

Baby-LIN-MB-Tool

V2.11

Lipowsky Industrie-Elektronik GmbH

Römerstraße 57 | 64291 Darmstadt | Germany

Phone: +49 (0) 6151 / 93591 - 0 | Fax: +49 (0) 6151 / 93591 - 28

Website: www.lipowsky.com | E-Mail: info@lipowsky.de

1 Overview	3
1.0.1 Overview	3
2 Search and configure devices	4
3 Communication and test scripts	6
4 Script language	9
5 Baby-LIN-MB simulators	11
6 Logging server and clients	14
7 ASCII host command protocol	15
7.1 Syntax	15
7.2 API modes	16
7.3 CmdDone API	17
7.4 TCP connections: Single and multi socket	18
7.5 Quick reference	19
7.6 Commando List	20
7.6.1 Command Version	20
7.6.2 Command SetApiMode	21
7.6.3 Command CMDDone	21
7.6.4 Command B	22
7.6.5 Command CurrentSdf	22
7.6.6 Command Start	23
7.6.7 Command LinStart	24
7.6.8 Command Stop	24
7.6.9 Commando LinStop	24
7.6.10 Commando LinSchedule	25
7.6.11 Commando SchedMode	25
7.6.12 Command RdSignal	27
7.6.13 Command LinRdSignal	28
7.6.14 Command WrSignal	28
7.6.15 Command LinWrSignal	29
7.6.16 Command VarRead	29
7.6.17 Command VarWrite	31
7.6.18 Command FormatSignals	33
7.6.19 Command LinMstReq	35

7.6.20 Command LinSivResp	36
7.6.21 Command LinState	38
7.6.22 Command RdFrameError	39
7.6.23 Command MacroExec	40
7.6.24 Command Diag22	41
7.6.25 Command RTCWrite	42
7.6.26 Command RTCRead	43
7.6.27 Command ReadById	43
7.6.28 Command ReadByIdCompare	46
7.6.29 Command WaitSignal	48
7.6.30 Command SeqRun	49
7.6.31 Command SeqExec	50
7.6.32 Command Delay	50
7.6.33 Command DigOut	51
7.6.34 Command DigIn	52
7.6.35 Command SetConfigVar	52
7.6.36 Command LicenceInstall	53
7.7 Return codes	55
8 Support information	57

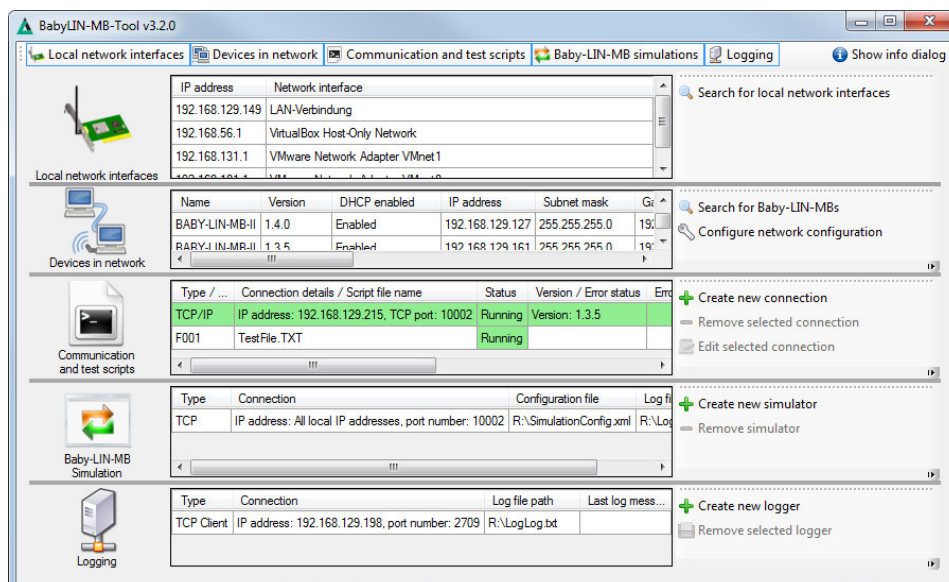
1 Overview

The Baby-LIN-MB-Tool is a tool especially designed to communicate with the Baby-LIN-MB-II. It can be used for the following tasks:

- Search for Baby-LIN-MB-II in your local network.
- Identify a single Baby-LIN-MB-II by enabling a special LED blinking pattern.
- Change the network configuration of the Baby-LIN-MB-II.
- Switch the connection mode.
- Develop, execute and debug test scripts.
- Create a virtual Baby-LIN-MB-II simulator to test your custom application.
- Create different types of loggers, that can log all kind of information from the Baby-LIN-MB-II.

1.0.1 Overview

The Baby-LIN-MB-Tool is used to find, configure and test Baby-LIN-MB devices and host software. Subsequently all mentions of the Baby-LIN-MB refer to the BabyLIN-MB-I as well as the BabyLIN-MB-II, as long as not specified explicitly otherwise.



The software consists of 5 main sections:

Section	Description
Local network interfaces	In this section all your local network cards are shown. This is mainly useful if you have separated networks each with Baby-LIN-MB devices connected.
Devices in network	In this section you can search for and configure Baby-LIN-MB devices. The table shows all the Baby-LIN-MB devices, that could be found on any of your local network interfaces.
Communication and test scripts	In this section you can communicate with the Baby-LIN-MB devices and execute test scripts. The table shows you all devices you are communicating with and the test scripts you are executing.
Baby-LIN-MB Simulation	In this section you can configure and run simulators, that can be configured to simulate certain Baby-LIN-MB behaviors. This can be used to test certain scenarios of your host application. The table shows you all configured simulators.
Logging	In this section you can configure and run different kind of loggers, that can be configured to log certain outputs of the Baby-LIN-MB devices. The table shows you all loggers.

The top toolbar allows you to hide sections you do not need. The remaining sections can be resized by grabbing and moving the splitter bars between the sections.

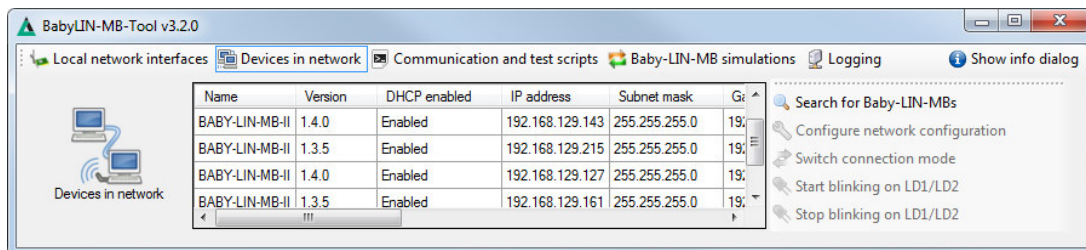
The configuration of the Baby-LIN-MB-Tool consists of the following items:

- The settings, which section is visible.
- All communication and test scripts.
- All simulators.
- All loggers.

The configuration can be saved, loaded and cleared by the buttons in the top right toolbar. Additionally the current configuration is saved automatically and loaded again when the application is starting the next time.

2 Search and configure devices

In this section you can search for Baby-LINs in your local network and configure their network configuration.



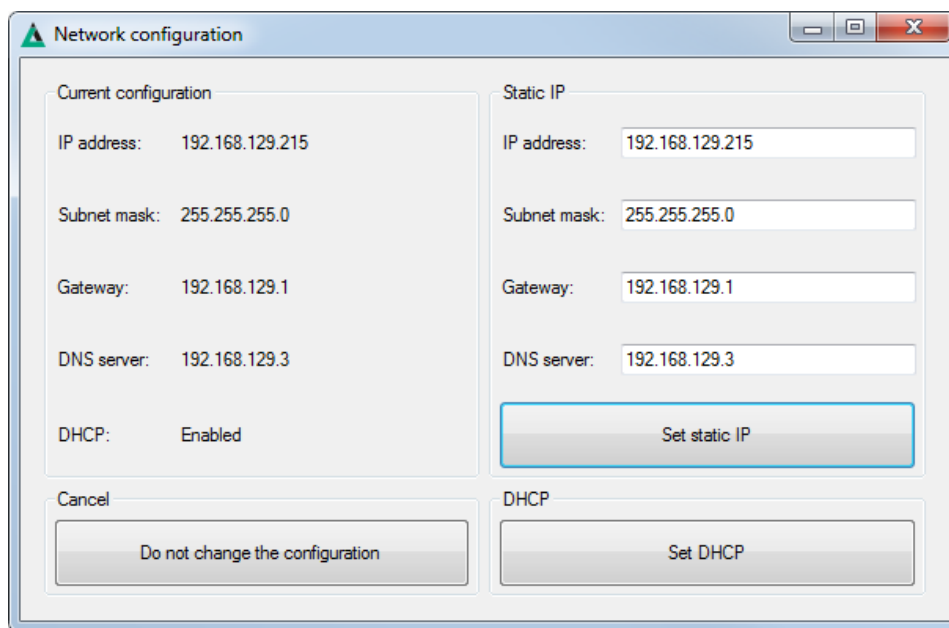
The table shows you all Baby-LIN-MB devices that could be found over your local network interfaces. Some information may be displayed as "Unknown" if the device or firmware is old.

Column name	Description
Name	This is the name of the device.
Version	This is the firmware version.
DHCP enabled	Is the device receiving its network configuration from a DHCP server.
IP address	The IP address of the device
Subnet mask	The subnet mask of the device
Gateway	The gateway address of the device
MAC address	The MAC address of the network interface hardware of the device.
Connection mode	The connection mode has the following options: <ul style="list-style-type: none"> • Stand-alone: The device can be controlled using the script language over a serial or network connection. • SimpleMenu: The device can be controlled by the SimpleMenu like other Baby-LIN devices. • Unknown: Since the firmware is old the device will be in the Stand-alone mode.
Connection mode duration	The connection mode duration indicates, if the connection mode is statically set or if it will change after the next restart.
Serial number	This is the serial number of the device.
Hardware revision	This is the hardware revision of the device.
Web	By clicking the icon the web interface is opened in your default browser.
Local IP address	This is the IP address of the local network interface the device is connected to.

The following actions are available via buttons:

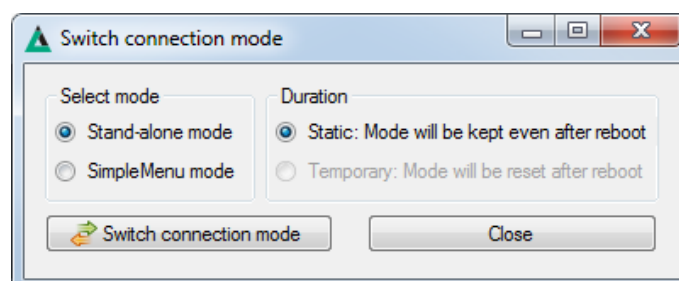
Button text	Description
Search for Baby-LIN-MBs	This button will init a new search for Baby-LIN-MBs and refresh the table.
Configure network configuration	This button will open a dialog in which you can change the network configuration of the device.
Switch connection mode	This button will open a dialog to switch the connection mode and its duration.
Start blinking on LD1/LD2	After pressing this button the LEDs LD1 and LD2 will start blinking alternately.
Stop blinking on LD1/LD2	After pressing this button the LEDs LD1 and LD2 will stop blinking alternately and continue their normal behavior.

With this dialog you can change the network configuration of the device.



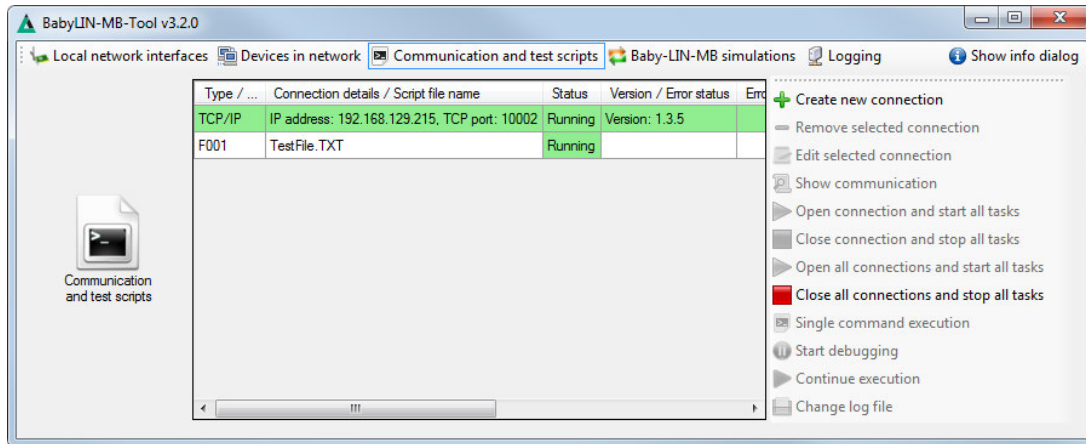
You can either enter the static IP data and commit the changes by pressing the "Set static IP" button or you simply press the "Set DHCP" button. In the latter case the network configuration will be received from a DHCP server.

The connection mode and its duration can be changed using this dialog.



3 Communication and test scripts

In this section you can configure connections to Baby-LINs and execute test scripts.



The table shows you all configured devices and test scripts no matter if running or not. A device line uses the following column headers:

Column name	Description
Type	The type of connection to the device. The following type are possible: <ul style="list-style-type: none"> • TCP/IP: The device has to be connected using the RJ-45 connector. • Serial: The device has to be connected using the Sub-D-9 male connector. • No connection: No connection is established. This is only useful if you have a script without any commands (starting with "C:").
Connection details	These are details about the connection configuration.
Status	The status of the communication: <ul style="list-style-type: none"> • Not started • Running • Not running
Version	The firmware version that is requested at the start of the communication.
Error code	Not used.
Start time	Not used.
Stop time	Not used.
Runtime	Not used.
Log file	The path to the log file
Last log message	The last log message of the device communication or any script file.

