Manual

CAN@net II/Generic

TCP/IP Gateway for CAN with Open Socket API

IXXAT

IXXAT
Member of the HMS group
IXXAT Automation GmbH
Leibnizstr. 15
D-88250 Weingarten

Tel.: +49 (0)7 51 / 5 61 46-0
Fax: +49 (0)7 51 / 5 61 46-29
Internet: www.ixxat.de
e-Mail: info@ixxat.de

Support

In case of unsolvable problems with this product or other IXXAT products please contact IXXAT in written form by:

Fax: +49 (0)7 51 / 5 61 46-29
e-Mail: support@ixxat.de

Copyright

Duplication (copying, printing, microfilm or other forms) and the electronic distribution of this document is only allowed with explicit permission of IXXAT Automation GmbH. IXXAT Automation GmbH reserves the right to change technical data without prior announcement. The general business conditions and the regulations of the license agreement do apply. All rights are reserved.

Registered trademarks

All trademarks mentioned in this document and where applicable third party registered are absolutely subject to the conditions of each valid label right and the rights of particular registered proprietor. The absence of identification of a trademark does not automatically mean that it is not protected by trademark law.

Document number: 4.01.0086.20001
Version: 1.2

Figures:

Screenshots:

1 Introduction

1.1 Overview

The CAN@net II/Generic features simple, flexible access to a CAN system via Ethernet. By supporting the TCP/IP protocol, the CAN@net II/Generic can be accessed worldwide via Internet.

With the IXXAT CAN@net II/Generic, you have purchased a high-quality electronic component that has been developed and manufactured according to the latest technological standards.

This manual is intended to help familiarize you with your device. Please read this manual before beginning with installation.

1.2 Support

For more information on our products, FAQ lists and installation tips please refer to the support area on our homepage (http://www.ixxat.de). There you will also find information on current product versions and available updates.

If you have any further questions after studying the information on our homepage and the manuals, please contact our support department. In the support area on our homepage you will find the relevant forms for your support request. In order to facilitate our support work and enable a fast response, please provide precise information on the individual points and describe your question or problem in detail.

If you would prefer to contact our support department by phone, please also send a support request via our homepage first, so that our support department has the relevant information available.

1.3 Returning hardware

If it is necessary to return hardware to us, please download the relevant RMA form from our homepage and follow the instructions on this form. In the case of repairs, please also describe the problem or fault in detail on the RMA form. This will enable us to carry out the repair quickly.

2 Functional concept

The CAN@net II/Generic hardware provides connectivity to Ethernet and CAN networks. The CAN@net II/Generic application firmware running on the CAN@net II/Generic provides functions to access a CAN bus from virtually every Ethernet/TCP/IP host.

In order to do so, the CAN@net II/Generic is running a server on TCP Port #19227, which is capable to handle the ASCII Protocol specified in chapter 4. Any Ethernet/TCP/IP host can connect to this server and exchange commands and CAN messages with the server using the ASCII Protocol. The server relays the commands and messages to the CAN bus and vice versa.

The server is restricted to accept only a single connection to TCP port #19227. Additional connection requests are rejected.

2.1 Gateway setup

The picture below shows the standard configuration for a CAN@net II/Generic used as gateway to a CAN system.

In the gateway configuration the CAN@net II/Generic usually is hooked to the company intranet at the site where the CAN system of interest is located. This allows any TCP/IP host within the reach of this intranet to connect to the CAN@net II/Generic and thus gain control of the CAN system.

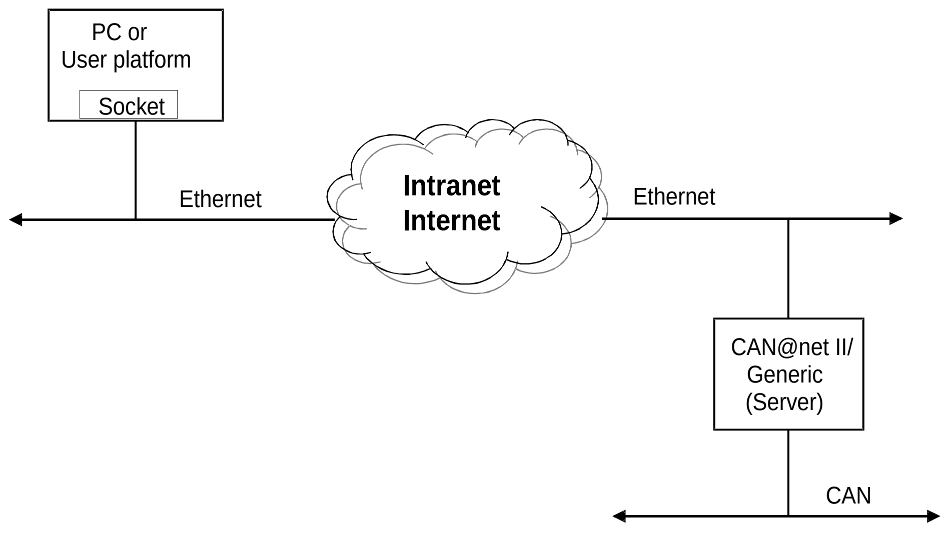


Figure 2-1 Gateway configuration

2.2 Bridge setup

The picture below shows the standard "Bridge" setup for the CAN@net II/Generic.

The bridge setup allows you to connect two CAN systems over an Ethernet/TCP/IP network, e.g. the company intranet or even the internet. For the bridge setup one of the CAN@net II/Generic takes over the role of the client which connects to a CAN@net II/Generic in standard server configuration. In Bridge mode standard and extended CAN messages are forwarded in both directions.

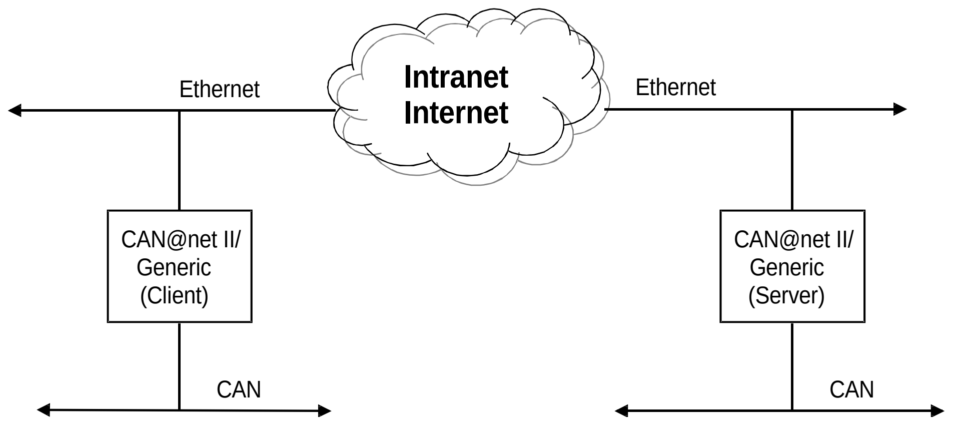


Figure 2-2 Bridge configuration

2.3 CAN ID filtering

The CAN@net II/Generic application firmware includes a filtering mechanism based on CAN Identifiers. The basis for the filtering is a filter list, which contains the CAN Identifiers of interest. A CAN message with an Identifier contained in the filter list is forwarded; all other CAN messages are discarded.

Filtering only applies to the direction CAN system → TCP/IP network. In the reverse direction, no filtering is available.

The CAN@net II/Generic provides functions to set or clear CAN Identifiers in the filter list. It also allows you to write the filter list to the Flash memory so that it can be loaded and used upon startup of the CAN@net II/Generic.

