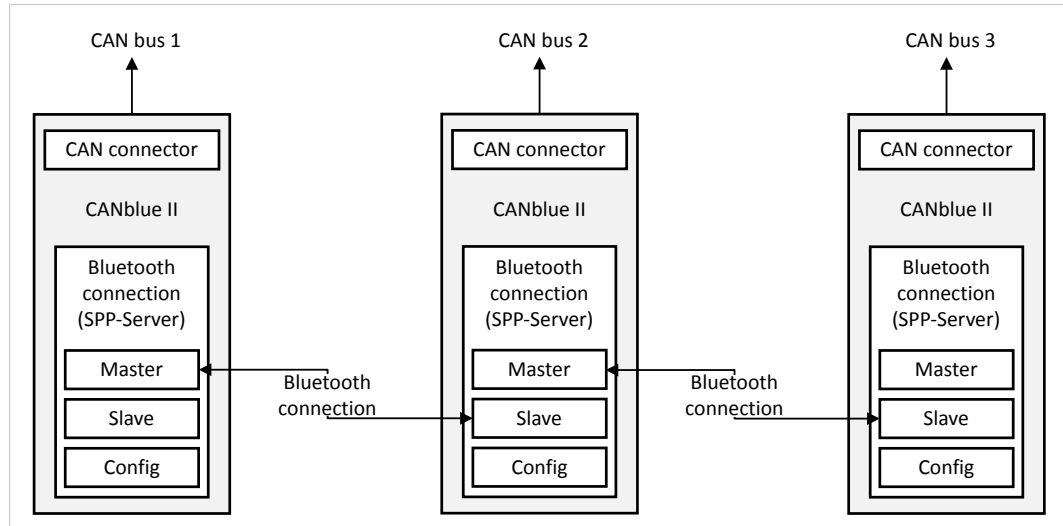


Fig. 14 Bridge chain configuration



*Each additional CAN bus increases the rate of CAN messages on the Bluetooth connections and reduces the maximum possible data rate of all connections.*



**This page intentionally left blank**