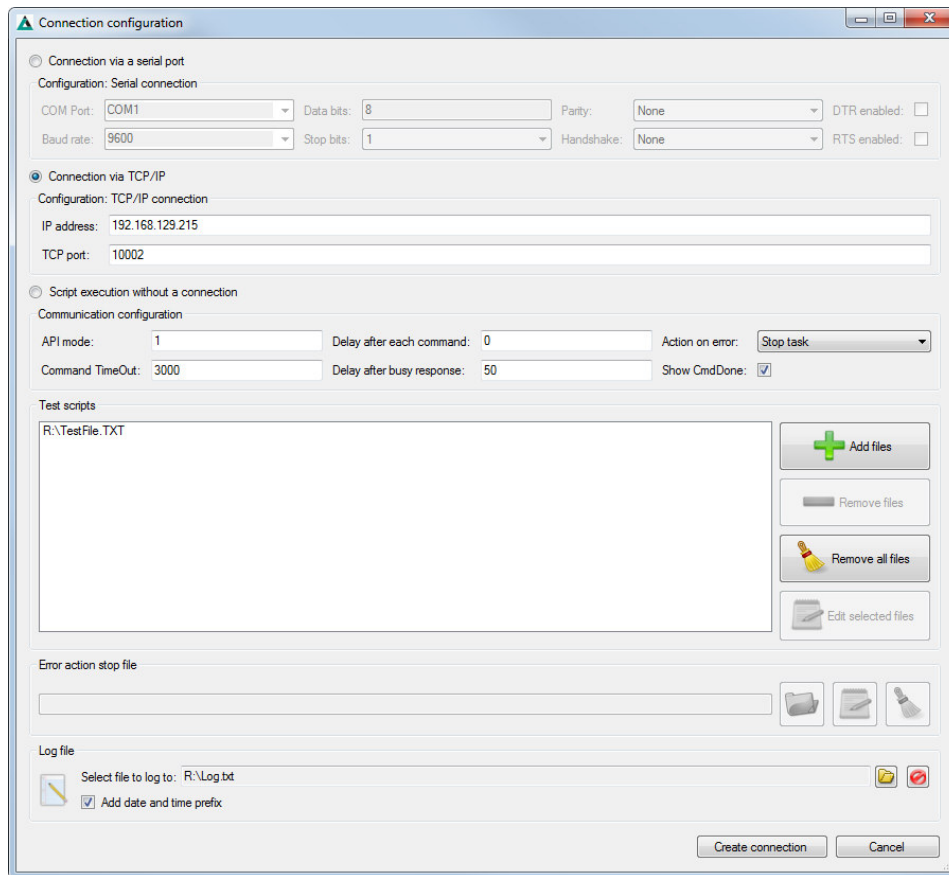
A script file line uses the following column headers:

Column name	Description
ID	The ID of the script file. This is a consecutive number.
Script file name	The name of the script file
Status	The status of the communication <ul style="list-style-type: none"> • Not started • Running • Not running • Paused
Error status	If an error or special event occurred, it shows the status of the script file execution: <ul style="list-style-type: none"> • Error occurred, but task goes on • Error occurred, task is stopped • Error occurred, all tasks are stopped • No error, another task had error • Stopped by user • Connection is lost • File finished
Error code	If an error occurred, it shows the error number and an error message.
Start time	The time the execution started.
Stop time	The time the execution stopped.
Runtime	The time the execution was running.
Script file path	The path to the script file.
Last log message	The last log message of this file's execution.

The following actions are available via buttons:

Button text	Description
Create new connection	This button will open a new dialog in which you can configure a new connection and choose the script files, that will be executed.
Remove selected connection	This button will stop and remove the selected communication and all its scripts.
Edit selected connection	This button will open a dialog in which you can edit the connection and the script files.
Show communication	This button will open a log window. It will show you all log messages depending on which line was selected: <ul style="list-style-type: none"> • Device line: Show log messages of the complete communication and all scripts. • Script file line: Show log messages only of the selected scripts.
Open connection and start all tasks	This button will initiate the communication and all scripts of the selected device.
Close connection and stop all tasks	This button will close the communication and stop all scripts of the selected device.
Open all connections and start all tasks	This button will initiate all the configured communications and all their scripts.
Close all connections and stop all tasks	This button will close all the configured communications and all their script.
Single command execution	This button will open a dialog in which you can send single commands while the communication is established.
Start debugging	This button will open the debugger window in which you can set breakpoints and execute each script line step by step.
Continue execution	If a script was paused by the pause command (x:pause), by pressing this button it will continue.
Change log file	This button will let you change the log file.

With this dialog you can create and edit the communication with a device.



First choose a connection type (Serial, TCP/IP, no connection) and configure it.

Then you can change the configuration of the communication:

Setting	Description
API mode	Here you can choose the API mode: Check chapter "API modes" for more information.
Commend TimeOut	This is the maximum number of milliseconds a command may endure. After this time an error is created.
Delay after each command	This is a general delay (in milliseconds), that is waited after each command.
Delay after busy response	This is the delay (in milliseconds), that is waited after a command returned a busy response.
Action on error	This defines what will happen, if an error occurs: Go on: The script execution continues. Stop task if error is not handled: The script execution continues if the error is handled by a Jump if positive/negative command Stop task: The script execution is stopped Stop all tasks: The script execution of all scripts of the device is stopped. Stop all tasks and execute stop file: The script execution of all scripts of the device is stopped and the stop file script is executed.
Shwo CmdDone	If checked all busy responses and CmdDone commands are shown in the log.

Finally you can select the script files, the stop script file if required and the log file.

This window shows you all the log messages of a device and all its scripts. With the command execution window you can enter a command that is sent to the device. An established connection is required.

```

Device: TCP/IP - IP address: 192.168.129.215, TCP port: 10002

[28.03.17, 13:40:21:0604] F001 Sleeping for 500 ms
[28.03.17, 13:40:21:5755] F001 [L003] j:-2
[28.03.17, 13:40:21:5794] F001 Jumping, Offset: -2
[28.03.17, 13:40:21:5844] F001 [L001] c:version
[28.03.17, 13:40:21:5894] [=> Sending:] F001 :version
[28.03.17, 13:40:21:6423] [<= Received:] F001 :T62991
[28.03.17, 13:40:21:6453] F001 Received token: 62991
[28.03.17, 13:40:21:6972] [=> Sending:] F001 :CmdDone 62991
[28.03.17, 13:40:21:7501] [<= Received:] F001 :1.3.5
[28.03.17, 13:40:21:7531] F001 Received answer: :1.3.5
[28.03.17, 13:40:21:7541] F001 [L002] d:500
[28.03.17, 13:40:21:7571] F001 Sleeping for 500 ms
[28.03.17, 13:40:22:2692] F001 [L003] j:-2
[28.03.17, 13:40:22:2721] F001 Jumping, Offset: -2
[28.03.17, 13:40:22:2732] F001 [L001] c:version
[28.03.17, 13:40:22:2772] [=> Sending:] F001 :version
[28.03.17, 13:40:22:3331] [<= Received:] F001 :T63004
[28.03.17, 13:40:22:3381] F001 Received token: 63004
[28.03.17, 13:40:22:3900] [=> Sending:] F001 :CmdDone 63004
[28.03.17, 13:40:22:4519] [<= Received:] F001 :1.3.5
[28.03.17, 13:40:22:4569] F001 Received answer: :1.3.5
[28.03.17, 13:40:22:4609] F001 [L002] d:500
[28.03.17, 13:40:22:4748] F001 Sleeping for 500 ms
[28.03.17, 13:40:22:9969] F001 [L003] j:-2
[28.03.17, 13:40:23:0029] F001 Jumping, Offset: -2
[28.03.17, 13:40:23:0059] F001 [L001] c:version
[28.03.17, 13:40:23:0109] [=> Sending:] F001 :version
[28.03.17, 13:40:23:0678] [<= Received:] F001 :T63014
[28.03.17, 13:40:23:0748] F001 Received token: 63014
[28.03.17, 13:40:23:1366] [=> Sending:] F001 :CmdDone 63014
[28.03.17, 13:40:23:1905] [<= Received:] F001 :1.3.5
[28.03.17, 13:40:23:1945] F001 Received answer: :1.3.5
[28.03.17, 13:40:23:2005] F001 [L002] d:500
[28.03.17, 13:40:23:2045] F001 Sleeping for 500 ms
    
```

```

Command execution

version

[28.03.17, 13:37:57:6345] Received answer: :1.3.5
[28.03.17, 13:37:57:6335] [<= :1.3.5
[28.03.17, 13:37:57:5217] => :CmdDone 60495
[28.03.17, 13:37:57:5217] Received token: 60495
[28.03.17, 13:37:57:5207] [<= :T60495
[28.03.17, 13:37:57:4658] => :version
    
```

In the Debugger window you can set break points and execute the script step by step for easier troubleshooting

Debugger: F001 - MBExample.txt

Execute current line | Jump to previous line | Jump to next line | Continue execution | Continue execution and close debugger | Stop execution and close debugger

Ignore empty lines

Breakpoint	Line	Content
<input type="checkbox"/>	001	C:Version
<input type="checkbox"/>	002	C:LoadSdf 0 ExampleSdf.sdf
<input type="checkbox"/>	003	D:500
<input checked="" type="checkbox"/>	004	C:LinState 0
<input type="checkbox"/>	005	C:LinStart 0
<input type="checkbox"/>	006	C:LinWrSignal 0 TestSignal 30
<input type="checkbox"/>	007	C:LinRdSignal 0 TestSignal
<input type="checkbox"/>	008	P:-4
<input type="checkbox"/>	009	C:LinStop 0

4 Script language

A script file contains a sequence of commands, which will be sent to the Baby-LIN-MB. Comments are declared with a leading ";" character. Therefore everything in a line behind a ';' character will be omitted.

Each line starts with a character defining the type:																													
C:	Defines a command line. The command is sent to the Baby-LIN-MB which will execute it. Please check chapter "ASCII host command protocol" for a detailed command overview.																												
D:	Defines a delay. The execution of the script is stopped for the given time span [in ms].																												
L:	Defines a label. Labels can be used as target for jump commands.																												
J:	Defines a jump.																												
P:	Defines a jump that will only be executed, if the previous command was successful.																												
N:	Defines a jump that will only be executed, if the previous command has failed.																												
<p>All jump commands support two kind of targets:</p> <ul style="list-style-type: none"> • Relative jumps: The jumping distance is relative to the jump command. <code>C:Version</code> ; Empty line <code>J:-1 ; jump to version command</code> • Absolute jumps to labels: The target is a label. <code>L:label</code> <code>C:version</code> <code>J:label</code> 																													
S:	Defines a synchronization. If the script contains at least one synchronization statement the file will be executed in the synchronous mode. In this mode a command will not be executed instantly but its execution will be delayed until the next synchronization command. This can be useful if one file controls multiple channels.																												
X:	This prefix controls the execution of the file. The following commands are defined:																												
X:exit	Defines an exit. The execution of the script is stopped immediately.																												
X:pause	Defines a pause. The execution of the script is paused until the user is pressing the continue button.																												
X:break	Defines a break. The execution of the script is paused and the debugger window will be opened automatically. Within this window the script can be executed step by step.																												
X:config	<p>The X:config command allows to configure the communication settings from within a script:</p> <table border="1"> <thead> <tr> <th>Parameter 1</th> <th>Parameter 2</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>timeout</td> <td>Value</td> <td>The time-out value in [ms].</td> </tr> <tr> <td>commanddelay</td> <td>Value</td> <td>The delay after each command in [ms].</td> </tr> <tr> <td>busydelay</td> <td>Value</td> <td>The delay after a busy response in [ms].</td> </tr> <tr> <td rowspan="5">erroraction</td> <td>0</td> <td>Go on</td> </tr> <tr> <td>1</td> <td>Stop task if error is not handled</td> </tr> <tr> <td>2</td> <td>Stop task</td> </tr> <tr> <td>3</td> <td>Stop all tasks</td> </tr> <tr> <td>4</td> <td>Stop all tasks and execute stop file.</td> </tr> <tr> <td rowspan="2">showcmddone</td> <td>0</td> <td>Off. The messages of the CmdDone API are not shown.</td> </tr> <tr> <td>1</td> <td>On. The messages of the CmdDone API are shown.</td> </tr> </tbody> </table> <div style="background-color: yellow; padding: 5px; margin-top: 10px;"> <p>Warning</p> <ul style="list-style-type: none"> • Changes made to the communication settings are not stored. If the scripts are restarted the original settings are restored. • The changes to the communication settings from within one script will affect all other scripts too, since they share one instance of settings. </div>	Parameter 1	Parameter 2	Description	timeout	Value	The time-out value in [ms].	commanddelay	Value	The delay after each command in [ms].	busydelay	Value	The delay after a busy response in [ms].	erroraction	0	Go on	1	Stop task if error is not handled	2	Stop task	3	Stop all tasks	4	Stop all tasks and execute stop file.	showcmddone	0	Off. The messages of the CmdDone API are not shown.	1	On. The messages of the CmdDone API are shown.
Parameter 1	Parameter 2	Description																											
timeout	Value	The time-out value in [ms].																											
commanddelay	Value	The delay after each command in [ms].																											
busydelay	Value	The delay after a busy response in [ms].																											
erroraction	0	Go on																											
	1	Stop task if error is not handled																											
	2	Stop task																											
	3	Stop all tasks																											
	4	Stop all tasks and execute stop file.																											
showcmddone	0	Off. The messages of the CmdDone API are not shown.																											
	1	On. The messages of the CmdDone API are shown.																											

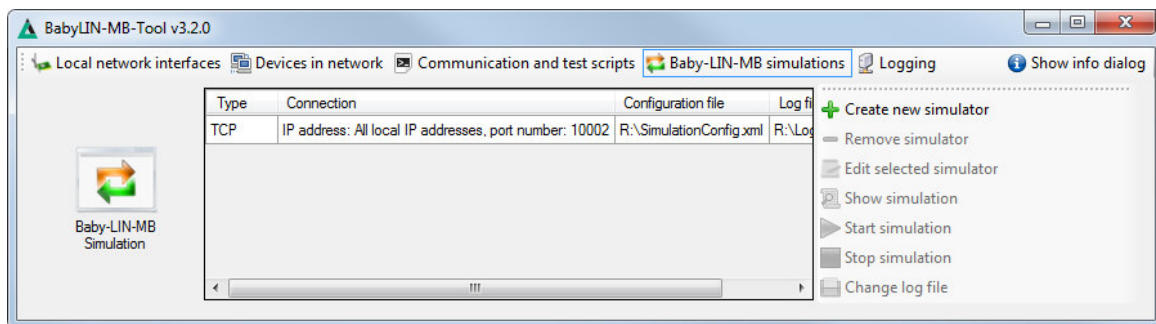
Each line starts with a character defining the type:	
X:checkconnection	Defines a query that checks the connection state. The result can be evaluated with a P/N command.
X:reconnect	Establishes a new connection to the device if the connection was lost.
X:evaluate	With this command you can evaluate the response of a command by using a regular expression. Afterwards you can use the P/N command. The following example queries the version of a device and stops if the version is not 1.4.0: c:version x:evaluate ":1.4.0" p:+2 x:exit

The following example shows a typical script:

```
C:Version
C:LoadSdf 0 ExampleSdf.sdf
D:500
C:LinState 0
C:LinState 0
C:LinWrSignal 0 !TestSignal 30
C:LinRdSignal 0 !TestSignal
P:-4
C:LinStop 0
```

5 Baby-LIN-MB simulators

In this section you can configure Baby-LIN-MB simulators, that simulate certain behaviors of the Baby-LINMB communication.



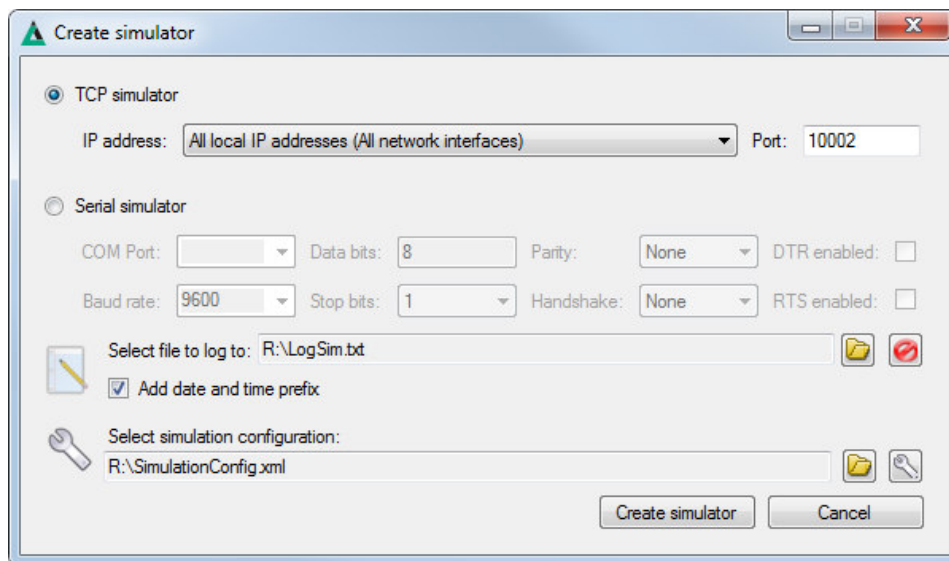
The table shows you all configured Baby-LIN-MB simulators.