3 Overview

3.1 Connectors of the CAN@net II/Generic

The following diagram shows the connectors of the CAN@net II/Generic. The pin assignment is described in detail in the manual "CAN@net II - Intelligent PC/CAN Interface".

The CAN@net II/Generic also has four LEDs. When connected to the power supply, the Power LED is on. The CPU LED indicates the status of the Firmware.

For the Ethernet and CAN connectors two Duo-LEDs are available which indicate the communication status.

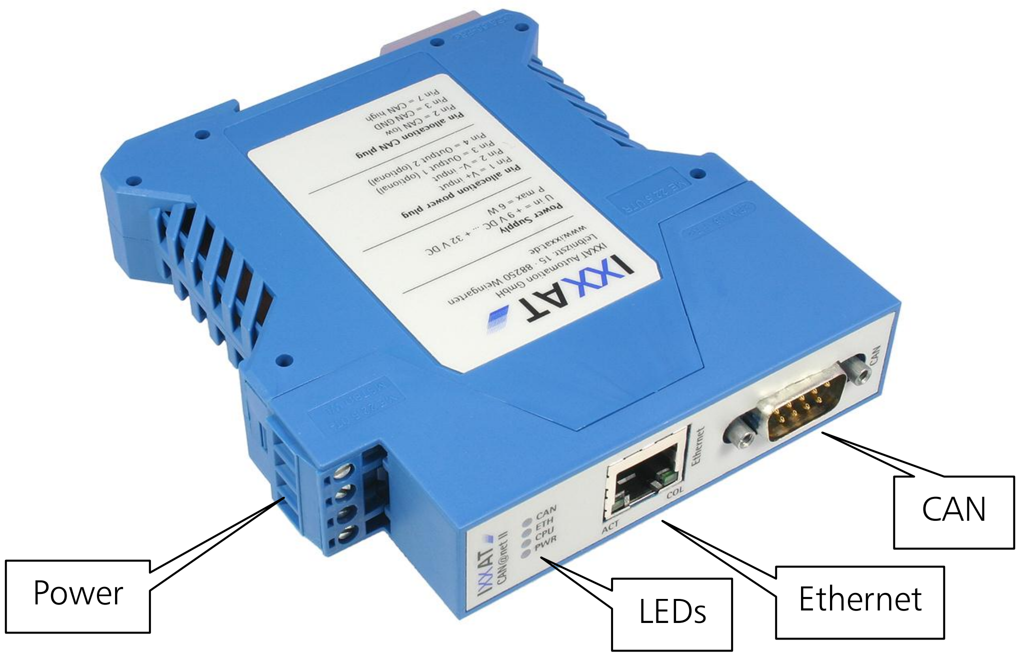


Figure 3-1 Connectors and displays of the CAN@net II/Generic

3.2 Basic device configuration

Before initial commissioning of the CAN@net II/Generic, the device must be configured. Please see chapter "4 Configuration" of the manual "CAN@net II - Intelligent PC/CAN Interface".

3.3 Gateway configuration

In order to use a CAN@net II/Generic as server in a gateway mode (as described in 2.1) it is sufficient to do a basic configuration.

3.4 Bridge configuration

For a bridge configuration, two CAN@net II/Generic are required. One of them must be configured as server and the other one as client.

(1) The basic configuration, as described in chapter 3.2 "Basic device configuration", must be done for both devices. For the CAN@net II/Generic, which shall act as server, it is sufficient to do the basic configuration.
(2) One of the two CAN@net II/Generic devices must be configured to act as client. This can be done via the Web Interface; see the description in chapter 5.2 "Bridge configuration".

An example for the bridge configuration can be found in Appendix A.

4 ASCII Protocol Server

Every CAN@net II/Generic that is configured as server runs a server on the TCP Port 19227. This server handles the ASCII Protocol described in the following sections.

4.1 Server function

After power-up the CAN@net II/Generic automatically starts the protocol server on TCP Port 19227.

The server accepts only a single connection. Additional connection requests are rejected. After a connection is established, the server is ready to exchange data and commands coded with the ASCII Protocol.

The ASCII Protocol is used to pack data (CAN messages) and commands for the transfer over the Ethernet/TCP/IP network. E.g.: When the server receives an ASCII Protocol message which contains a CAN message, it extracts the original CAN message and sends it on the attached CAN bus. Messages seen on the CAN bus are packed into the ASCII protocol and forwarded to the connected Ethernet/TCP/IP client.

In addition to CAN messages the server also handles commands. These can be commands to start/stop the CAN controller, to set the CAN baudrate and so on. It is also possible to retrieve the version numbers of the CAN@net II/Generic firmware or of the ASCII-Protocol.

4.2 ASCII Protocol

The following sections describes the ASCII Protocol and how it is handled by the ASCII Protocol server.

4.2.1 Basic message format

There are some basic rules the ASCII-Protocol does follow:

- The messages are coded with ASCII characters
- Valid characters are letters from a to z (no national characters) and numbers from 0 to 9. Small or capital letters are not distinguished.
- A message starts with a valid ASCII character and is terminated with \r\n. The \r\n is separated by a space from the last significant character.
- Directly after the \r\n the next message can follow.
- Instead of using \r\n, any combination of \r and \n can be used (e.g. \r, \n, \n\n, …).
- Messages containing invalid characters are discarded.

- Message contents like CAN Identifier or CAN data are noted in HEX notation always. Other formats are not supported. The HEX specifiers (0x…, ..HEX) are omitted.
- An ASCII-Protocol message consists of groups of ASCII characters, each group separated by a space character (0x20).
- More than one consecutive space characters (0x20) are reduced to a single space character.
- The groups of ASCII characters describe different types of messages or commands contained in the ASCII-Protocol message.
- The single characters of an ASCII-Protocol message are sent over the TCP connection in readable order; beginning with the "message type" group of ASCII characters and ending with the termination \r\n.

The ASCII-Protocol in its Version 1.0 supports 5 different message types which are described in more detail in the following chapters. The message types are:

- CAN message
- CAN controller command
- Device command
- Error message
- Info message

4.2.2 CAN message

The ASCII-Protocol messages of type "CAN message" are used to exchange CAN messages between the CAN@net II/Generic and the Ethernet/TCP/IP host connected to server port 19227. The "CAN message" is used to exchange information in both directions; to and from the CAN@net II/Generic.

- When the CAN@net II/Generic receives a message on the CAN bus it packs this CAN messages into an ASCII-Protocol message of type "CAN message" and sends it over Ethernet/TCP/IP.
- When the CAN@net II/Generic receives an ASCII-Protocol message of the type "CAN message" from Ethernet/TCP/IP it unpacks this message, translates it into a CAN message which it sends on the CAN bus immediately.

Message format:

M	FTD	ID	XX	XX	XX	XX	XX	XX	XX	XX	\r\n

The groups of characters are separated by a space each. For a valid "CAN message", the fields M, FTD, ID and \r\n must be specified. The groups shown as XX correspond to the CAN data bytes and are optional; a "CAN message" may contain 0 … 8 of these fields.

Detailed:

M	Message type "CAN Message"
FTD	CAN message details: F - Frame Format (S – standard; E – extended) T – Frame Type (D – Data; R – RTR) D – optional; Data Length Code (0 through 8)
ID	CAN Message ID
XX	CAN Message Data; 0 through 8 Bytes long
\r\n	Message terminator

Here some examples for "CAN messages":

Extended Data, ID 0x100, Data [0x11 0x22 0x 33 0x44] :
M ED4 100 11 22 33 44 \r\n

Standard Data, DLC 7, ID 0x200, Data [0x1 0x2 0x3 0x4 0x5 0x6 0x7] :
M SD7 200 1 2 3 4 5 6 7 \r\n

4.2.3 CAN controller command

Messages of the type "CAN Controller Command" are used to control the CAN controller on the CAN@net II/Generic and to modify the settings of the filter table.