Columns name	Description
Type	The connection type this simulator expects. Possible values are: <ul style="list-style-type: none"> • TCP simulator • Serial simulator
Connection	here you can find the connections details
Configuration file	This is the configuration file, that is used to define the behavior of the simulator.
Log file path	This is the path to the log file
Last log message	This is the last log message of the simulator

The following actions are available via buttons:

Button text	Description
Create new simulator	This button will open a new dialog in which you can configure a new simulator and its behavior.
Remove simulator	This button will stop and remove the selected simulator.
Edit selected simulator	This button will open a dialog in which you can edit the simulator and its behavior.
Show simulation	This button will open a log window. It will show you all log messages of the simulator.
Start simulation	This button will start the simulator.
Stop simulation	This button will stop the simulator.
Change log file	This button will let you change the log file.

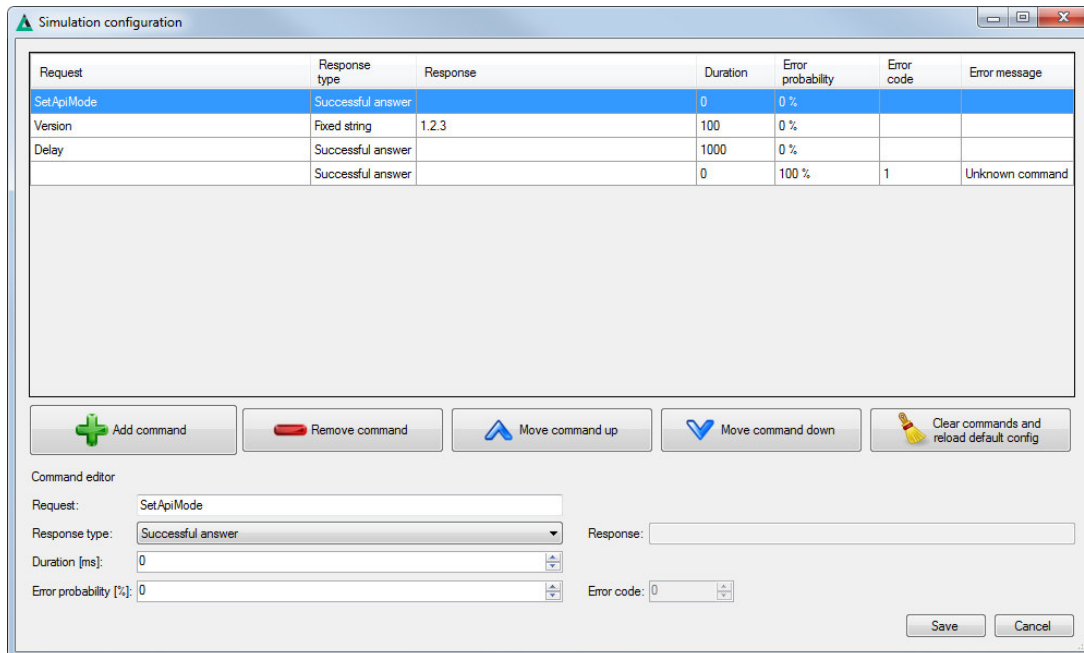
With this dialog you can create and edit Baby-LIN-MB simulators.



First configure the connection the simulator will handle. At the bottom you can select a simulation configuration file and edit it, by pressing the wrench button.

If you have no configuration file simply set the path where you want the file to be created and press the wrench button.

This will open the following dialog.



In this dialog you can edit the commands you want the simulator to react to. If your configuration is empty, you can load a minimal configuration by pressing the "Clear commands and reload default config" button.

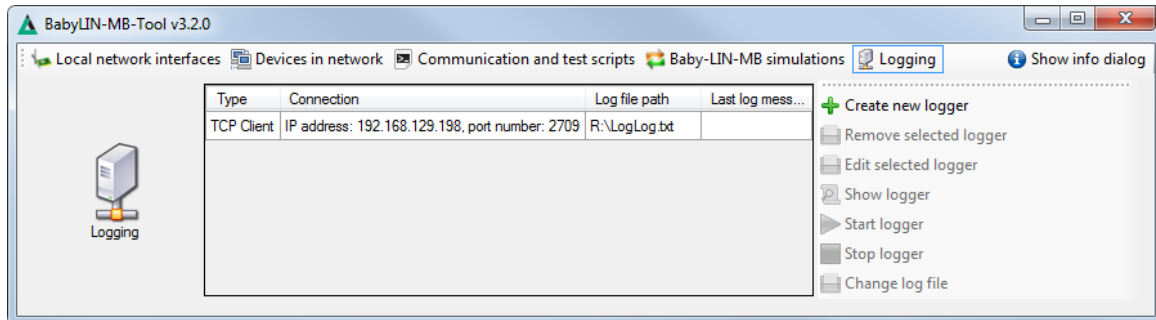
The table contains all configured commands. Each line represents a request pattern, which is compared to a received command. If the request pattern matches, the simulator responds with the defined response. The request patterns will be evaluated from the beginning to the end. The first match will be used. An empty request pattern will always match.

A command contains the following components:

Command component	Description
Request	This string is compared to the beginning of a received command. If the received command starts with this string, its response is sent back.
Response type	The type of answer, that is returned to the sender. The following types are available: <ul style="list-style-type: none"> Fixed string: You can freely define a string, that is returned. Successful answer: The answer is :0, which represents a successful execution of the command. Up counting number: A number is returned, that is counted up for each command.
Response	If the response type is "Fixed string", this string will be returned as answer.
Duration	This is the time the command will delay its response depending on the API mode: <ul style="list-style-type: none"> API mode 0: The response is held back for the duration. API mode 1: The response will be a token that needs to be polled by sending CmdDone commands. The command will be busy until the duration has elapsed.
Error probability	This is the probability the command will respond with an error. This can be useful to test the error handling of a host. <ul style="list-style-type: none"> Probability = 0: The command will never return an error. Probability = 1-99: A random number generator will decide if the command succeeds or an error is returned. Probability = 100: The command will always return an error.
Error code	This is the code a command returns if an error is raised.

6 Logging server and clients

In this section you can configure logging servers and clients to receive log information from the Baby-LIN-MB.



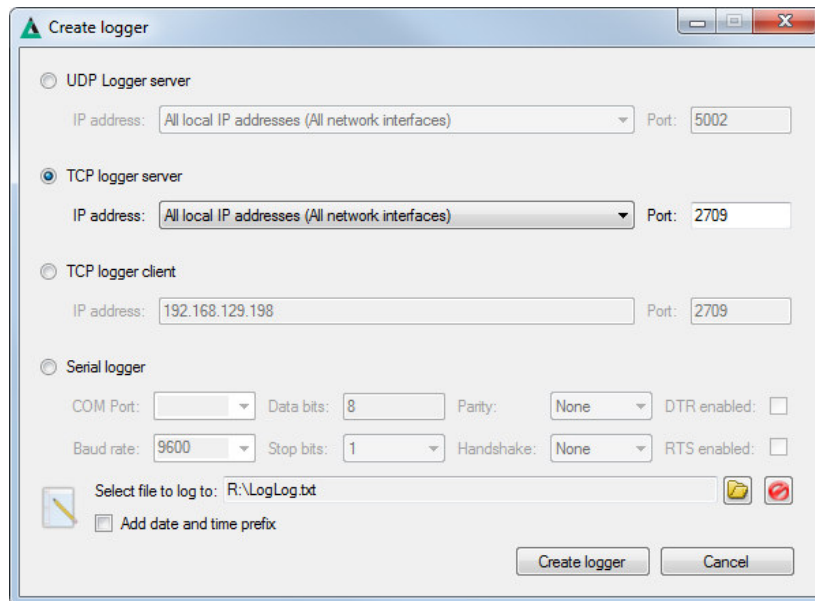
The table shows you all configured logger.

Column name	Description
Type	The connection type this logger is using. Possible values are: <ul style="list-style-type: none"> • UDP logger server • TCP logger server • TCP logger client • Serial logger
Connection	Here you can find the connection details.
Log file path	This is the path to the log file.
Last log message	This is the last log message received by the logger

The following actions are available via buttons:

Button text	Description
Create new logger	This button will open a new dialog in which you can configure a new logger.
Remove selected logger	This button will stop and remove the selected logger.
Edit selected logger	This button will open a dialog in which you can edit the logger.
Show logger	This button will open a log window. It will show you all log messages of the logger.
Start logger	This button will start the logger.
Stop logger	This button will stop the logger.
Change log file	This button will let you change the log file.

With this dialog you can create and edit a logger.



The following types of loggers are available.

Logger type	Description
UDP logger server	This logger opens a UDP server and waits for telegrams from a client. Each telegram is one line within the log file. This logger can be used to receive the messages from Baby-LIN-MB-II's UDP push logger plugin.
TCP logger server	This logger listens as server to TCP connections from a client. Any new line character will be used to extract a line for the log view. The log file will contain the original new line characters.
TCP logger client	This logger opens a TCP connection as client. Any new line character will be used to extract a line for the log view. The log file will contain the original new line characters. This logger can be used to receive the debug messages from the Baby-LIN-MB-II's Debug log server on port 2709.
Serial logger	This logger opens a serial connection. Any new line character will be used to extract a line for the log view. The log file will contain the original new line characters.

7 ASCII host command protocol

7.1 Syntax

The ASCII host command protocol is an ASCII based protocol to communicate with the Baby-LIN-MB-II. This protocol can be used via the "X10 - Ethernet" or "X7 - RS-232" connectors. This protocol is mainly used for the communication with a PLC, but other devices able to process ASCII communication can be used as well, e.g. PCs.

A command always starts with a colon : (ASCII code: 0x3A) followed by the name of the command. Depending on the command a list of parameters can follow, that are separated by a blank (ASCII code: 0x20). The command is always terminated with a carriage return (ASCII code: 0x0D).

```
:MacroExec 0 1 1 5000<CR>
```

The result also starts with a colon and is terminated with a carriage return. In between is the result of the command

```
:0<CR>
```


Numbers can be entered as decimal or hexadecimal values:

Number format	Description	Example
Decimal	Decimal numbers can be written without formatting characters.	42
Hexadecimal	Hexadecimal numbers can be written by adding a leading H character (ASCII code: 0x48).	2AH

Signals can always be referenced by the signal index or the signal name:

Number format	Description	Example
Signal index	Signal indices can be written without formatting characters.	15
Signale name	Signal names can be written by adding a leading ! character (ASCII code: 0x21).	!Ignation

7.2 API modes

The Baby-LIN-MB-II supports different API modes to allow efficient processing as well as backwards compatibility.

API mode value	0	1	2
API mode name	Immediate API	CmdDone API	Compatibility API
History	The Baby-LIN-MB only supported one LINBus channel. Since no parallel processing was required, a command blocked until it finished. After the reception of a response the next command could be send by the client, e.g. a PLC. The immediate API simulates this mode, even though it is not optimal to handle multiple channels.	The Baby-LIN-MBII introduced multiple channels. The blocking commands prevented the efficient operation of multiple channels. This is now possible with the introduction of none blocking commands. Since a command may return, before it has finished, the execution state has to be polled.	Over time certain differences between the Immediate API and the command handling of the Baby-LIN-MBwere found. Since many new customers used the Baby-LIN-MB-II
Usage	Use this mode, if your application was developed for a Baby-LIN-MB-II and you do not want to handle execution states of commands and busy tokens.	Use this mode, if your application was developed for a Baby-LIN-MB-II and you want to handle multiple channels efficiently.	Use this mode, if your application was developed for a Baby-LINMB and you want to replace the device with a Baby-LIN-MB-II.
Immediate execution	✓	Only some commands return immediately.	✓
Busy tokens		✓	
Compatible with the Baby-LIN-MB	Most commands, but not all.		✓

The differences between the Immediate API and the Compatibility API are minimal, but could break a working Baby-LIN-MB application. The following commands are affected:

- "Command Version"
- "Command ReadByld"
- "Command ReadByldCompare"

The Baby-LIN-MB-II starts by default in the Immediate API.

7.3 CmdDone API

Since the Baby-LIN-MB-II supports more than one channel, the Immediate API and Compatibility API can not be used efficiently. Commands executed in these API modes will block and not return until the command is finished. A parallel control of multiple channels can only be achieved by the CmdDone API.

Using this API mode, a command can either return a result immediately or it returns a token. This token then has to be queried to check if a command is finished and to finally receive the result. That way commands can be send to multiple channels and the execution happens parallel.

The token response can be identified by a leading "T" character in the response.

Direction	Command / Response	Description
Client → Baby-LIN-MB-II	:<Command Par1 ... ParX><CR>	A command is sent to the Baby-LIN-MB-II
Baby-LIN-MB-II → Client	:T<token><CR>	The Baby-LIN-MB-II starts the execution and returns a token.
Client → Baby-LIN-MB-II	:CmdDone <token><CR>	The client queries the execution state of the command.
Baby-LIN-MB-II → Client	:B<CR>	The Baby-LIN-MB-II indicates, that is still busy executing the command.
Client → Baby-LIN-MB-II	:CmdDone <token><CR>	The client queries the execution state of the command.
Baby-LIN-MB-II → Client	:XXX<CR>	The Baby-LIN-MB-II returns the result, since the execution is finished. XXX : result if the command succeeded. YYY : error code if the command failed.
	:@YYY<CR>	

The Baby-LIN-MB-II starts by default in the Immediate API. To efficiently communicate with multiple channels, the CmdDone API must be activated. Then one command per channel can be executed parallel.

Direction	Command / Response	Description
Client → Baby-LIN-MB-II	:MacroExec 0 1 1 5000<CR>	A command is sent to the first channel.
Baby-LIN-MB-II → Client	:T4711<CR>	The Baby-LIN-MB-II starts the execution and returns a token.
Client → Baby-LIN-MB-II	:MacroExec 1 1 1 5000<CR>	A command is sent to the second channel.
Baby-LIN-MB-II → Client	:T4713<CR>	The Baby-LIN-MB-II starts the execution and returns a token.
Client → Baby-LIN-MB-II	:CmdDone 4711<CR>	The state of the command execution for the first command is queried.
Baby-LIN-MB-II → Client	:B<CR>	The Baby-LIN-MB-II indicates it is still busy executing the command.
Client → Baby-LIN-MB-II	:CmdDone 4712<CR>	The state of the command execution for the second command is queried.
Baby-LIN-MB-II → Client	:0<CR>	The second command finished and succeeded.
Client → Baby-LIN-MB-II	:CmdDone 4713<CR>	The state of the command execution for the third command is queried.
Baby-LIN-MB-II → Client	:@12<CR>	The third command finished and failed.
Client → Baby-LIN-MB-II	:CmdDone 4711<CR>	The state of the command execution for the first command is queried.
Baby-LIN-MB-II → Client	:0<CR>	The first command finished and succeeded.

7.4 TCP connections: Single and multi socket

The current firmware of the Baby-LIN-MB-II allows two different connection modes:

The single socket mode uses a single TCP connection to control all channels of the Baby-LIN-MB-II. The target channel of channel specific commands is identified by a channel parameter in the command.

The multi socket mode uses an individual TCP connection to control each channel individually. The target channel of channel specific commands is identified by the TCP port of the connection. The channel parameter of the channel specific commands must be omitted.



Warning

The switch from one connection mode to the other requires changes in the command parameters, since the multi socket mode does not use the channel parameter.

The following table sums up the properties of the connection modes:

Property	Signal socket connection	Mult socket connection
TCP ports and channel access	10002for all channels.	10003 and folloing: <ul style="list-style-type: none"> • 10003: LIN 1 • 10004: LIN 2 • 10005: LIN 3 • 10006: LIN 4 • 10007: LIN 5 • 10008: LIN 6 • 10009: CAN-HS • 10010: CAN-FD 1 • 10011: CAN-FD 2



Advice

The default single socket TCP port can be changed in the settings. The multi socket TCP ports are always the nine following ports. Check chapter "Registry variables" for more information.

Property	Signal socket connection	Mult socket connection
API mode	<ul style="list-style-type: none"> • 0: Immediate API • 1 : CmdDone API • 2 : Campatibility API <p>If multiple channels should be controlled simultaneously, the CmdDone API must be used.</p> <p>Check chapter "API modes" for more information.</p>	<ul style="list-style-type: none"> •0: Immediate API • 2 : Campatibility API <p>Since each channel has its own connection, there is no need for the CmdDone API.</p>
Channel parameter	All commands, that target a certain channel, have an explicit channel parameter.	The channel parameter of the channel specific commands must be omitted.

7.5 Quick reference

The following table gives you an overview over the available commands. Not all commands are available for all versions of the device. Check the "Supported by" columns. Some commands do not require a channel, while others do. Check the "Required channels" column:

- Empty: No channel is required. The command applies to the device as a whole.
- LIN: This command only works with a LIN channel.
- CAN: This command only works with a CAN channel.
- LIN, CAN: This command works with a LIN or CAN channel.

Command (and alias)	Description	MB	MB-II	Required channels	Link
Version	Return the firmware version of the device.	✓	✓		"Command Version"
SetApiMode	Switch between immediate and CmdDone API.		✓		"Command SetApiMode"
CmdDone	Query command token in CmdDone API.		✓		"Command CmdDone"
B	Set the baudrate of the serial connection.	✓			"Command B"
LoadSdf	Load SDFFile	✓	✓	LIN,CAN	"Command LoadSdf"
CurrentSdf	Check if any SDFFile was loaded to a channel.		✓	LIN,CAN	"Command CurrentSdf"
Start LinStart	Start the simulation.	✓	✓	LIN,CAN	"Command Start" "Command LinStart"
Stop StopStart	Start the simulation.	✓	✓	LIN,CAN	"Command Stop" "Command LinStop"
LinSchedule	Select and run a LIN schedule.	✓	✓	LIN	"Command LinSchedule"
SchedMode	Select a schedule operation mode.		✓	LIN	"Command SchedMode"
RdSignal LinRdSignal	Read a signal value.	✓	✓	LIN,CAN	"Command RdSignal" "Command LinRdSignal"
WrSignal LinWrSignal	Write a signal value.	✓	✓	LIN,CAN	"Command WrSignal" "Command LinWrSignal"
VarRead	Reads multiple signal values.		✓	LIN,CAN	"Command VarRead"
VarWrite	Writes multiple signal values.		✓	LIN,CAN	"Command VarWrite"
FormatSignals	Format signal values as strings.		✓	LIN,CAN	"Command FormatSignals"
LinMstReq	Issue a Master Request.	✓	✓	LIN	"Command LinMstReq"
LinSlvResp	Fetch a value from the Slave response.	✓	✓	LIN	"Command LinMstReq"
LinState	Read the LIN power supply state.	✓	✓	LIN	"Command LinState"
MacroExec	Execute a SDF macro	✓	✓	LIN,CAN	"Command MacroExec"
Diag22	Execute the UDS diagnostic service 22 "Read data by identifier".	✓	✓	LIN	"Command Diag22"
RTCWrite	Write the date and time of the RTC.	✓			"Command RTCWrite"

Command (and alias)	Description	MB	MB-II	Required channels	Link
RtcById	Execute the LIN node configuration service "Read by identifier".		✓	LIN	Command ById"
ReadByIdCompare	Execute the LIN node configuration service "Read by identifier similar".		✓	LIN	"Command ReadByIdCompare"
WaitSignal	Wait for a specified signal value.	✓	✓	LIN,CAN	"Command WaitSignal"
SeqRun	Execute a sequence from the SDF.	✓	✓	LIN,CAN	"Command SeqRun"
SeqExec	Execute a sequence from the SDF. (Deprecated: Use SeqRun)	✓	✓	LIN	"Command SeqExec"
Delay	Wait a specified time.		✓		"Command Delay"
DigOut	Set digital output		✓		"Command DigOut"
DigIn	Set digital input		✓		"Command DigIn"
SetConfigVar	Set a config variable.		✓		"Command SetConfigVar"
GetConfigVar	Get a config variable		✓		"Command GetConfigVar"
LicenceInstall	Install a license		✓		"Command LicenceInstall"

7.6 Commando List

7.6.1 Command Version

The `Version` command can be used to query the firmware version of the Baby-LIN-MB-II.



Tip

This command should always be the first command in a communication for the following reasons:

- This command can be used to test the connection. It will always respond, as long as the connection is established, no matter what hardware or software components you have installed or want to use.
- The firmware version is one of the most important questions, we will ask, if you encounter any problems and need support.

Return value	Example
The version of the firmware is returned.	:1.5.4
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Typ	Example	Description
Command	:Version	Query the firmware version.
Response	:1.5.4	The firmware version of the Baby-LIN-MB-II.



Version incompatibility

The composition of the version depends on the "API modes":

API mode		Version composition
0	Immediate API	1.5.4
1	CmdDone API	1.5.4
2	Compatibility API	V.1.5

7.6.2 Command SetApiMode



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.

The `SetApiMode` command can be used to switch between the different API modes. Check chapter "API modes" for more information.

Parameter	Description								
1	This parameter will set the API mode. This decides if command calls are blocking or not. Check chapter "API modes" for more information.								
	<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Immediate API</td> </tr> <tr> <td>1</td> <td>CmdDone API</td> </tr> <tr> <td>2</td> <td>Compatibility API</td> </tr> </tbody> </table>	Value	Description	0	Immediate API	1	CmdDone API	2	Compatibility API
Value	Description								
0	Immediate API								
1	CmdDone API								
2	Compatibility API								

Return value	Example
:0 is returned, if the mode was set.	:0
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	<code>:SetApiMode 0</code>	Set mode to the immediate API.
Response	:0	The API was set to the immediate API.

7.6.3 Command CMDDone



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.

The `CmdDone` command is required for the `CmdDone` API. If a command returns a token, it must be queried with this `CmdDone` command. The response will either contain the command's result or a busy result.

Check chapter "API modes" for more information.

Parameter	Description
1	This parameter is the token, a previous command returned. With this token the <code>CmdDone</code> command can check, if the previous command is finished. Check chapter "API modes" for more information.

Return value	Example
The busy response :B is returned, if the queried command has not yet finished. Try to send the <code>CmdDone</code> again some time later.	:B
The return value of the queried command is returned. This can also be an error.	:0
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	:SetApiMode 1	Set mode to the immediate API.
Response	:0	The API was set to the immediate API.
Command	:version	Request firmware version.
Response	:T4711	Received a token to request the command state.
Command	:CmdDone 4711	Check if command is finished.
Response	:B	The command is still executed.
Command	:CmdDone 4711	Check if command is finished.
Response	:1.2.3	The command is finished and the commands result is returned.

7.6.4 Command B



Warning

This command is only available for the Baby-LIN-MB. This command was intentionally not implemented for the Baby-LIN-MB-II. The baud rate of the Baby-LIN-MB-II can be changed using the "Web interface".

The B command changes the baud rate from the default value of 9600 Baud to the specified speed.



Advice

The response to this command is sent with the current baud rate. After the Baby-LIN-MB has sent the response, it will switch to the new baud rate.

Parameter	Description
1	This parameter is the designated baud rate. The minimum baud rate is 4800. The maximum baud rate is 115000.

Return value	Example
:0 if the new baud rate will be set..	:0
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	:B 38400	Set the baud rate to 38.400 Baud.
Response	:0	The baud rate will be set for the next command to 38.400 Baud. This response is sent with the old baud rate.

7.6.5 Command CurrentSdf



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.

The CurrentSdf command can be used to check if a SDFFile is loaded to a channel.

Parameter	Description
1	This parameter is the index of the channel, that should be checked, if a SDFile is loaded. This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.
2	The name of a SDFile, that exist in the internal memory of the Baby-LIN-MB-II.

Return value	Example
The name of the SDFile is returned, if a SDFile is loaded.	: 0
:@30 is returned, if no SDFile is loaded.	:@30
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	:CurrentSdf 0	Check if a SDFile is loaded to the channel with index 0.
Response	:@30	No SDFile was loaded to the channel with index 0.
Command	:SetApiMode 0	Load a SDFile to the channel with index 0.
Response	: 0	The SDFile was loaded successfully to the channel with index 0.
Command	:CurrentSdf 0	Check if a SDFile is loaded to the channel with index 0.
Response	:test.sdf	The name of the SDFile, that was loaded, is returned.




7.6.6 Command Start



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.

The Start command can be used to start the simulation for a specific channel. The channel index is passed in the first parameter. If the channel is a LIN channel, a schedule index can be passed using the optional second parameter.

Parameter	Description
1	<p>This parameter is the index of the channel, on which the simulation should be started.</p>  <p>Warning This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p>
2	<p>This parameter is the index of a schedule, that should be started with the simulation.</p>  <p>Advice If no parameter is passed, the first schedule is started automatically.</p>  <p>Warning This parameter can only be used on LIN channels. For CAN channels the second parameter will be ignored.</p>

Return value	Example
The name of the SDFFile is returned, if a SDFFile is loaded.	: 0
: 0 is returned, if the simulation was started successfully.	: 0
A default error may be returned. Check chapter "Return codes" for more information.	: @1

Type	Example	Description
Command	:Start 1 2	The simulation for the channel with index 1 is started and the schedule with index 2 is started.
Response	: 0	The simulation was started successfully.

7.6.7 Command LinStart

The LinStart command is an old alias for the "Command Start".

7.6.8 Command Stop



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.

The Stop command can be used to stop the simulation for a specific channel. The channel index is passed in the first parameter.

Parameter	Description
1	<p>This parameter is the index of the channel, on which the simulation should be started.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>

Return value	Example
The name of the SDFFile is returned, if a SDFFile is loaded.	: 0
: 0 is returned, if the simulation was started successfully.	: 0
A default error may be returned. Check chapter "Return codes" for more information.	: @1

Type	Example	Description
Command	:Stop 1	The simulation for the channel with index 1 is stopped.
Response	: 0	The simulation was stopped successfully.

7.6.9 Commando LinStop

The LinStop command is an old alias for the "Command Stop".

7.6.10 Commando LinSchedule



Warning

This command is only available on LIN channels. If it is executed on a CAN channel, an error will be returned.

The LinSchedule command can be used to start a schedule for a specific channel. The channel index is passed in the first parameter. The schedule index is passed in the second parameter.

Parameter	Description
1	<p>This parameter is the index of the channel, on which the schedule should be started.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>
2	This parameter is the index of a schedule, that should be started.

Return value	Example
The name of the SDFFile is returned, if a SDFFile is loaded.	: 0
: 0 is returned, if the simulation was started successfully.	: 0
A default error may be returned. Check chapter "Return codes" for more information.	: @1

Type	Example	Description
Command	:LinSchedule 1 2	The schedule with index 2 is started on the channel with index 1.
Response	: 0	The schedule was started successfully.

7.6.11 Commando SchedMode



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.



Warning

This command is only available on LIN channels. If it is executed on a CAN channel, an error will be returned.


The SchedMode command can be used to set the operation mode of a schedule.

After loading a SDFFile, all schedule tables are initiated with the cyclic schedule mode by default. A schedule table will keep it's mode until a new mode is explicitly set or the SDFFile is reloaded. So typically, the schedule mode for a particular table is given only once in the beginning.



Tip

The SchedMode command only sets the mode of the schedule. It will not start or queue the schedule. The schedule has to be started using "Command LinSchedule".

Parameter	Description
1	<p>This parameter is the index of the channel, on which the schedule should be started.</p>  <p>Warning This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p>
2	This parameter is the index of a schedule, that should be started.
3	This parameter is the operation mode for the specified schedule. Check the table below for the different modes and allowed values.

Mode value	Mode	Cyclic	Schedule switch	Description
0	Cyclic (Default)	✓	Immediately	In this mode the schedule table will be processed cyclically. It will be repeated until another schedule is requested. If a switch to another schedule table is requested, it will be performed immediately. This mode is the default mode and will be used when no SchedMode command has been issued for a schedule table.
1	Single run		After the schedule is finished	In this mode the schedule table will be processed once and completely. If no other schedule is requested, no schedule will be processed subsequently. If a switch to another schedule is requested, it will be queued and not executed before the current Single run schedule is finished. It is possible to queue up to 32 schedules.
2	Exit on complete	✓	After the schedule is finished	In this mode the schedule table will be processed cyclically. It will always finish and be repeated until another schedule is requested. If a switch to another schedule table is requested, it will be queued and not executed before the current Exit on complete schedule is finished. It is possible to queue up to 32 schedules.

Return value	Example
: 0 is returned, if the mode was set successfully.	: 0
A default error may be returned. Check chapter "Return codes" for more information.	: @1

Type	Example	Description
Command	:SchedMode 0 1 1	Sets the single run mode for the second schedule of the first channel 0. The schedule is not started.
Response	:0	The mode was successfully set.
Command	:SchedMode 0 2 1	Sets the single run mode for the third schedule of the first channel. The schedule is not started.
Response	:0	The mode was successfully set.
Command	:SchedMode 0 0 0	Sets the cyclic mode for the first schedule of the first channel 0. The schedule is not started.
Response	:0	The mode was successfully set.
Command	:Schedule 0 1	Start the second schedule in the single run mode. The schedule will immediately start, since no other schedule was running.
Response	:0	The mode was successfully started.
Command	:Schedule 0 2	Enqueue the third schedule in the single run mode. This schedule will not start, until the second schedule has finished.
Response	:0	The mode was successfully enqueued..
Command	:Schedule 0 0	Enqueue the third schedule in the cyclic mode. This schedule will not start, until the third schedule has finished. This schedule will run until another schedule is started.
Response	:0	The mode was successfully enqueued..



Attention

The queue for the schedule tables can contain a maximum of 32 entries. The user has to make sure, not to overfill the queue.






Tip

The system variable "@@SYSBUSSTATE" can be used to check, if a schedule is currently running.

7.6.12 Command RdSignal

The RdSignal command can be used to read the value of signals for a specific channel. The channel index is passed in the first and the signals in the following parameter.