Message format:

C	Command	Parameters	\r\n

The groups of characters are separated by a space each. For a valid "CAN controller command", the fields C, Command and \r\n must be specified. The field Parameters is optional and its usage depends on the exact Command.

Detailed:

C	Message type "CAN Controller Command"
Command Parameters	Usage see the following table
\r\n	Message terminator

Commands:

Command	**Parameters**	**Description**
INIT	Baud rate	CAN controller standard initialization. Possible baud rate values - 0, 10, 20, 50, 100, 125, 250, 500, 1000. If baud rate is set to 0, the CAN controller tries to auto detect a baud rate on the bus.
INIT CUSTOM	BT0, BT1, Name	CAN controller custom initialization with register values BT0, BT1 in hex format and baud rate name , for example for 800 kBd (bt0 is 0x0, bt1 is 0x16) : INIT CUSTOM 00 16 800_kBd (BT0, BT1 are registers of the SJA1000 running at 16 MHz)
START	-	Start an initialized CAN controller
STOP	-	Stop a CAN controller
RESET	-	Reset a CAN controller
STATUS	-	Device issues an info message which contains the actual busload and CAN controller status flags.

FILTER ADD	CAN ID	Add CAN ID to the controller filter list
FILTER REMOVE	CAN ID	Remove CAN ID from the controller filter list
FILTER CLEAR	-	Clear the filter list
FILTER ENABLE	-	Enable the filter list
FILTER DISABLE	-	Disable the filter list
FILTER LOAD	-	Load the filter list from flash
FILTER SAVE	-	Save the filter list to flash
FILTER SHOW	-	Show content and status of the controller filter list

Table 4.2.1 CAN controller commands

All messages of type "CAN controller command" are answered by the ASCII-Protocol server with either an error message (see 4.2.5) or an OK packed in an Info message (see 4.2.6).

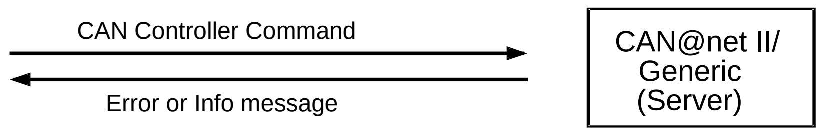


Figure 4-1 CAN Controller Command Request/Response Cycle

It is good practice to issue CAN controller commands to the ASCII-Protocol server only one at a time and then wait for the response. As some commands may take longer to be processed than others, the timing of the response messages may vary. So the sequence of responses to commands which were issued in a row may be changed. And the ASCII-Protocol provides no mechanism (like message identifiers) to match commands to their responses.

4.2.4 Device command

Messages of the type "Device command" are used to exchange information regarding the CAN@net II/Generic as a whole.

Message format:

D	Command	Parameters	\r\n

The groups of characters are separated by a space each. For a valid "Device command", the fields D, Command and \r\n must be specified. The field Parameters is optional and its usage depends on the exact Command.

Detailed:

D	Message type "Device Command"
Command Parameters	Usage see the following table
\r\n	Message terminator

Commands:

Command	Parameters	Description
VER	-	Device issues an info message which contains the firmware version information
PROTO	-	Device issues an info message which contains supported protocol version information
CONFIG SAVE	-	Stores the actual setting of the filter into the flash. The filter settings are loaded after a device reset or power on.

Table 4.2.2 Device commands

All messages of type "Device command" are answered by the ASCII-Protocol server with either an error message (see 4.2.5) or requested information packed in an Info message (see 4.2.6).

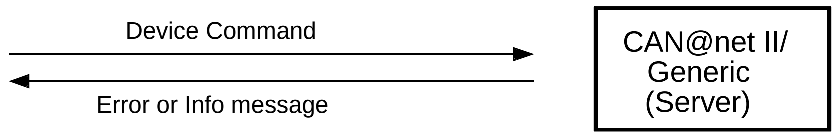


Figure 4-2 Device Command Request/Response Cycle

It is good practice to issue Device commands to the ASCII-Protocol server only one at a time and then wait for the response. As some commands may take longer to be processed than others, the timing of the response messages may vary. So the sequence of responses to commands which were issued in a row may be changed. And the ASCII-Protocol provides no mechanism (like message identifiers) to match commands to their responses.

4.2.5 Error message

Messages of the type "Error message" are used to inform the ASCII-Protocol client about problems or errors on the server side.

Message format:

E	Error number	Error description	\r\n

The groups of characters are separated by a space each. For a valid "Error message", all fields must be specified.

Detailed:

E	Message type "CAN Controller Command"
Error number	Error code
Error description	Readable error description
\r\n	Message terminator

Error codes and their description:

Error	**Code**	**Description**
ERR_QUEUE_OVERRUN	0x10	Software queue overrun
ERR_DATA_OVERRUN	0x11	CAN controller buffer overrun
ERR_STATUS_SET	0x12	CAN error status set
ERR_STATUS_RESET	0x13	CAN error status reset
ERR_BUS_STATUS_SET	0x14	Bus off
ERR_MSG_FRAME_ FORMAT_UNKNOWN	0x20	Unknown message frame format

ERR_MSG_RTR_FLAG_ UNKNOWN	0x21	Unknown message RTR flag
ERR_CONNECTION_ REJECTED	0x70	Device rejected incoming con-nection because it is already connected
ERR_ID_NOT_FOUND	0x75	CAN ID was not found in the filter list
ERR_ID_IN_LIST	0x76	CAN ID is already in the filter list
ERR_LIST_FULL	0x77	Filter list is full
ERR_CMD_UNKNOWN	0x80	Syntax error. Unknown com-mand.
ERR_CAN_BAUD_UNKNOWN	0x81	Unknown CAN baud rate in "C INIT" command
ERR_CAN_AUTOBAUD_ FAILED	0x82	CAN bus baud rate was not detected
ERR_CAN_SET_BAUD_ FAILED	0x83	CAN controller failed to set requested baud rate
ERR_INTERFACE_STATE_ WRONG	0x90	Interface is in the wrong state to execute this command

Table 4.2.3 CAN@net II/Generic error codes

4.2.6 Info message

Messages of the type "Info message" are used to feed the ASCII-Protocol client with actual information. This message is sent only as response to requests like "CAN controller command" or "Device command". They will never be sent in an asynchronous way without previous request.

Message format:

I	Message text	\r\n

The groups of characters are separated by a space each. For a valid "Info message", all fields must be specified.

Detailed:

I	Message type "Info message"
Message text	Readable message text
\r\n	Message terminator

Example for an "Info message":

Response to a "CAN controller Command"; the command has been executed successfully:

I OK \r\n

4.3 CAN message filter

As described in chapter 2.3 CAN ID filtering, the ASCII-Protocol server includes a filtering mechanism for CAN messages. Filtering is done on the basis of CAN Identifiers and is applied to the direction CAN bus → Ethernet only.

The filter mechanism can be enabled or disabled via the Web Interface as shown in 5.3 Filter configuration. With the filter list disabled, no filtering is applied to the CAN messages; all messages are forwarded from CAN bus to Ethernet regardless of their ID. With the filter list enabled, only CAN messages with Identifiers stated in the filter list are forwarded.

Upon startup of the CAN@net II/Generic, the filter list is loaded from Flash to RAM. Now the filter list is ready for use and will be applied if filtering is enabled. Modifications to the filter list can be made via the Web Interface and apply to the filter list in RAM only. The modifications come into effect immediately after they have been acknowledged by the Web Interface. If, for example, a new Identifier has been added to the list, CAN messages with this new ID will now be forwarded to the Ethernet.