Parameter	Description									
1	<p>This parameter is the index of the channel, on which the signal value should be read.</p>  <p>Warning This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p>									
2...17	<p>This parameter identifies the signal, whose value should be read.</p>  <p>Tip To identify a signal the following alternatives are possible:</p> <table border="1"> <thead> <tr> <th>Signal property</th> <th>Description</th> <th>Example</th> </tr> </thead> <tbody> <tr> <td>Signal index</td> <td>The signal index can be used simply by passing the index number.</td> <td>:LinRdSignal 0 12</td> </tr> <tr> <td>Signal name</td> <td>The signal name can be used by putting an exclamation mark directly in front of the name.</td> <td>:RdSignal 0 !SignalName</td> </tr> </tbody> </table>  <p>Warning The maximum number of signals, that can be passed, is 16.</p>	Signal property	Description	Example	Signal index	The signal index can be used simply by passing the index number.	:LinRdSignal 0 12	Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	:RdSignal 0 !SignalName
Signal property	Description	Example								
Signal index	The signal index can be used simply by passing the index number.	:LinRdSignal 0 12								
Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	:RdSignal 0 !SignalName								

This command will try to read the signal values directly from the bus. If they do not appear on the bus, an error is returned. The timeout depends on the number of signals, that should be read.

For one signal, the timeout is 300 ms. For each additional signal, 200 ms are added. So if four signals should be read, the maximum time that is waited is 900 ms. If not all signals appeared on the bus within this time, an error is returned.

Return value	Example
The signal values are returned as list of values, if all could be read. If any signal could not be read, an error will occur.	:2 -314 17
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	:RdSignal 1 2 !KeepAlive	The signals with index 2 and the name <code>KeepAlive</code> are read from the channel with index 1.
Response	:15 20	All signals could be read.

7.6.13 Command LinRdSignal

The `LinRdSignal` command is an old alias for the "Command `RdSignal`".

7.6.14 Command WrSignal

The `WrSignal` command can be used to write the value of a signal for a specific channel.



Attention

Only those signals can be written, that are published by a node, that is simulated by the Baby-LINMB- II. For CAN signals the frame has to be simulated as well.

Parameter	Description									
1	<p>This parameter is the index of the channel, on which the signal value should be written.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>									
2	<p>This parameter identifies the signal, whose value should be writtem.</p> <div style="background-color: #008080; color: white; padding: 5px;"> <p>Tip</p> <p>To identify a signal the following alternatives are possible:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>Signal property</th> <th>Description</th> <th>Example</th> </tr> </thead> <tbody> <tr> <td>Signal index</td> <td>The signal index can be used simply by passing the index number.</td> <td>:LinRdSignal 0 12</td> </tr> <tr> <td>Signal name</td> <td>The signal name can be used by putting an exclamation mark directly in front of the name.</td> <td>:RdSignal 0 !SignalName</td> </tr> </tbody> </table> </div>	Signal property	Description	Example	Signal index	The signal index can be used simply by passing the index number.	:LinRdSignal 0 12	Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	:RdSignal 0 !SignalName
Signal property	Description	Example								
Signal index	The signal index can be used simply by passing the index number.	:LinRdSignal 0 12								
Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	:RdSignal 0 !SignalName								
3	The value that should be written to the the signal.									

Return value	Example
:0 is returned, if the simulation was started successfully.	: 0
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	:WrSignal 1 !KeepAlive 12	The value 12 should be written to the signal with the name <code>KeepAlive</code> on the channel with index 1.
Response	: 0	The new value was written to the signal successfully..

7.6.15 Command LinWrSignal

The `LinWrSignal` command is an old alias for the "Command `WrSignal`".

7.6.16 Command VarRead



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.

The `VarRead` command is used to read multiple signal values at once. One signal will be used as starting signal. Starting with this signal, the values of each following signal will be returned. The format of the returned values can be configured.

Parameter	Description										
1	<p>This parameter is the index of the channel, whose signals should be written.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>										
2	<p>This parameter identifies the mode. The mode defines how the values will be returned.</p> <table border="1"> <thead> <tr> <th>Mode</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The values will be returned as single integers encoded as decimal values separated by space characters.</td> </tr> <tr> <td>1</td> <td>The values will be returned as single integers encoded as hexadecimal values separated by space characters.</td> </tr> <tr> <td>2</td> <td>The values will be returned as characters of a single string. Each signal will be evaluated as byte and encoded as ASCII character. Certain characters will be returned by escaping them. The ' character and the hexadecimal value will be used. A carriage return character would be represented as "\0D". If the length parameter is 0, values will be read until the first signal value is 0.</td> </tr> <tr> <td>3</td> <td>The values will be returned as characters of a single string. Each signal will be evaluated as byte and encoded as ASCII character. Certain characters will be returned as '.' characters. If the length parameter is 0, values will be read until the first signal value is 0.</td> </tr> </tbody> </table>	Mode	Description	0	The values will be returned as single integers encoded as decimal values separated by space characters.	1	The values will be returned as single integers encoded as hexadecimal values separated by space characters.	2	The values will be returned as characters of a single string. Each signal will be evaluated as byte and encoded as ASCII character. Certain characters will be returned by escaping them. The ' character and the hexadecimal value will be used. A carriage return character would be represented as "\0D". If the length parameter is 0, values will be read until the first signal value is 0.	3	The values will be returned as characters of a single string. Each signal will be evaluated as byte and encoded as ASCII character. Certain characters will be returned as '.' characters. If the length parameter is 0, values will be read until the first signal value is 0.
Mode	Description										
0	The values will be returned as single integers encoded as decimal values separated by space characters.										
1	The values will be returned as single integers encoded as hexadecimal values separated by space characters.										
2	The values will be returned as characters of a single string. Each signal will be evaluated as byte and encoded as ASCII character. Certain characters will be returned by escaping them. The ' character and the hexadecimal value will be used. A carriage return character would be represented as "\0D". If the length parameter is 0, values will be read until the first signal value is 0.										
3	The values will be returned as characters of a single string. Each signal will be evaluated as byte and encoded as ASCII character. Certain characters will be returned as '.' characters. If the length parameter is 0, values will be read until the first signal value is 0.										
3	<p>This parameter identifies the starting signal of the array.</p> <div style="background-color: #008080; color: white; padding: 5px;"> <p>Tip</p> <p>To identify a signal the following alternatives are possible:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #cccccc;">Signal property</th> <th style="background-color: #cccccc;">Description</th> <th style="background-color: #cccccc;">Example</th> </tr> </thead> <tbody> <tr> <td>Signal index</td> <td>The signal index can be used simply by passing the index number.</td> <td>:LinRdSignal 0 12</td> </tr> <tr> <td>Signal name</td> <td>The signal name can be used by putting an exclamation mark directly in front of the name.</td> <td>:RdSignal 0 !SignalName</td> </tr> </tbody> </table> </div>	Signal property	Description	Example	Signal index	The signal index can be used simply by passing the index number.	:LinRdSignal 0 12	Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	:RdSignal 0 !SignalName	
Signal property	Description	Example									
Signal index	The signal index can be used simply by passing the index number.	:LinRdSignal 0 12									
Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	:RdSignal 0 !SignalName									
4	This length parameter identifies the number of signals, that will be read.										

In modes 2 and 3 certain characters will be escaped or replaced.

These are all characters smaller than 32 (0x20, Space) and larger than 126 (0x7E, ~).

Return value	Example
A list of decimal encoded integers may be returned, if the mode is 0.	:1 2 3
A list of hexadecimal encoded integers may be returned, if the mode is 1.	:AB CD EF
A string containing escaped characters may be returned, if the mode is 2.	:new\0Dline
A string containing a lot of '.' characters may be returned, if the mode is 3.	:new.line
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	<code>:VarRead 0 0 !SIG1 4</code>	The values of 4 signals starting with SIG1 will be returned as list of decimal integers.
Response	<code>:1 2 3 4</code>	The 4 values are: 1, 2, 3, 4
Command	<code>:VarRead 0 1 !SIG1 5</code>	The values of 5 signals starting with SIG1 will be returned as list of decimal integers.
Response	<code>:67 89 AB CD EF</code>	The 5 values are: 103, 137, 171, 205, 239
Command	<code>:VarRead 0 2 !SIG1 0</code>	The values of the signals starting with SIG1 will be returned as string with escaped characters for escaped characters. The end of the string is defined by the first signal with value 0.
Response	<code>:new\0Dline</code>	8 characters are returned, because the ninth signal value was a 0. The fourth character is an escaped carriage return character.
Command	<code>:VarRead 0 3 !SIG1 8</code>	The values of 8 signals starting with SIG1 will be returned as string with '!' for escaped characters.
Response	<code>:new.line</code>	8 characters are returned. The fourth character was a carriage return character and therefore is returned as '!'.


7.6.17 Command VarWrite



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.

The `VarWrite` command is used to write multiple signal values at once. One signal will be used as starting signal. Starting with this signal, the values of each following signal will be set. The format of the passed values can be configured.

Parameter	Description															
1	<p>This parameter is the index of the channel, whose signals should be written.</p>  <div style="background-color: yellow; padding: 5px; margin-top: 10px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>															
2	<p>This parameter identifies the mode. The mode defines how the values can be passed.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Mode</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The value for each signal is passed as an individual value. If the value is larger than the signal, then the value is cut. The maximum number of values, that can be set at once, is 400.</td> </tr> <tr> <td>1</td> <td>A single string is passed. Each character is assigned to a single signal. The string uses the ASCII encoding. Non printable characters can be passed by escaping them. Just use the ' character and add the hexadecimal value of the character. A carriage return character would be represented as "\0D". If the value is larger than the signal, then the value is cut. The maximum number of values, that can be set at once, is 400.</td> </tr> <tr> <td>2</td> <td>A single value can be passed. That value is assigned to all signals. The number of target signals is defined by the length parameter. If the value is larger than the signal, then the value is cut.</td> </tr> </tbody> </table>	Mode	Description	0	The value for each signal is passed as an individual value. If the value is larger than the signal, then the value is cut. The maximum number of values, that can be set at once, is 400.	1	A single string is passed. Each character is assigned to a single signal. The string uses the ASCII encoding. Non printable characters can be passed by escaping them. Just use the ' character and add the hexadecimal value of the character. A carriage return character would be represented as "\0D". If the value is larger than the signal, then the value is cut. The maximum number of values, that can be set at once, is 400.	2	A single value can be passed. That value is assigned to all signals. The number of target signals is defined by the length parameter. If the value is larger than the signal, then the value is cut.							
Mode	Description															
0	The value for each signal is passed as an individual value. If the value is larger than the signal, then the value is cut. The maximum number of values, that can be set at once, is 400.															
1	A single string is passed. Each character is assigned to a single signal. The string uses the ASCII encoding. Non printable characters can be passed by escaping them. Just use the ' character and add the hexadecimal value of the character. A carriage return character would be represented as "\0D". If the value is larger than the signal, then the value is cut. The maximum number of values, that can be set at once, is 400.															
2	A single value can be passed. That value is assigned to all signals. The number of target signals is defined by the length parameter. If the value is larger than the signal, then the value is cut.															
3	<p>This parameter identifies the starting signal of the array.</p> <div style="background-color: #008080; color: white; padding: 5px; margin-top: 10px;"> <p>Tip</p> <p>To identify a signal the following alternatives are possible:</p> <table border="1" style="width: 100%; border-collapse: collapse; background-color: white; color: black;"> <thead> <tr> <th style="width: 20%;">Signal property</th> <th style="width: 50%;">Description</th> <th style="width: 30%;">Example</th> </tr> </thead> <tbody> <tr> <td>Signal index</td> <td>The signal index can be used simply by passing the index number.</td> <td>:LinRdSignal 0 12</td> </tr> <tr> <td>Signal name</td> <td>The signal name can be used by putting an exclamation mark directly in front of the name.</td> <td>:RdSignal 0 !SignalName</td> </tr> </tbody> </table> </div>	Signal property	Description	Example	Signal index	The signal index can be used simply by passing the index number.	:LinRdSignal 0 12	Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	:RdSignal 0 !SignalName						
Signal property	Description	Example														
Signal index	The signal index can be used simply by passing the index number.	:LinRdSignal 0 12														
Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	:RdSignal 0 !SignalName														
4...	<p>The following parameters depend on the mode.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Mode</th> <th style="width: 10%;">Parameter</th> <th style="width: 80%;">Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>4...n</td> <td>The values, that should be individually assigned to the target signals.</td> </tr> <tr> <td>1</td> <td>4</td> <td>The string, whose characters should be individually assigned to the target signals.</td> </tr> <tr> <td>2</td> <td>4</td> <td>The length of the array. The target value will be assigned to this number of signals.</td> </tr> <tr> <td>2</td> <td>5</td> <td>The target value, that will be assigned to each signal.</td> </tr> </tbody> </table>	Mode	Parameter	Description	0	4...n	The values, that should be individually assigned to the target signals.	1	4	The string, whose characters should be individually assigned to the target signals.	2	4	The length of the array. The target value will be assigned to this number of signals.	2	5	The target value, that will be assigned to each signal.
Mode	Parameter	Description														
0	4...n	The values, that should be individually assigned to the target signals.														
1	4	The string, whose characters should be individually assigned to the target signals.														
2	4	The length of the array. The target value will be assigned to this number of signals.														
2	5	The target value, that will be assigned to each signal.														

Return value	Example
: 0 is returned, if the values were written.	:@0
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	<code>:VarWrite 0 0 !SIG1 1 127 FFh</code>	The values 1, 127 and 255 will be assigned to the 3 signals starting with SIG1.
Response	<code>:0</code>	The new values were written successfully.
Command	<code>:VarWrite 0 1 !SIG1 "abcde"</code>	The values 97, 98, 99, 100 and 101 are assigned to the 5 signals starting with SIG1 .
Response	<code>:0</code>	The new values were written successfully.
Command	<code>:VarWrite 0 2 !SIG1 7 FFh</code>	The value 255 will be assigned to all 7 signals starting with SIG1.
Response	<code>:0</code>	The new values were written successfully.

7.6.18 Command FormatSignals




Warning

This command is only available on LIN channels. If it is executed on a CAN channel, an error will be returned.

The `FormatSignals` command is used to format one or multiple signals as text. One signal will be used as starting signal. Starting with this signal, the bytes of each signal will be evaluated as one continuous byte stream.

If the length of a signal is not a multiple of 8 bits, the missing bits will be filled with 0s. Therefore each signal will be interpreted as a 1-8 Byte wide value.

Parameter	Description									
1	<p>This parameter is the index of the channel, whose signals should be formatted.</p> <div style="background-color: yellow; padding: 5px;">  <p>Warning This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>									
2	<p>This parameter identifies the starting signal, from whose value a string should be created.</p> <div style="background-color: #008080; color: white; padding: 5px;"> <p>Tip To identify a signal the following alternatives are possible:</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="background-color: #cccccc;">Signal property</th> <th style="background-color: #cccccc;">Description</th> <th style="background-color: #cccccc;">Example</th> </tr> </thead> <tbody> <tr> <td>Signal index</td> <td>The signal index can be used simply by passing the index number.</td> <td><code>:LinRdSignal 0 12</code></td> </tr> <tr> <td>Signal name</td> <td>The signal name can be used by putting an exclamation mark directly in front of the name.</td> <td><code>:RdSignal 0 !SignalName</code></td> </tr> </tbody> </table> </div>	Signal property	Description	Example	Signal index	The signal index can be used simply by passing the index number.	<code>:LinRdSignal 0 12</code>	Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	<code>:RdSignal 0 !SignalName</code>
Signal property	Description	Example								
Signal index	The signal index can be used simply by passing the index number.	<code>:LinRdSignal 0 12</code>								
Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	<code>:RdSignal 0 !SignalName</code>								
3	The number of signals, that should be evaluated.									

Parameter	Description																		
4	This optional parameter selects the mode, what kind of string should be created.																		
	<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>This is the default case, if this optional parameter is not set. In this mode each signal byte is interpreted as ASCII coded character. The characters will be concatenated without additional space characters. Check mode 7 for an inverted byte order.</td> </tr> <tr> <td>1</td> <td>In this mode each signal byte is formatted as hexadecimal value. The bytes will be concatenated with an additional space character between each byte.</td> </tr> <tr> <td>2</td> <td>In this mode each signal byte is formatted as decimal value. The bytes will be concatenated with an additional space character between each byte.</td> </tr> <tr> <td>3</td> <td>In this mode each signal byte is formatted as BCD coded value. The nibble order is high nibble first (0x1234 => "1234").</td> </tr> <tr> <td>4</td> <td>In this mode each signal byte is formatted as BCD coded value. The nibble order is low nibble first (0x1234 => "1432").</td> </tr> <tr> <td>5</td> <td>In this mode all bytes of the signal are formatted as single decimal value. Signed signals will be formatted with sign.</td> </tr> <tr> <td>6</td> <td>In this mode all bytes of the signal are formatted as single hexadecimal value. Signed signals will be formatted without sign.</td> </tr> <tr> <td>7</td> <td>In this mode each signal byte is interpreted as ASCII coded character. The characters will be concatenated without additional space characters. This mode is like mode 0 with inverted byte order.</td> </tr> </tbody> </table>	Value	Description	0	This is the default case, if this optional parameter is not set. In this mode each signal byte is interpreted as ASCII coded character. The characters will be concatenated without additional space characters. Check mode 7 for an inverted byte order.	1	In this mode each signal byte is formatted as hexadecimal value. The bytes will be concatenated with an additional space character between each byte.	2	In this mode each signal byte is formatted as decimal value. The bytes will be concatenated with an additional space character between each byte.	3	In this mode each signal byte is formatted as BCD coded value. The nibble order is high nibble first (0x1234 => "1234").	4	In this mode each signal byte is formatted as BCD coded value. The nibble order is low nibble first (0x1234 => "1432").	5	In this mode all bytes of the signal are formatted as single decimal value. Signed signals will be formatted with sign.	6	In this mode all bytes of the signal are formatted as single hexadecimal value. Signed signals will be formatted without sign.	7	In this mode each signal byte is interpreted as ASCII coded character. The characters will be concatenated without additional space characters. This mode is like mode 0 with inverted byte order.
	Value	Description																	
	0	This is the default case, if this optional parameter is not set. In this mode each signal byte is interpreted as ASCII coded character. The characters will be concatenated without additional space characters. Check mode 7 for an inverted byte order.																	
	1	In this mode each signal byte is formatted as hexadecimal value. The bytes will be concatenated with an additional space character between each byte.																	
	2	In this mode each signal byte is formatted as decimal value. The bytes will be concatenated with an additional space character between each byte.																	
	3	In this mode each signal byte is formatted as BCD coded value. The nibble order is high nibble first (0x1234 => "1234").																	
	4	In this mode each signal byte is formatted as BCD coded value. The nibble order is low nibble first (0x1234 => "1432").																	
	5	In this mode all bytes of the signal are formatted as single decimal value. Signed signals will be formatted with sign.																	
6	In this mode all bytes of the signal are formatted as single hexadecimal value. Signed signals will be formatted without sign.																		
7	In this mode each signal byte is interpreted as ASCII coded character. The characters will be concatenated without additional space characters. This mode is like mode 0 with inverted byte order.																		
5	This optional parameter defines, if the processing will stop as soon as a byte is 0. This is useful for ASCII coded strings with different lengths. If the fourth parameter is 1 or 2 this behaviour might be wanted.																		
	<table border="1"> <thead> <tr> <th>value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The processing will stop as soon as a byte is 0.</td> </tr> <tr> <td>1</td> <td>This is the default case, if this optional parameter is not set. The processing will not stop as soon as a byte is 0.</td> </tr> </tbody> </table>	value	Description	0	The processing will stop as soon as a byte is 0.	1	This is the default case, if this optional parameter is not set. The processing will not stop as soon as a byte is 0.												
	value	Description																	
0	The processing will stop as soon as a byte is 0.																		
1	This is the default case, if this optional parameter is not set. The processing will not stop as soon as a byte is 0.																		

Return value	Example
The formatted string if no error occurred.	:Test
:@14 is returned, if a nibble is not a valid BCD encoded nibble.	:@14
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	:FormatSignals 0 !SIG1 2 0	The 2 signals starting with SIG1 are formatted as ASCII coded string.
Response	:HalloWelt	The ASCII coded string is returned.
Command	:FormatSignals 0 !SIG1 2 1	The 2 signals starting with SIG1 are formatted as hexadecimal coded bytes.
Response	:48 61 6c 6c 6f 57 65 6c 74	The hexadecimal coded bytes are returned.
Command	:FormatSignals 0 !SIG1 2 2	The 2 signals starting with SIG1 are formatted as decimal coded bytes.
Response	:72 97 108 108 111 87 101 108 116	The decimal coded bytes are returned.


7.6.19 Command LinMstReq



Warning

This command is only available on LIN channels. If it is executed on a CAN channel, an error will be returned.

The LinMstReq command can be used to issue a Master Request on the specified channel.

Parameter	Description
1	This parameter is the index of the channel, on which the Master Request should be issued.  <p>Warning This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p>
2...9	These parameters are the 8 data bytes of the Master Request.
10	This is the timeout in ms, that is waited to receive all Slave Responses.
11	This is the number of Slave Responses, that are expected after the Master Request. This parameter is optional. If it is omitted, one Slave Response will be expected. The maximum number of expected Slave Responses is 32.

Return value	Example
The formatted string if no error occurred.	:Test
:0 is returned, if the Master Request was issued. At this point the Master Request is only scheduled and no Slave Responses have been received.	:0
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description															
Command	:LinMstReq 0 43H 6H B2H 1H 2H 0H 0H 27H 1000	A Master Request is issued with the following properties:															
		<table border="1"> <thead> <tr> <th>Property</th> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>channel index</td> <td>0</td> <td>The index of the channel</td> </tr> <tr> <td>Data bytes</td> <td>43H 06H B2H 01H 02H 00H 00H 27H</td> <td>The data bytes</td> </tr> <tr> <td>Timeout</td> <td>1000</td> <td>The time within which the Slave Responses are expected.</td> </tr> <tr> <td>Expected Slave Responses</td> <td>1</td> <td>The number of expected Slave Responses. One is the default value, if none is passed.</td> </tr> </tbody> </table>	Property	Value	Description	channel index	0	The index of the channel	Data bytes	43H 06H B2H 01H 02H 00H 00H 27H	The data bytes	Timeout	1000	The time within which the Slave Responses are expected.	Expected Slave Responses	1	The number of expected Slave Responses. One is the default value, if none is passed.
		Property	Value	Description													
		channel index	0	The index of the channel													
		Data bytes	43H 06H B2H 01H 02H 00H 00H 27H	The data bytes													
Timeout	1000	The time within which the Slave Responses are expected.															
Expected Slave Responses	1	The number of expected Slave Responses. One is the default value, if none is passed.															
Response	: 0	The Master Request was issued.															

7.6.20 Command LinSlvResp



Warning

This command is only available on LIN channels. If it is executed on a CAN channel, an error will be returned.

The `LinSlvResp` command can be used to fetch a value from the Slave Responses from a previous `LinMstReq`. The parameters define the position within the data bytes of all received Slave Responses.

If a Master Request received more than one Slave Response, their data bytes are merged together to one big array. The position and length of the value, you are interested in, related to this array.

Parameter	Description								
1	<p>This parameter is the index of the channel, on which the Master Request was issued, this Slave Response was following.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>								
2	This is the starting bit of the value within the Slave Response's data bytes, you are interested in.								
3	This is the length in bits of the value within the Slave Response's data bytes, you are interested in.								
4	<p>This optional parameter lets you choose the format of the response.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>This is the default case, if this optional parameter is not set. The frame data is formatted as 64 bit integer. The maximum number of one frame can be returned.</td> </tr> <tr> <td>1</td> <td>The frame data are returned as hexadecimal coded bytes. The bytes are separated with space characters.</td> </tr> <tr> <td>2</td> <td>The frame data are returned as string. The encoding of the string depends on the sender.</td> </tr> </tbody> </table>	Value	Description	0	This is the default case, if this optional parameter is not set. The frame data is formatted as 64 bit integer. The maximum number of one frame can be returned.	1	The frame data are returned as hexadecimal coded bytes. The bytes are separated with space characters.	2	The frame data are returned as string. The encoding of the string depends on the sender.
Value	Description								
0	This is the default case, if this optional parameter is not set. The frame data is formatted as 64 bit integer. The maximum number of one frame can be returned.								
1	The frame data are returned as hexadecimal coded bytes. The bytes are separated with space characters.								
2	The frame data are returned as string. The encoding of the string depends on the sender.								



Version incompatibility

There are incompatibilities between the Baby-LIN-MB-I and the Baby-LIN-MB-II:

- Baby-LIN-MB-I:

The command always returns :B, if not all data are available and the timeout is not reached yet.

- Baby-LIN-MB-II, API mode 0:

For compatibility reasons the behavior is the same like with the Baby-LIN-MB-I:

The command always returns :B, if not all data are available and the timeout is not reached yet.

- Baby-LIN-MB-II, API mod 1:

Since a command returns :B if it is busy, the LinSlvResp command returns :I (incomplete).

Return value	Example
The value is returned, if the Master Request was executed, all Slave Responses were received and the value could be extracted from the data bytes.	:0
Only in API mode 0: :B is returned, if the Master Request is not finished or not all Slave Responses have been received	:B
Only in API mode 1 : linebreak :I is returned, if the Master Request is not finished or not all Slave Responses have been received.	:I
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	:SetApiMode 0	Show example in API mode 0.
Response	:0	The immediate API mode is activated.
Command	:LinMstReq 0 43H 6H B2H 1H 2H 0H 0H 27H 1000 2	The Master Request is scheduled and two Slave Responses are expected.
Response	:0	The immediate API mode is activated.
Command	:LinSlvResp 1 56 24	A value from the slave response is requested.
Response	:B	The Master Request was not yet sent and/or the Slave Responses have not been received completely.
Command	:LinSlvResp 1 56 24	A value from the slave response is requested again.
Response	:16777215	The Master Request was sent and the Slave Responses have been received completely. A 24 bit value was extracted from the data bytes.
Command	:SetApiMode 1	Show example in API mode 1.
Response	:0	The CmdDone API mode is activated.

Type	Example	Description
Command	:LinMstReq 0 43H 6H B2H 1H 2H 0H 0H 27H 1000 2	The Master Request is scheduled and two Slave Responses are expected.
Response	:0	The Master Request was issued.
Command	:LinSlvResp 1 56 24	A value from the slave response is requested.
Response	:I	The Master Request was not yet sent and/or the Slave Responses have not been received completely.
Command	:LinSlvResp 1 56 24	A value from the slave response is requested again.
Response	:16777215	The Master Request was sent and the Slave Responses have been received completely. A 24 bit value was extracted from the data bytes.

7.6.21 Command LinState



Warning

This command is only available on LIN channels. If it is executed on a CAN channel, an error will be returned.

The `LinState` command can be used to query the state of a LIN interface. The channel index is passed in the first parameter.



Advice

If the command is send to a CAN channel, the result will always be `:1`, which means bus power is supplied.

Parameter	Description
1	<p>This parameter is the index of the channel, on which the schedule should be started.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>

Return value	Example
:0 is returned, if no LIN power is supplied.	:@0
:1 is returned, if the LIN power is supplied.	:1
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	:LinState 1	Query the LIN state of the channel with index 1.
Response	:0	The channel is not supplied with bus power.
Command	:LinState 0	Query the LIN state of the channel with index 0.
Response	:1	The channel is supplied with bus power.

7.6.22 Command RdFrameError



Attention

The command is deprecated and is no longer supported as of firmware version V1.14.17. With the SDFV3 function to send a frame via blocking inject can be better checked whether a frame could be sent successfully.

The `RdFrameError` command can be used to query the error code that occurred the last time the specified frame was sent. The channel index is passed in the first and the frame ID in the second parameter.

Parameter	Description
1	<p>This parameter is the index of the channel, on which the error code should be queried.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>
2	This parameter identifies the frame, whose error code should be read.

Return value	Example
No error has occurred after the frame was sent.	:0
2: Read data from BUS does not match send data	:2
3: Framing error in data reception	:3
4: Checksum failed	:4
5: Data timed out (incomplete message reception)	:5
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	:RdFrameError 1 2	The error code, that occurred after the frame with ID 2 was last sent on the channel with index 1.
Response	:0	No error occurred.

7.6.23 Command MacroExec

The `MacroExec` command is used to start the execution of a macro on the specified channel.

Parameter	Description						
1	<p>This parameter is the index of the channel, that the macro should be executed on.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>						
2	The index of the macro, that should be executed.						
3	<p>This optional parameter configures, if the call just starts the macro and then instantly returns or if this call is blocking and will not return until the macro was finished.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="width: 10%;">Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The macro is just started and the command returns immediately.</td> </tr> <tr> <td>1</td> <td>The command does not return, until the macro finished its execution.</td> </tr> </tbody> </table> <div style="background-color: #008080; color: white; padding: 5px; margin-top: 10px;"> <p>Tip</p> <p>If you want to evaluate the result of a macro, the call must be blocking, until the macro has finished.</p> </div> <div style="background-color: yellow; padding: 5px; margin-top: 10px;"> <p>Warning</p> <p>If no value is passed for this parameter, the default value of 0 is used. Therefor the command will return immediately.</p> </div>	Value	Description	0	The macro is just started and the command returns immediately.	1	The command does not return, until the macro finished its execution.
Value	Description						
0	The macro is just started and the command returns immediately.						
1	The command does not return, until the macro finished its execution.						
4	<p>This optional parameter defines the maximum timeout in Milliseconds, that the command will be blocked. After this time the command will return with a <code>:@5</code>.</p> <div style="background-color: yellow; padding: 5px; margin-top: 10px;"> <p>Warning</p> <p>If no value is passed for this parameter, the default value of 1000 is used. Therefor the timeout is 1 second.</p> </div>						
5...14	These optional parameters will be passed as parameters to the called macro.						

Return value	Example
: 0 if the command was not blocking and the macro was started successfully or the command was blocking and the macro returned 0.	: 0
If the command was blocking, any value can be returned as result of the macro.	: 314
If the command was blocking, failures of the macro are returned with an offset of 50000. So any error between 50000 and 115535 are error results from the macro.	: @50698
If the command was blocking and the timeout was exceeded, a timeout error is returned.	: @5
A default error may be returned. Check chapter "Return codes" for more information.	: @1

Type	Example	Description
Command	:MacroExec 0 1	The macro with index 1 on the first channel will be started. The command will return immediately.
Response	:0	The macro was started successfully.
Command	:MacroExec 1 3 1	The macro with index 3 on the second channel will be executed. The command will be blocked until the macro is finished
Response	:7	The macro was executed and finished successfully with a return value of 7.
Command	:MacroExec 2 5 1 100	The macro with index 5 on the third channel will be executed. The command will be blocked until the macro is finished.
Response	:5	The macro was started, but did not finish within the given timeout time.
Command	:MacroExec 3 7 1 100 15 16383	The macro with index 7 on the fourth channel will be executed. The command will be blocked until the macro is finished. The first parameter of the macro will be set to 15. The second parameter of the macro will be set to 16383.
Response		

7.6.24 Command Diag22



Warning

This command is only available on LIN channels. If it is executed on a CAN channel, an error will be returned.

The `Diag22` command can be used to execute the UDS diagnostic service 0x22 Read Data by Identifier on a specific node. Although the UDS diagnostic service 0x22 features the query of multiple values with one call, this command allows the query of just one value at each call.