Due to the fact that this filter list is kept in RAM, the modifications to it are volatile. This means that with power down the modifications will be gone. Upon the next startup, the old, unmodified filter list would be loaded from Flash.

If the modifications to the filter list should be kept, the user must write the filter list from RAM to Flash. The Web Interface offers a "Save" button which exactly does this. When done, the filter list including the latest modifications is in Flash and will be loaded automatically upon the next startup.

For all possible operations on the filter list see 5.3 Filter configuration.

4.4 Message sequencing

In some cases it might be important to understand how and in which order the ASCII-Protocol server processes commands and messages.

4.4.1 Command processing

When the ASCII-Protocol server receives any command, it tries to execute it immediately. When done, it responds with an acknowledge or an error message to the client. Due to the fact that command processing takes more or less time for different commands, race conditions for the responses can occur. It is recommended to wait for the response on a previous command before sending a following command.

This behavior is also referred to in chapters 4.2.3 and 4.2.4.

4.4.2 Message processing

The timing behavior for forwarding CAN messages from the CAN bus to the Ethernet or vice versa depends on the workload the CAN@net II/Generic has to handle. Also the network connection has influence on the transmission time for CAN messages. Measurements showed that these time can vary from 2 … 200 msec or even worse.

When the workload on the CAN@net II/Generic is so high that it can not process all messages, the excess messages are discarded and an error message will be issued. This is valid for message forwarding in both directions.

Regardless the timing issue or even the loss of messages, it is guaranteed that the order of the messages is not changed. A message which was seen on the CAN bus first will be transmitted to the Ethernet side first. This is valid for both directions.

4.5 Getting started

This chapter describes the necessary steps to start the CAN data exchange between the ASCII-Protocol server and the client.

After Power Up, the CAN@net II/Generic is busy booting. The ASCII-Protocol server is started and ready to receive and process commands. This is indicated by the CPU LED, which is blinking with a frequence of 1Hz.

The next step is to initialize the CAN controller hardware. Because the CAN controller is not started up yet, it is not possible to exchange CAN messages. Any

CAN messages received by the ASCII-Protocol server during this state are discarded.

The necessary steps are as follows:

- Issue a CAN init command to the ASCII-Protocol server. This command also sets the desired CAN bus baud rate. The command "C INIT 500 \r\n" for example will initialize the can controller hardware to 500 kBit/sec.
- Wait for the ASCII-Protocol Server to respond, either with an error message or with an OK.
- Issue a CAN start command “C START” to the ASCII-Protocol server. This command will put the CAN controller hardware into operating mode.
- Wait for the ASCII-Protocol Server to respond; either with an error message or with an OK.

Now the ASCII-Protocol server is fully configured and operational. If there was any CAN data sent to the ASCII-Protocol server during this setup session, the data has been discarded.

After the CAN controller hardware is started up (did send start command to CAN controller), the ASCII-Protocol server can receive data from the CAN bus and does forward this data to the ASCII-Protocol client.

An example for these necessary steps can be seen in the coding example described in section 6.2.

5 Web interface

The CAN@net II/Generic also includes a standard WebServer (HTTP Server) which can be accessed using any standard WebBrowser with Javascript enabled. This could be the Microsoft Internet Explorer, the Netscape Navigator, Firefox, … .

The Web Interface allows the user to modify the device configuration and the filter settings of the CAN@net II/Generic. The modifications can be done in a convenient, interactive way which will be described in the following chapters.

In order to use the Web Interface, the CAN@net II/Generic must be configured as described in chapter 3.2 Basic device configuration. At least the CAN@net II/Generic must have its own IP address and also the rest of the IP settings. Also make sure that the CAN@net II/Generic is connected to the network.

In order to connect to the CAN@net II/Generic Web Interface simply type in the URL "http://xxx.xxx.xxx.xxx" with xxx.xxx.xxx.xxx being the actual IP address of the CAN@net II/Generic.

Please make sure that your Web Browser has no Proxy settings enabled and that the IP network between your host running the WebBrowser and the CAN@net II/Generic is transparent for HTTP data exchange. Please contact your system administrator to clarify these issues. The Bridge Configuration need a javascript capable WebBrowser.

5.1 Main page

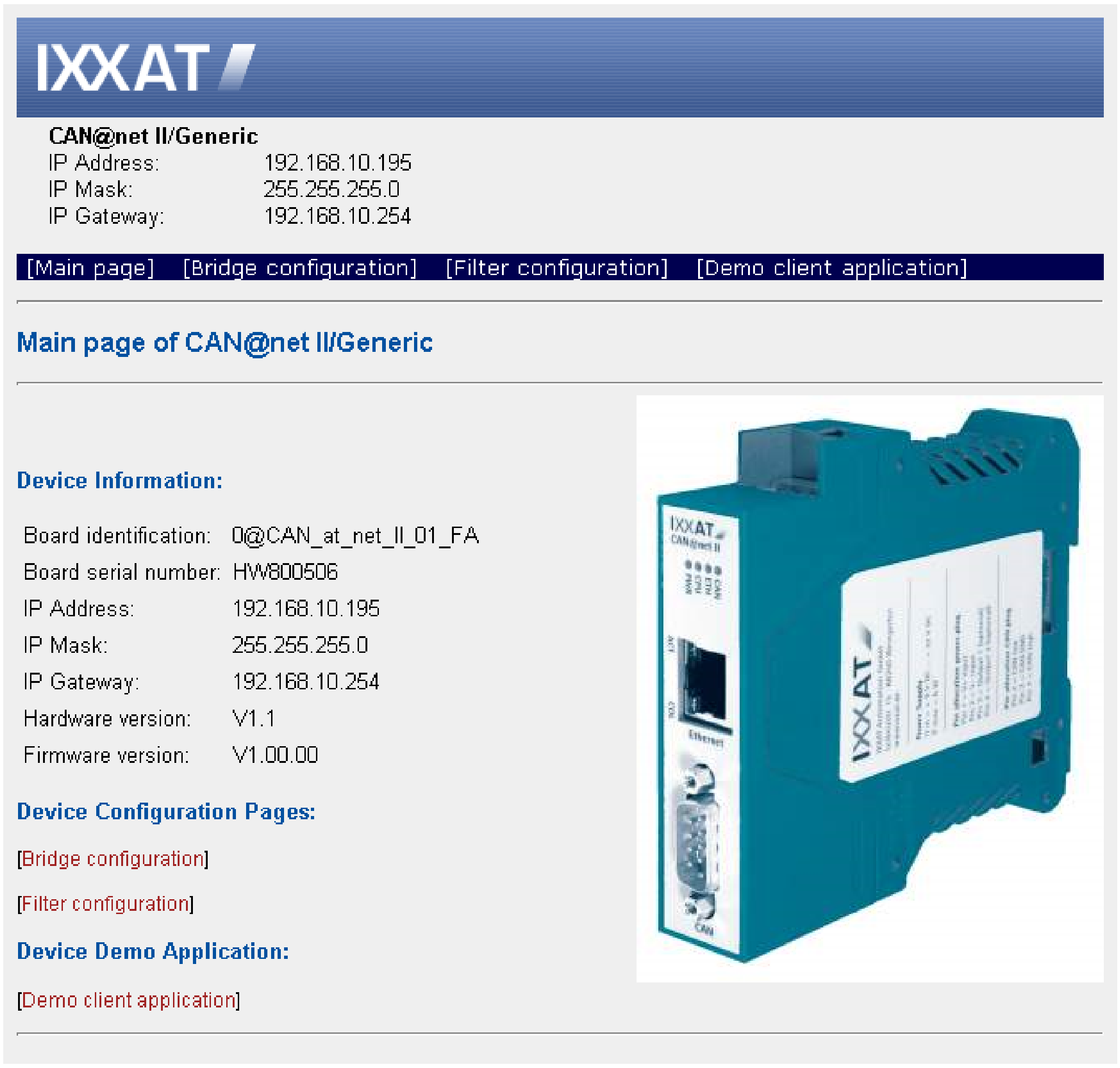
When the WebBrowser manages to open a connection to the Web Interface, the main page or CAN@net II/Generic Home Page will be displayed by the Web-Browser. The looks of the main page are shown with Screenshot 5.1 on the following page.