Warning

To execute this UDS diagnostic service a diagnostic schedule with MRQ (Frame ID: 0x3C) and SRP (Frame ID: 0x3D) frames must exist in the underlying LDF.



Warning
The diagnostic schedule will keep running even after the command has finished. If another schedule is required, it has to be started manually using the "Command LinSchedule".

Parameter	Description
1	This parameter is the index of the channel, on which the UDS diagnostic service should be executed.  Warning This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.
2	This parameter is the index of a schedule, that contains the MRQ and SRP frames.
3	This parameter is the NAD of the node.
4	This parameter is the 16 Bit service ID.

Return value	Example
: 0 if the command was not blocking and the macro was started successfully or the command was blocking and the macro returned 0.	: 0
The response will contain the ASCII data extracted from the multiple response frames coming from the slave.	: 3C8959537
A default error may be returned. Check chapter "Return codes" for more information.	: @1

Type	Example	Description
Command	:DIAG22 0 1 10 065EH	The request with the service ID 0x065E (in this example the software part number) is sent to the node with the NAD 10 on channel 0. Therefor the schedule with index 1 will be started.
Response	: 3C8959537	The answer as ASCII data (in this example the software part number).

7.6.25 Command RTCWrite



Warning
This command is only available for compatibility reasons for the Baby-LIN-MB-II. The function will always return :0 but not trigger any actions.

The `RTCWrite` command will not trigger any actions. It is only available due to compatibility reasons. The real-time clock is set using the "Web interface".

Parameter	Description
1	This parameter is the day of the month of the date. Valid values are: 1 . . . 31.
2	This parameter is the month of the date. Valid values are: 1 . . . 12.
3	This parameter is the year of the date. Valid values are: 2013 . . . 2200.
4	This parameter is the hour of the time. Valid values are: 0 . . . 23.
5	This parameter is the minute of the time. Valid values are: 0 . . . 59.
6	This parameter is the second of the time. Valid values are: 0 . . . 59.

Return value	Example
: 0 is always returned. Even though the real-time clock was not set.	: 0
A default error may be returned. Check chapter "Return codes" for more information.	: @1

Type	Example	Description
Command	:RTCWrite 24 12 2017 17 30 00	The real-time clock will not be changed. This command will do nothing and return a success.
Response	: 0	The real-time clock was not set. A success was only returned for compatibility reasons.

7.6.26 Command RTCRead

The RTCRead command can be used to read the date and time of the real-time clock.

Return value	Example
The date and time of the real-time clock.	:24 12 2017 17 30 00
A default error may be returned. Check chapter "Return codes" for more information.	: @1

Type	Example	Description
Command	:RTCRead	Read the date and time of the real-time clock.
Response	:24 12 2017 17 30 00	The successful command returns the date and time with the following list: <ul style="list-style-type: none"> • The day of the month of the date. • The month of the date. • The year of the date. • The hour of the time. • The minute of the time. • The second of the time.

7.6.27 Command ReadById



Warning

This command is only available on LIN channels. If it is executed on a CAN channel, an error will be returned.



Tip

This command is deprecated. The protocols in the SDFile are more powerful and more flexible. Check chapter "Custom Protocols" for more information. The output of the response data can be formatted with the "Command FormatSignals".

The `ReadById` command can be used to read node configuration and identification using the "Read by identifier" service from the LIN standard diagnostic.

The 8 data bytes of the Master Request always have the following structure:


Data bytes	D1	D2	D3	D4	D5	D6	D7	D8
Description	NAD	PCI	SID	Identifier	Supplier ID LSB	Supplier ID MSB	Funkction ID LSB	Functio ID MSB
Constant values		0x06	0xB2					
Values set via System Variables	@@Diag NodeId				@@Diag SupplierId		@@Diag FunctionId	
Values set via parameters				Second Parameter				


Additionally a schedule will be executed, that is defined by the system variable `@@ScheduleDiag`. This schedule should contain the MRQ and SRP diagnostic frames.

If the diagnostic schedule, NAD, supplier ID or function ID system variables are not set within the SDF, they will receive default values:

System variable	Description	Default value if not explicitly set
<code>@@ScheduleDiag</code>	Diagnostic schedule	0
<code>@@DiagNodeId</code>	NAD	0x7F
<code>@@DiagSupplierId</code>	Supplier ID	0x7FFF
<code>@@DiagFunctionId</code>	Function ID	0xFFFF

The default values of `@@DiagNodeId`, `@@DiagSupplierId` and `@@DiagFunctionId` are wildcards and address every node.

Parameter	Description																		
1	<p>This parameter is the index of the channel, that the SDFile should be loaded to.</p>  <p>Warning This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p>																		
2	<p>This parameter is the identifier used in the request. Certain identifiers are already defined in the LIN specification:</p> <table border="1"> <thead> <tr> <th>Identifier</th> <th>Description</th> <th>Length of response</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>LIN product identification</td> <td>RSID + 5</td> </tr> <tr> <td>1</td> <td>Serial number</td> <td>RSID + 4</td> </tr> <tr> <td>2...31</td> <td>Reserved</td> <td></td> </tr> <tr> <td>32...63</td> <td>user defined</td> <td>User defined</td> </tr> <tr> <td>64...255</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Identifier	Description	Length of response	0	LIN product identification	RSID + 5	1	Serial number	RSID + 4	2...31	Reserved		32...63	user defined	User defined	64...255	Reserved	
Identifier	Description	Length of response																	
0	LIN product identification	RSID + 5																	
1	Serial number	RSID + 4																	
2...31	Reserved																		
32...63	user defined	User defined																	
64...255	Reserved																		
3	<p>This parameter defines how the response data has to be interpreted. The following values are possible:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Read response as bytes as binary array.</td> </tr> </tbody> </table>	Value	Description	0	Read response as bytes as binary array.														
Value	Description																		
0	Read response as bytes as binary array.																		
4	<p>This parameter defines, how the response data should be formatted and returned by the Baby-LIN-MB-II. The following values are possible:</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The response will consist of the length of the data and the data bytes itself coded as hexadecimal values.</td> </tr> <tr> <td>1</td> <td>The response will consist of the length of the data and the data bytes itself coded as decimal values.</td> </tr> <tr> <td>2</td> <td>The response will consist of a single string. The data bytes are interpreted as ASCII characters.</td> </tr> <tr> <td>3</td> <td>The response will consist of a single decimal number. The data bytes will be interpreted as BCD coded bytes. The LSB is interpreted first.</td> </tr> </tbody> </table>	Value	Description	0	The response will consist of the length of the data and the data bytes itself coded as hexadecimal values.	1	The response will consist of the length of the data and the data bytes itself coded as decimal values.	2	The response will consist of a single string. The data bytes are interpreted as ASCII characters.	3	The response will consist of a single decimal number. The data bytes will be interpreted as BCD coded bytes. The LSB is interpreted first.								
Value	Description																		
0	The response will consist of the length of the data and the data bytes itself coded as hexadecimal values.																		
1	The response will consist of the length of the data and the data bytes itself coded as decimal values.																		
2	The response will consist of a single string. The data bytes are interpreted as ASCII characters.																		
3	The response will consist of a single decimal number. The data bytes will be interpreted as BCD coded bytes. The LSB is interpreted first.																		

Return value	Example
The interpreted data bytes if request was successfully.	:2 FF FF
A default error may be returned. Check chapter "Return codes" for more information.	:@1
If the slave responded with a NRC, the error also includes the first two bytes after the NRC.	:@31 FF FF
 <p>Version incompatibility This result type is only returned when the Compatibility API is active. If not, only the error code is returned. Check chapter "API modes" for more information.</p>	

Type	Example	Description
Command	:ReadById 0 1 0 0	The serial number is requested. The output format should be hexadecimal bytes.
Response	:5 F2 87 08 01 03	5 data bytes are returned. The request was successful, since the RSID is 0xF2. The 4 data bytes form the serial number.

7.6.28 Command ReadByIdCompare



Warning

This command is only available on LIN channels. If it is executed on a CAN channel, an error will be returned.



Tip

This command is deprecated. The protocols in the SDFFile are more powerful and more flexible. Check chapter "Custom Protocols" for more information.

The `ReadByIdCompare` command can be used to read node configuration and identification using the "Read by identifier" service from the LIN standard diagnostic and compare them with data stored in the SDFFile.

The 8 data bytes of the Master Request always have the following structure:

Data bytes	D1	D2	D3	D4	D5	D6	D7	D8
Description	NAD	PCI	SID	Identifier	Supplier ID LSB	Supplier ID MSB	Funktion ID LSB	Funciotn ID MSB
Constant values		0x06	0xB2					
Values set via System Variables	@@Diag NodeId				@@Diag SupplierId		@@Diag FunctionId	
Values set via parameters				Second Parameter				

Additionally a schedule will be executed, that is defined by the system variable `@@ScheduleDiag`. This schedule should contain the MRQ and SRP diagnostic frames.

If the diagnostic schedule, NAD, supplier ID or function ID system variables are not set within the SDF, they will receive default values:

System variable	Description	Default value if not explicitly set
<code>@@ScheduleDiag</code>	Diagnostic schedule	0
<code>@@DiagNodeId</code>	NAD	0x7F
<code>@@DiagSupplierId</code>	Supplier ID	0x7FFF
<code>@@DiagFunctionId</code>	Function ID	0xFFFF

The default values of `@@DiagNodeId`, `@@DiagSupplierId` and `@@DiagFunctionId` are wildcards and address every node.

The response will be compared with a compare block, that is defined in the SDFFile. A compare block always consist of the start index, the length and the data bytes, the response is compared with. The result will only show a match, if all data bytes from the compare block match the data from the response.

Up to 8 data blocks can be defined. The following examples show the structure of the compare blocks:

System variables	Description
<code>@@CmpBlockXStartPos</code>	This variable defines the index of the first byte within the data bytes in the response, that should be compared. X identifies the the compare block.
<code>@@CmpBlockXLen</code>	This variable defines the number of bytes, that should be compared. X identifies the compare block.
<code>@@CmpBlockXDataY</code>	This variable defines the value of a byte within the comapre block. X identifies the compare block. Y identifies the data byte index.

The following example shows the definition of two compare blocks:

System variable name	Initial value	Block identifier	Description
@@CmpBlock1StartPos	1	1	The comparison will start at the byte with index 1, which is the second byte.
@@CmpBlock1Len	4		The compare block consists of 4 data bytes.
@@CmpBlock1Data0	0x87		The response data will be compared with: 0x87 0x08 0x01 0x03
@@CmpBlock1Data1	0x08		
@@CmpBlock1Data2	0x01		
@@CmpBlock1Data3	0x03		
@@CmpBlock2StartPos	2	2	The comparison will start at the byte with index 2, which is the third byte.
@@CmpBlock2Len	6		The compare block consists of 6 data bytes.
@@CmpBlock2Data0	0x03		The response data will be compared with: 0x03 0x5D 0x3A 0x38 0x12 0x9F
@@CmpBlock2Data1	0x5D		
@@CmpBlock2Data2	0x3A		
@@CmpBlock2Data3	0x38		
@@CmpBlock2Data4	0x12		
@@CmpBlock2Data5	0x9F		

Parameter	Description																		
1	<p>This parameter is the index of the channel, that the SDFFile should be loaded to.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>																		
2	<p>This parameter is the identifier used in the request. Certain identifiers are already defined in the LIN specification:</p> <table border="1" style="width: 100%;"> <thead> <tr> <th>Identifier</th> <th>Description</th> <th>Length of response</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>LIN product identification</td> <td>RSID + 5</td> </tr> <tr> <td>1</td> <td>Serial number</td> <td>RSID + 4</td> </tr> <tr> <td>2...31</td> <td>Reserved</td> <td></td> </tr> <tr> <td>32...63</td> <td>user defined</td> <td>User defined</td> </tr> <tr> <td>64...255</td> <td>Reserved</td> <td></td> </tr> </tbody> </table>	Identifier	Description	Length of response	0	LIN product identification	RSID + 5	1	Serial number	RSID + 4	2...31	Reserved		32...63	user defined	User defined	64...255	Reserved	
Identifier	Description	Length of response																	
0	LIN product identification	RSID + 5																	
1	Serial number	RSID + 4																	
2...31	Reserved																		
32...63	user defined	User defined																	
64...255	Reserved																		
3	The identifier of the compare block in the SDFFile.																		

Return value	Example
: 0 if the response and the compare block are equal.	: 0
A default error may be returned. Check chapter "Return codes" for more information.	: @1
<p>If the slave responded with a NRC, the error also includes the first two bytes after the NRC.</p> <div style="background-color: orange; padding: 5px;"> <p>Version incompatibility</p> <p>This result type is only returned when the Compatibility API is active. If not, only the error code is returned. Check chapter "API modes" for more information.</p> </div>	: @31 FF FF

Type	Example	Description
Command	<code>:ReadByIdCompare 0 1 3</code>	The serial number is requested. The data bytes in the response are compared with block 3.
Response	<code>:0</code>	The data bytes in the response match the data in block 3.

Response 0 if the response data matches the compare values. The `ReadByIdCompare` command is available since firmware version V. 2.20.

7.6.29 Command WaitSignal

The `WaitSignal` command can be used to wait until a signal equals a certain target value or a timeout is reached.

Parameter	Description									
1	<p>This parameter is the index of the channel, on which the signal value should be queried.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>									
2	<p>This parameter identifies the signal, whose value should be compared.</p> <p>To identify a signal the following alternatives are possible:</p> <table border="1" style="width: 100%;"> <thead> <tr> <th>Signal property</th> <th>Description</th> <th>Example</th> </tr> </thead> <tbody> <tr> <td>Signal index</td> <td>The signal index can be used simply by passing the index number.</td> <td><code>:LinRdSignal 0 12</code></td> </tr> <tr> <td>Signal name</td> <td>The signal name can be used by putting an exclamation mark directly in front of the name.</td> <td><code>:RdSignal 0 !SignalName</code></td> </tr> </tbody> </table>	Signal property	Description	Example	Signal index	The signal index can be used simply by passing the index number.	<code>:LinRdSignal 0 12</code>	Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	<code>:RdSignal 0 !SignalName</code>
Signal property	Description	Example								
Signal index	The signal index can be used simply by passing the index number.	<code>:LinRdSignal 0 12</code>								
Signal name	The signal name can be used by putting an exclamation mark directly in front of the name.	<code>:RdSignal 0 !SignalName</code>								
3	This parameter must always be "=".									
4	This parameter is the target value, that the signal value needs to equal.									
5	This parameter is the timeout value in Milliseconds. If this time is passed and the signal has not equaled the target value, the command will return.									

Return value	Example
<code>:0</code> is returned, if the signal equals the target value.	<code>0:</code>
<code>:@16</code> is returned, if the signal did not equal the target value within the given timeout.	<code>:@16</code>
A default error may be returned. Check chapter "Return codes" for more information.	<code>:@1</code>

Type	Example	Description
Command	<code>:WaitSignal 0 !test = 5 3000</code>	This call will block until the signal value of test equals the target value 5 or 3 seconds have passed.
Response	<code>:@16</code>	The timeout was reached. The signal value never equaled the target value within this time period.
Command	<code>:WaitSignal 0 !test = 2 2000</code>	This call will block until the signal value of test equals the target value 2 or 2 seconds have passed.
Response	<code>:0</code>	The signal value equaled the target value.

7.6.30 Command SeqRun

The `SeqRun` command can be used to start a sequence. A sequence is a list of commands, that are stored in the SDFile. Check chapter "Sequences" for more information.