The main page states device information (firmware/ hardware version, device serial number, …) of the CAN@net II/Generic.

The main page serves as menu to the different sub menus which help a user to configure the CAN@net II/Generic. In the current version there are three sub menus:

- bridge configuration
- filter configuration
- demo application download

The user can proceed to the sub menus by clicking the appropriate link provided on the main page.



CAN@net II/Generic

IP Address:	192.168.10.195
IP Mask:	255.255.255.0
IP Gateway:	192.168.10.254

[Main page] [Bridge configuration] [Filter configuration] [Demo client application]

Main page of CAN@net II/Generic

Device Information:

Board identification:	0@CAN_at_net_II_01_FA
Board serial number:	HW800506
IP Address:	192.168.10.195
IP Mask:	255.255.255.0
IP Gateway:	192.168.10.254
Hardware version:	V1.1
Firmware version:	V1.00.00

Device Configuration Pages:

[Bridge configuration]

[Filter configuration]

Device Demo Application:

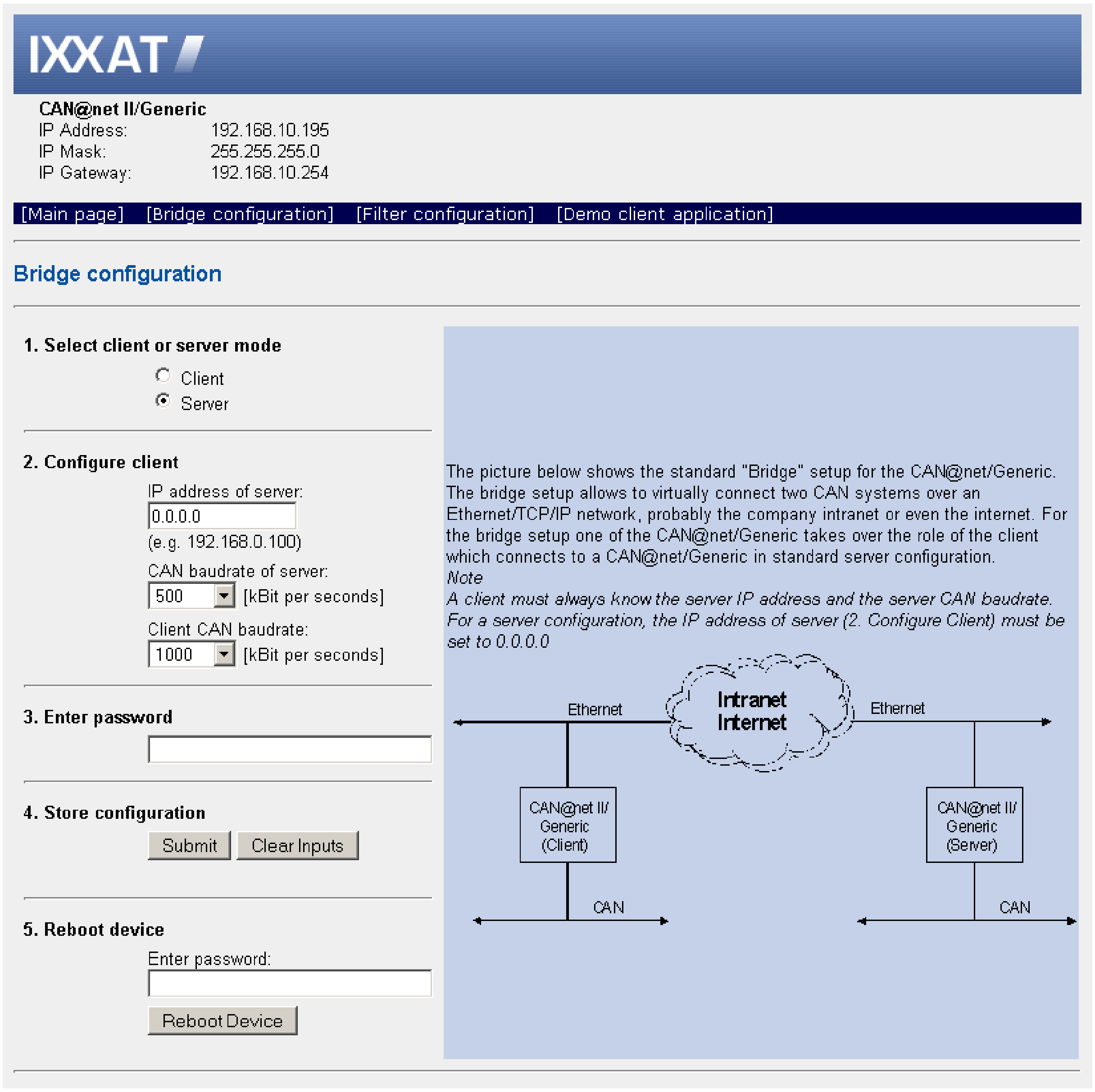
[Demo client application]

Screenshot 5.1 Main page

5.2 Bridge configuration

The bridge configuration page allows to modify whether the CAN@net II/Generic should act as ASCII-Protocol server or client.

The look of the device configuration page are shown with Screenshot 5.2 on the following page. When opened, the device configuration page displays the current settings. By clicking into the fields with the current values, the settings can be edited.



CAN@net II/Generic

IP Address:	192.168.10.195
IP Mask:	255.255.255.0
IP Gateway:	192.168.10.254

[Main page] [Bridge configuration] [Filter configuration] [Demo client application]

Bridge configuration

1. Select client or server mode

Client [✓ Server]

2. Configure client

IP address of server: 0.0.0.0
(e.g. 192.168.0.100)

CAN baudrate of server: 500 [kBit per seconds]

Client CAN baudrate: 1000 [kBit per seconds]

3. Enter password

4. Store configuration

5. Reboot device

Enter password:

The picture below shows the standard "Bridge" setup for the CAN@net/Generic. The bridge setup allows to virtually connect two CAN systems over an Ethernet/TCP/IP network, probably the company intranet or even the internet. For the bridge setup one of the CAN@net/Generic takes over the role of the client which connects to a CAN@net/Generic in standard server configuration.

Note

A client must always know the server IP address and the server CAN baudrate. For a server configuration, the IP address of server (2. Configure Client) must be set to 0.0.0.0

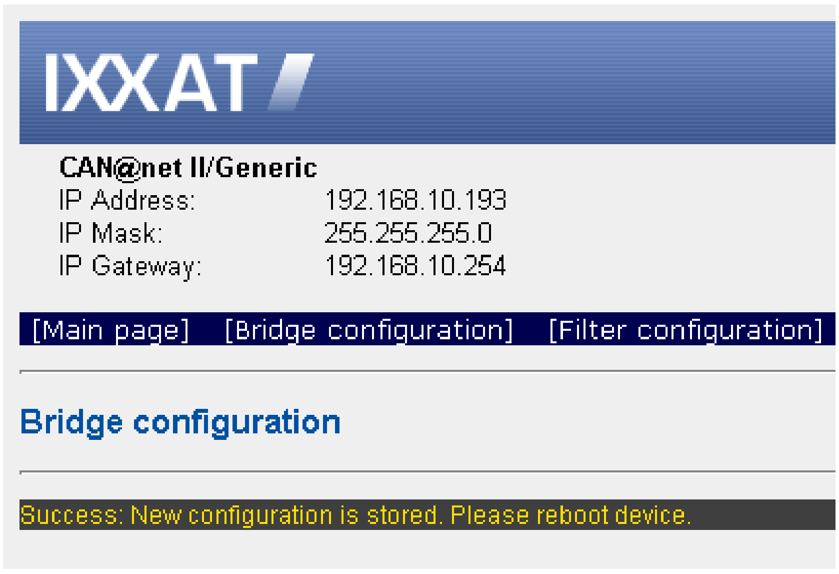
Screenshot 5.2 Bridge configuration

Bridge configuration steps:

1. If the CAN@net II/Generic should act as ASCII-Protocol server only, the "IP address of Server " field of step 2 must be set to 0.0.0.0 as shown in the screenshot.
2. Under the header "Configure client " there are the settings that must be given, when a CAN@net II/Generic should act as client in a bridge configuration as described in 2.2

Bridge setup. As soon as a valid IP address is entered here, the CAN@net II/Generic will act as client in a bridge configuration. It will use the "IP address of Server" as IP address of the ASCII-Protocol server it will connect to. When acting as client, also the CAN baudrates for both, the client and the server, must be specified.

3. Enter the password for your device. Without the right password, changes for the bridge configuration are not stored in the device.
4. While editing the fields on the device configuration page, the new values are stored locally inside the WebBrowser. In order to adopt the new values, the submit button must be clicked. Only then the new values are transferred to the WebServer on the CAN@net II/Generic and become valid. The WebServer responds to this by displaying the same configuration page with additional process reply information right under the "Bridge configuration" Headline.



CAN@net II/Generic

IP Address:	192.168.10.193
IP Mask:	255.255.255.0
IP Gateway:	192.168.10.254

Bridge configuration

Success: New configuration is stored. Please reboot device.

Screenshot 5.3 Bridge configuration reply information

5. The new bridge configuration is now stored in the device. In order to apply the new settings, the device must reboot. Enter the password for your device and press the "Reboot Device" button.

5.3 Filter configuration

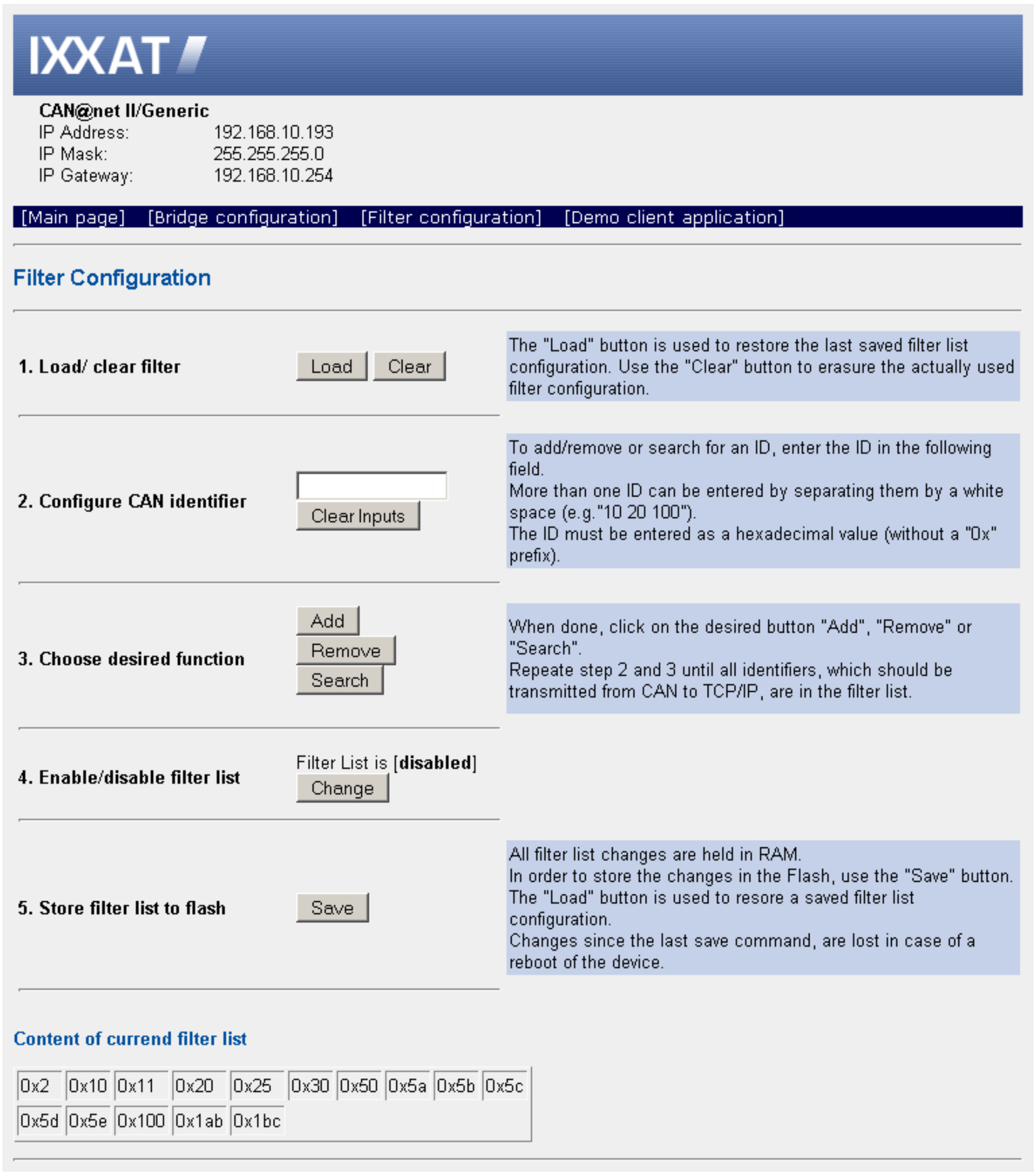
The filter configuration page allows you to modify the settings for the CAN@net II/Generic filtering mechanism as described in chapter 4.3 CAN message filter.

The filter configuration page is shown with screenshot 5.4. When opened, the filter configuration page displays a table with the filter IDs which are currently set.

There are also buttons to add or remove IDs to/from the list or to read or store the complete list from/to Flash. The only field on this page, that can be edited, is the CAN ID field.

All changes to the filter list which have been submitted to the CAN@net II/Generic are held in RAM. And this is the filter list which the CAN@net II/Generic actually applies on received CAN messages. With a power cycle this filter list in RAM will vanish and will be replaced by the filter list previously stored in Flash.

By clicking on the "Save" button, the actual filter list will be stored from RAM to Flash and will be available with the next power cycle. Or, by clicking the "Load" button, the filter list from flash can be restored, overwriting the filter list in RAM.


IXXAT
[Main page] [Bridge configuration] [Filter configuration] [Demo client application]
Load
Clear
Clear Inputs
Add
Remove
Search
Change
Save

CAN@net II/Generic

IP Address:	192.168.10.193
IP Mask:	255.255.255.0
IP Gateway:	192.168.10.254

Filter Configuration

1. Load/ clear filter

The "Load" button is used to restore the last saved filter list configuration. Use the "Clear" button to erasure the actually used filter configuration.

2. Configure CAN identifier

To add/remove or search for an ID, enter the ID in the following field.
More than one ID can be entered by separating them by a white space (e.g."10 20 100").
The ID must be entered as a hexadecimal value (without a "0x" prefix).

3. Choose desired function

When done, click on the desired button "Add", "Remove" or "Search".
Repeate step 2 and 3 until all identifiers, which should be transmitted from CAN to TCP/IP, are in the filter list.

4. Enable/disable filter list

Filter List is [**disabled**]

5. Store filter list to flash

All filter list changes are held in RAM.
In order to store the changes in the Flash, use the "Save" button.
The "Load" button is used to resore a saved filter list configuration.
Changes since the last save command, are lost in case of a reboot of the device.

Content of currend filter list

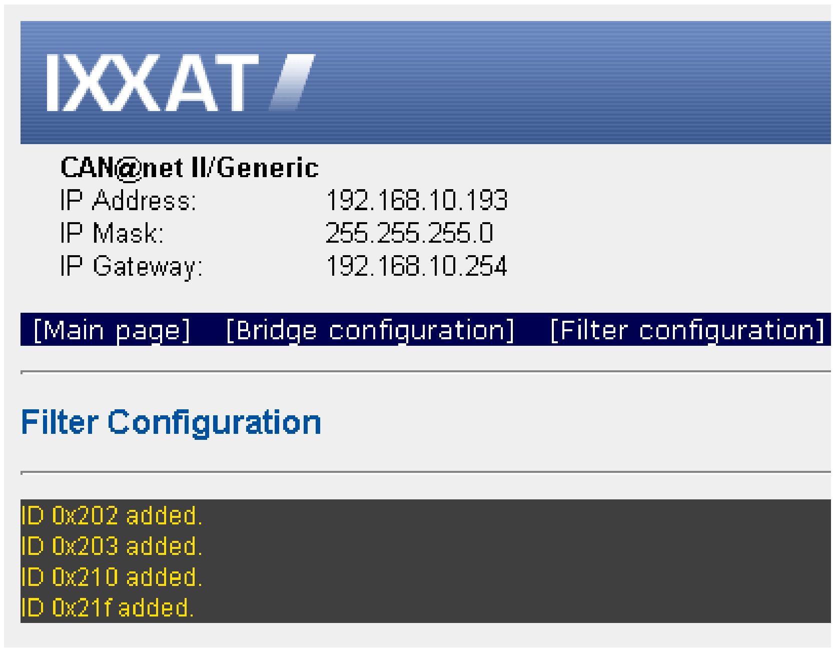
0x2	0x10	0x11	0x20	0x25	0x30	0x50	0x5a	0x5b	0x5c
0x5d	0x5e	0x100	0x1ab	0x1bc					

Screenshot 5.4 Filter configuration

Filter configuration steps:

1. The "Load" button is used to restore the last saved filter list configuration. Use the "Clear" button to erase the actually used filter configuration.
2. To add/remove or search for an ID, enter the ID in the field of step 2. More than one ID can be entered by separating them by a space (e.g."10 20 100"). The ID must be entered as a hexadecimal value (without a "0x" prefix).
3. When done, click on the desired button "Add", "Remove" or "Search". Repeate step 2 and 3 until all identifiers, which should be transmitted from CAN to TCP/IP, are in the filter list. The content of the filter list is shown at the bottom of the page.
4. Enable or disable the filter list. With the filter list disabled, all CAN messages are forwarded by the CAN@net II/Generic, regardless their ID. If the filter list is enable, only the identifier which are in the list are forwarded from CAN to TCP/IP.
5. All filter list changes are held in RAM. In order to store the changes in the Flash, use the "Save" button. Changes since the last save command are lost in case of a reboot of the device.

After submitting the filter configuration formular, the WebServer responds to this by displaying the same configuration page with additional process reply information right under the “Filter configuration” Headline.



CAN@net II/Generic

IP Address:	192.168.10.193
IP Mask:	255.255.255.0
IP Gateway:	192.168.10.254

Filter Configuration

```
ID 0x202 added.
ID 0x203 added.
ID 0x210 added.
ID 0x21f added.
```

Screenshot 5.5 Add CAN ids reply information

6 Writing applications

This chapter gives some hints when writing applications for the ASCII-Protocol and also describes a small demo application.

6.1 TCP streaming

When writing TCP applications in general, the data-stream character of TCP must be considered. Let's have a look at an example:

Imagine, the CAN@net II/Generic receives 3 messages on the CAN bus. After packing these three CAN messages into ASCII-Protocol messages they look as shown in the table below:

M	ED2	20	22	22	\r\n
M	ED2	30	33	33	\r\n
M	ED2	40	44	44	\r\n

Now these three messages will be handed over to the TCP/IP protocol stack using the Socket API's **send()** call. There they will end up concatenated somewhere in the TCP stack's send buffer:

M	**ED2**	**20**	**22**	**22**	**\r\n**	**M**	**ED2**	**30**	**33**	**33**	**\r\n**	**M**	**ED2**	**40**	**44**	**44**	**\r\n**

From now on it is up to the TCP/IP protocol stack to handle these concatenated messages. The TCP/IP will wrap these messages into TCP, then into IP and then into Ethernet frames. Due to size restrictions within the wrapping protocols, it can occur that only part of the original ASCII messages is packed into an Ethernet frame. The following table shows an example. Only the first two and part of the third ASCII-Protocol message are packed into a network message.

M	**ED2**	**20**	**22**	**22**	**\r\n**	**M**	**ED2**	**30**	**33**	**33**	**\r\n**	**M**	**ED2**	**40**	**44**

This is what an ASCII-Protocol client will receive when it makes a Socket API **recv()** call: two complete ASCII-Protocol messages and part of a third. Due to the data-stream character of TCP you can be assured, that the missing part of the third ASCII-Protocol message will arrive eventually and that it can be retrieved with a second **recv()** call. But as shown in the example, the ASCII-Protocol messages can be split across the boundaries of consecutive **recv()** calls.

When writing an ASCII-Protocol application, make sure that the message parser works on complete messages (including the message terminator "\r\n")!

6.2 C Demo application

Provided with this delivery you find a brief demo for an application which acts as an ASCII-Protocol client. It shows how to do initial setup of the system and the basic exchange of CAN data. The demo was written in C using the Microsoft Visual Studio.

For your convenience the source code for the demo application can be retrieved from the CAN@net II/Generic device in file form using the Web Interface. Check the home (or /index) page of the CAN@net II/Generic Web Interface for the "[CAN@net II/Generic demo client application]" link.

The demo does not respect the TCP data-stream character as described in chapter 6.1. It might not work in your environment!

What the demo does:

- In general: Make printf()s to show progress to user.
- Connect to the CAN@net II/Generic [connect_to_cannet()]. The standard IP address and TCP port number for the the CAN@net II/Generic under test can be modified by specifying them with the command line. This will hand the new address to the code via the main(char **argv) function parameters.
- Initialize the CAN controller to 1000 kBaud and start it. This is done with the consecutive commands "c init 1000 \r\n" and "c start \r\n". After the commands each the application waits for a response from the CAN@net II/Generic.
- Disable the filter list with the command "c filter disable \r\n". Now the CAN@net II/Generic will forward all messages from the CAN bus to Ethernet.
- Send three messages via the ASCII-Protocol on the CAN bus. These are the CAN messages with the Identifiers 0, 1 and 2, each having 8 data bytes attached to the message. In order to verify this, an analyzing tool must be connected to your CAN bus.
- Receive three messages on the CAN bus. In order to verify this, an analyzing tool or a CAN sender must be connected to your CAN bus. The CAN messages will be received by the CAN@net II/Generic which sends them as ASCII-Protocol messages to the demo application. Here the messages are displayed using the printf() function.
- Clear the complete filter list, set a single CAN ID to be allowed from CAN to Ethernet and enable the filter mechanism. From now on, only CAN messages with the ID 100 will be forwarded to Ethernet.
- To verify this, an analyzing tool or a CAN sender must be connected to your CAN bus. After receiving the CAN ID 100 three times the demo application prompts this with an OK, stops the CAN controller and exits.

A. Bridge configuration example

Following is an example of a bridge setup as described in chapter 2.2. In the given example, **CAN Bus 1** with baudrate 250 kBd and **CAN Bus 2** with 100 kB will be connected using an Ethernet/TCP/IP network.

Two IP addresses are required for the bridge setup. Please ask your system administrator to provide two IP addresses for your use. In the example, the two IP addresses **192.168.10.100** and **192.168.10.101** are used.

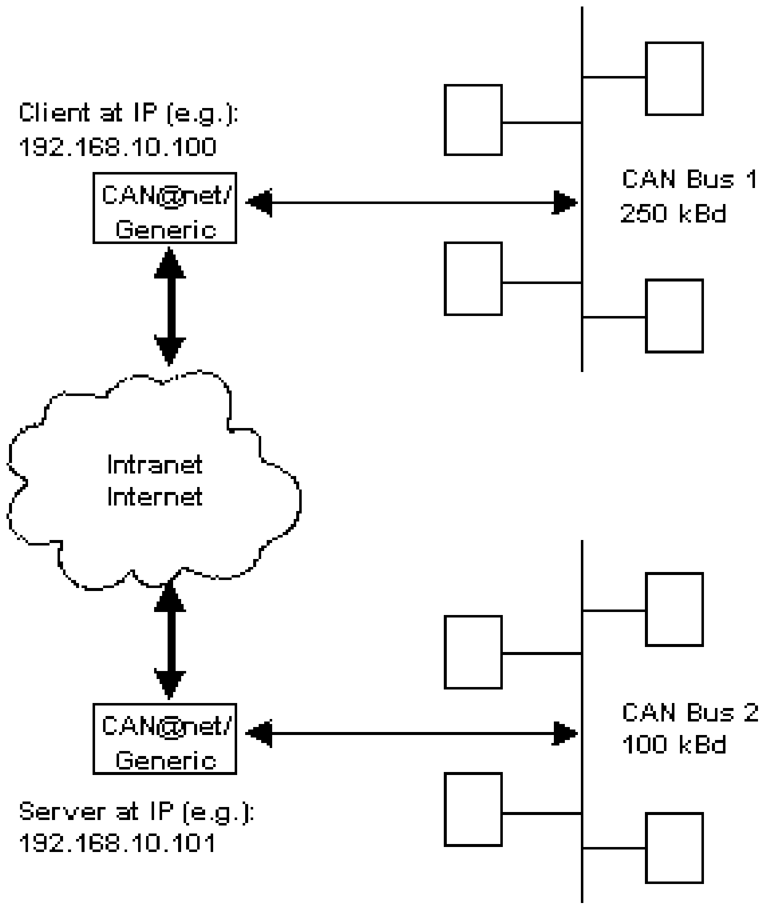


Figure A-1: Example for Bridge configuration

Bridge configuration example

For a functional bridge configuration, the following steps must be done:

(1) Basic configuration of the Server (See Chapter 3.2 Basic device configuration).
The IP setting of the Server should be :
IP address: **192.168.10.101**
Subnet mask: **255.255.255.0**
Gateway address: **0.0.0.0**

(2) Basic configuration of the Client (See Chapter 3.2 Basic device configuration).
The IP setting of the Client should be :
IP address: **192.168.10.100**
Subnet mask: **255.255.255.0**
Gateway address: **0.0.0.0**

(3) Devices must be connected to the Ethernet network and must be reset.

(4) Connect to the Client web interface with any Internet browser with enabled javascript, typing **http://192.168.10.100** in the address field (URL).

(5) Go to the Bridge Configuration page (See Chapter 5) and enter appropriate values in **Configuration client** section (step 2 of the bridge configuration page):
Server device address: **192.168.10.101**
Server baudrate: **100**
Client baudrate: **250**

(6) Store the new values into the device flash memory by entering the device password and pressing the **Submit** button in step 3 and 4 of the bridge configuration page.

(7) Reset or reboot the Client device.

B. FAQ List

Network settings

Q: Which settings are necessary to operate the CAN@net II/Generic in my Ethernet network?
A: At least the IP address for the CAN@net II/Generic must be set. Make sure that the CAN@net II/Generic has its own IP Address; ask your system administrator.
When operating the CAN@net II/Generic across local network boundaries (subnet), also the "Net Mask" and the "Default Gateway" must be set. Ask your system administrator for the correct settings for your local network.

Web Interface

Q: How can I connect to the CAN@net II/Generic Web Interface?
A: By using any Internet Browser, type in the IP address of your CAN@net II/Generic in the URL field of the Internet Browser (example: "http://192.168.10.24") and hit enter.

Q: (1) The Web Browser cannot open the CAN@net II/Generic Web Interface.
A: Make sure the Internet Browser you are using has its proxy settings disabled. With the proxy settings enabled, a URL request to the IP address of the CAN@net II/Generic will end up in the proxy server.

Q: (2) The Web Browser cannot open the CAN@net II/Generic Web Interface.
A: Make sure that the network between your Internet Browser and the CAN@net II/Generic is transparent for the HTTP protocol. This is especially the case when crossing network boundaries with switches or gateways. Please check with your system administrator.

Q: I always get the error message “Error: Wrong password. Access denied.” when I try to change the bridge configuration.
A: Make sure that the your Internet Browser can handle javascripts. Enable javascript in your Internet Browser settings.

Operation

Q: Cannot connect to the ASCII-Protocol server.
A: Make sure that the network between your Internet Browser and the CAN@net II/Generic is transparent for connections to TCP Port 19227. This is especially the case when crossing network boundaries with switches or gateways. Please check with your system administrator.

Q: The ASCII-Protocol server sends many ERR_QUEUE_OVERRUN and ERR_DATA_OVERRUN messages.
A: Every time the CAN@net II/Generic receives a CAN message on the CAN bus and cannot hand it to the ASCII-Protocol server, it sends one of these two error messages. One or more CAN messages will be discarded. The CAN@net II/Generic is not able to follow the data rate of your system.

Q: The CAN@net II/Generic does not send my ASCII-Protocol CAN messages on the CAN bus.
A: Make sure that the CAN controller is initialized with the correct baud rate and that it is started. Check the demo.cpp described in the manual as a simple example.

Q: The CAN@net II/Generic forwards CAN messages from the CAN bus to the Ethernet side with IDs which clearly are not listed in the filter list.
A: Make sure that filtering is enabled. This is done by selecting "enable" on the filter configuration page of the CAN@net II/Generic Web Interface.

Q: The CAN@net II/Generic does not show all messages from the CAN bus on the Ethernet side.
A: Check the current settings in the filter list (if enabled). Only CAN messages with IDs set in the filter list are forwarded to the Ethernet side.

Bridge setup

Q: Data exchange between the two CAN systems is not functional.
A: For the Bridge setup, one CAN@net II/Generic must be configured to act as server (standard configuration) and one CAN@net II/Generic must be configured to act as client. The client can be configured via the Web Interface. For proper operation, the IP address of the server CAN@net II/Generic and the CAN baudrates for both devices must be set correctly.