Parameter	Description								
1	<p>This parameter is the index of the channel, on which the sequence should run on. All channel indices within the sequence are replaced, except the special # notation is used.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>								
2	This parameter is the index of the sequence.								
3	<p>This optional parameter changes the value, that is returned.</p> <table border="1" style="width: 100%;"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>None</td> <td>If the all commands of the sequence returned expected values, the</td> </tr> <tr> <td>0</td> <td>return value is 0.</td> </tr> <tr> <td>1</td> <td>If the all commands of the sequence returned expected values, the return value is the response of the last command.</td> </tr> </tbody> </table>	Value	Description	None	If the all commands of the sequence returned expected values, the	0	return value is 0.	1	If the all commands of the sequence returned expected values, the return value is the response of the last command.
Value	Description								
None	If the all commands of the sequence returned expected values, the								
0	return value is 0.								
1	If the all commands of the sequence returned expected values, the return value is the response of the last command.								

Return value	Example	Description
Command	:SeqRun 0 1	The sequence with index 1 is started on channel 0.
Response	:0	All commands of the sequence were executed successfully.
Command	:SeqRun 1 4	The sequence with index 4 is started on channel 1.
Response	:103	The third command of the schedule returned not an expected value.
Command	:SeqRun 0 2 1	The sequence with index 2 is started on channel 0.
Response	:15	All commands of the sequence were executed successfully and the result of the last command was 15.
Command	:SeqRun 2 3 1	The sequence with index 3 is started on channel 2.
Response	:@101	The first command of the schedule returned not an expected value.

7.6.31 Command SeqExec



Version incompatibility

The SeqExec is deprecated and should not be used. Use "Command SeqRun" instead.

The SeqExec is missing the first parameter of "Command SeqRun", the channel index. The remaining parameters and the behaviour is the same.

7.6.32 Command Delay



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.

The Delay command can be used to create a delay of a given time.

Parameter	Description
1	The time in Milliseconds, that this command will take to execute.


Return value	Example
:0 returned, when the delay time elapsed.	0:
A default error may be returned. Check chapter "Return codes" for more information.	:@1

Type	Example	Description
Command	:Delay 1000	This command will take 1 second to finish.
Response	:0	The 1 second has passed.

7.6.33 Command DigOut

The DigOut command can be used to set the digital outputs of the Baby-LIN-MB-II.

Check chapter "X9 - LIN, CAN and IO" for more information.

Parameter	Description																		
1	This parameter selects the digital output.																		
	<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>Switch port (X9-24, X9-12)</td> </tr> <tr> <td>1</td> <td>Power switch (X9-22)</td> </tr> <tr> <td>2</td> <td>MIF-DIO-IO-4 (X9-19)</td> </tr> <tr> <td>3</td> <td>MIF-DIO-IO-5 (X9-6)</td> </tr> <tr> <td>4</td> <td>MIF-DIO-IO-6 (X9-18)</td> </tr> <tr> <td>5</td> <td>MIF-DIO-IO-1 (X9-5)</td> </tr> <tr> <td>6</td> <td>MIF-DIO-IO-2 (X9-4)</td> </tr> <tr> <td>7</td> <td>MIF-DIO-IO-3 (X9-7)</td> </tr> </tbody> </table>	Value	Description	0	Switch port (X9-24, X9-12)	1	Power switch (X9-22)	2	MIF-DIO-IO-4 (X9-19)	3	MIF-DIO-IO-5 (X9-6)	4	MIF-DIO-IO-6 (X9-18)	5	MIF-DIO-IO-1 (X9-5)	6	MIF-DIO-IO-2 (X9-4)	7	MIF-DIO-IO-3 (X9-7)
	Value	Description																	
	0	Switch port (X9-24, X9-12)																	
	1	Power switch (X9-22)																	
	2	MIF-DIO-IO-4 (X9-19)																	
	3	MIF-DIO-IO-5 (X9-6)																	
	4	MIF-DIO-IO-6 (X9-18)																	
	5	MIF-DIO-IO-1 (X9-5)																	
	6	MIF-DIO-IO-2 (X9-4)																	
7	MIF-DIO-IO-3 (X9-7)																		
 Warning These outputs are only available if a MIF-DIO is installed. Check chapter "MIF extension modules" for more information.																			
2	This parameter selects the target state for the digital output.																		
	<table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> <th>Switch port X9-24, X9-12</th> <th>Power switch X9-22</th> <th>MIF-DIO-IO-X</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>The output will be set to the low state.</td> <td>X9-24 and X9-12 are disconnected.</td> <td>X9-22 is floating.</td> <td>The pin ist floating</td> </tr> <tr> <td>1</td> <td>The output will be set to the high state.</td> <td>X9-24 and X9-12 are connected.</td> <td>X9-22 has the level of the LINSupply (X9-9).</td> <td>The pin has the level of MIF-DIOGND (X9-17)</td> </tr> </tbody> </table>	Value	Description	Switch port X9-24, X9-12	Power switch X9-22	MIF-DIO-IO-X	0	The output will be set to the low state.	X9-24 and X9-12 are disconnected.	X9-22 is floating.	The pin ist floating	1	The output will be set to the high state.	X9-24 and X9-12 are connected.	X9-22 has the level of the LINSupply (X9-9).	The pin has the level of MIF-DIOGND (X9-17)			
	Value	Description	Switch port X9-24, X9-12	Power switch X9-22	MIF-DIO-IO-X														
0	The output will be set to the low state.	X9-24 and X9-12 are disconnected.	X9-22 is floating.	The pin ist floating															
1	The output will be set to the high state.	X9-24 and X9-12 are connected.	X9-22 has the level of the LINSupply (X9-9).	The pin has the level of MIF-DIOGND (X9-17)															

Return value	Example
: 0 is returned, if the output was set successfully.	: 0
A default error may be returned. Check chapter "Return codes" for more information.	: @1

Type	Example	Description
Command	:DigOut 0 0	Set the switch port to disconnected
Response	: 0	The output was successfully set to the low state
Command	:Digout 1 1	Set the power switch to the level of the LIN-Supply
Response	: 0	The output was successfully set to the high state



Warning
Some plugins allow the inversion of the output states. If the digital outputs do not react like expected please check the configuration variables of the installed plugins.

7.6.34 Command DigIn



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.

The DigIn command can be used to evaluate the digital inputs of the Baby-LIN-MB-II.

Check chapter "X9 - LIN, CAN and IO" for more information.

Parameter	Description																
1	<p>This parameter selects the digital input.</p> <table border="1"> <thead> <tr> <th>Value</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>DigIn (X9-13, X9-25))</td> </tr> <tr> <td>1</td> <td>MIF-DIO-IN-1 (X9-5)</td> </tr> <tr> <td>2</td> <td>MIF-DIO-IN-2 (X9-4)</td> </tr> <tr> <td>3</td> <td>MIF-DIO-IN-3 (X9-7)</td> </tr> <tr> <td>4</td> <td>MIF-DIO-IO-4 (X9-19)</td> </tr> <tr> <td>5</td> <td>MIF-DIO-IO-5 (X9-6)</td> </tr> <tr> <td>6</td> <td>MIF-DIO-IO-6 (X9-18)</td> </tr> </tbody> </table>	Value	Description	0	DigIn (X9-13, X9-25))	1	MIF-DIO-IN-1 (X9-5)	2	MIF-DIO-IN-2 (X9-4)	3	MIF-DIO-IN-3 (X9-7)	4	MIF-DIO-IO-4 (X9-19)	5	MIF-DIO-IO-5 (X9-6)	6	MIF-DIO-IO-6 (X9-18)
Value	Description																
0	DigIn (X9-13, X9-25))																
1	MIF-DIO-IN-1 (X9-5)																
2	MIF-DIO-IN-2 (X9-4)																
3	MIF-DIO-IN-3 (X9-7)																
4	MIF-DIO-IO-4 (X9-19)																
5	MIF-DIO-IO-5 (X9-6)																
6	MIF-DIO-IO-6 (X9-18)																
	<p>Warning These inputs are only available if a MIF-DIO is installed. Check chapter "MIF extension modules" for more information.</p>																

Return value	Example	Description
0 : is returned, if the input is in low state.	: 0	
: 1 is returned, if the input is in high state.	: 1	
A default error may be returned. Check chapter "Return codes" for more information.	: @1	

Type	Example	Description
Command	DigIn 0	get the state of the DigIn
Response	: 0	The input was evaluated successfully and it is in low state
Command	DigIn 4	Get teh state of the MIF-DIO-IO-4.
Response	: 1	The input was evaluated successfully and ist is in high state

7.6.35 Command SetConfigVar



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.

The SetConfigVar command can be used to set the value of the configuration variables of the Baby-LIN-MB-II. Check chapter "Configuration variables" for more information.

Parameter	Description
1	This parameter selects the configuration variable by the name.
2	<p>This parameter selects the target value for the configuration variable. Depending on the type, different values are possible.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>This channel parameter is only used, if you use the single socket connection. If you instead are using multi socket connections, this channel parameter must be omitted. Check chapter "TCP connections: Single and multi socket" for more information.</p> </div>

Return value	Example
: 0 is returned, if the configuration variable was set successfully.	: 0
A default error may be returned. Check chapter "Return codes" for more information.	: @1

Type	Example	Description
Command	<code>:SetConfigVar autostart.mode 1</code>	Set the value of the configuration variable autostart.mode to value 1.
Response	: 0	The configuration variable was successfully set.
Command	<code>:SetConfigVar web.support.mail.subject "Support Mail"</code>	Set the value of the configuration variable web.support.mail.subject to value Support Mail.
Response	: 0	The configuration variable was successfully set.

7.6.36 Command LicencelInstall



Warning

This command is only available for the Baby-LIN-MB-II. The older Baby-LIN-MB does not support it.

The LicencelInstall command can be used to install an activation code to activate new features.



Warning

After a new activation code is installed, the Baby-LIN-MB-II must be restarted, before the activation code is active.

Parameter	Description
1	<p>This parameter is the activation code, you want to install. Check chapter "Handle voucher and activation codes" for more information.</p> <div style="background-color: yellow; padding: 5px;"> <p>Warning</p> <p>The activation code must be framed with quote signs.</p> </div>

Return value	Example
0: is returned, if the activation code was installed successfully.	: 0
A default error may be returned. Check chapter "Return codes" for more information.	: @1

Type	Example	Description
Command	:LicenceInstall "YKDFE TEA3F Q1FZ1 BJCFB IPM10 B"	Install the activation code YKDFE TEA3F Q1FZ1 BJCFB IPM10 B.
Response	0	The activation code was installed successfully.

7.7 Return codes

If a command was not understood or parameters were invalid or any specific error condition on the LIN Bus side causes a command to fail, the system will respond with a error code. All error codes start with the leading @ character. The error code with ID 1 would be returned as:

:@1

The following generic error codes exist:

Error code	Example	Error code	Example
1	Unknown command	60	Could not retrieve section infos
2	Invalid Parameter	61	Section could not be downloaded
3	Parameter out of range	70	Configuration variable not found
4	Parameter missing	71	Configuration variable could not be set
5	Command Timeout (incomplete command)	71	Configuration variable could not be received
6	File not found	80	Could not receive pin state
7	File load error	81	Invalid mode
8	SDF download to MIF-LIN failed	82	License code invalid
9	Internal MIF-LIN operation failed	83	Maximum reached
10	Error Signal look up failed	84	Transport protocol error: P2 timing elapsed
11	Node Timeout (no answer from LIN Bus)	99	Critical error. Please contact us: "Support information"
12	LIN bus supply missing	100 - 255	Error in sequence. Return value - 100 = the sequence step that returned an error
13	Channle unknown not activated	256	Power missing, when trying to establish EOL session
14	Data supplied by nodes is invalid or unknown	257	Eol request was responded by negative response
15	Command rejected due to missing prerequisite	260	Slave did not stop reporting Command in progress
16	Application function was not executed within given timeout	267	Slave did not respond on MasterRequest
17	Command failed due to settings in provided SDF	268	Power turned off, after being present in phase eol_initialisation
19	Error occurred during SDF parsing	280	Angle sent in Write OW Offset out of range range set by @@LimitForOWOffset != 0
20	A SDF download is pending	300 - 399	Parameter error. If the first parameter is invalid, 301 is returned. If the second parameter is invalid, 302 is returned.
30	No SDF loaded	400	USB mass storage device is missing
31	DTL negative response	401	USB mass storage device could not be mounted
32	DTL Processing	402	An escape sequence was incomplete
33	DTL invalid data	403	An escape sequence was illegal
34	DTL invalid Response lenght	404	An invalid character was entered: ' ' (0x20), '"' (0x22), ''' (0x7E)
35	Invalid NAD	405	Writing to th log file failed
40	Wrong password	406	Accessing the log path failed
41	Acess denied	420	The DT1 response date ID was not valid
42	Access level is not sufficient	430	A bus error occurred
50	Command lenght	431	A schedule index was invalid
		432	An internal library failure occurred

Error code	Example	Error code	Example
512 ... 768	A NRC (negative response code) other than 0x78 occurred. The error code is the NRC code with an offset of 512.	4000	An internal error occurred during the startup of the Baby-LIN-MB-II
790	PLugin inti failed	4001	A plugin could not be opened
791	Plugin configuration data invalid	4002	The initialization function of a plugin could not be found
792	Could not lookup command reference	4003	The initialization function of a plugin failed
793	Could not lookup frame	4004	The command handler could not be found
794	Error in diagnostic schedule	4005	The CmdDone handler could not be found
800	Invalid sequence index in SeqExec command	10000	Plugin init failed
801	Commend parser error in sequence	10001	Plugin init failed twice
802	Not expected value in sequence	10002	Plugin init RTV register failed
803	Plugin command not immediate	50000 ... 50255	Custom macro result error
810	Invalid macro index in MacroExec command	70000 ...	BabyLIN device error offset
811	Invalid macro (e.g. empty macro)	-100001	Internal resource allocation problem. Maybe out of memory/handles/etc..
812	Macro operation refused	-100002	Specified handle invalid
813	Macro still running	-100003	There is no connection open
815	Same macro still running	-100004	Serial port could not be opened or closed
816	Other Macro still running	-100005	Baby-LIN command syntax error
817	Macro start failed	-100006	Baby-LIN does not answer within in timeout
818	Macro has never been executed	-100007	Unable to open file
819	Macro exited with an unhandled exeception	-100008	Wrong parameter given to function
820	Invalid signal width	-100009	No datat available upon request
840	Unknown mode	-100010	No SDFFile was loaded previously
960	Error in BCD ccoded data from node	-100011	Internal message fformat error
961	Invalid compare block used (no definition in SDF file)	-100012	The given signal_nr or name does not exist in loaded SDFFile
962	Compare block mismatch	-100013	The signal chosen is a scalar, but an array function was called
998	Missing parameters for array type parameter	-100014	The signal chosen is an array, but an scalar function was called
999	Memory exhausted (Heap Error)	-100015	The SDFFile is unsupported by connected Baby-LIN due to insufficient firmware version
1000 ... 1999	Internal error	-100016	The given signal has no encoding
2000	Host response error	-100017	The given buffer is too small
2001	Double command	-100018	There is no additional answer data present from last sendCommand-call
2002	Parsing of sub command failed	-100019	Additional data with given index/name not present
3001	PMDM: Buspower is on	-100020	Device supports no Channels
3002	PMDM: Buspower is off	-100021	Unknown command passed to sendCommand
3003	PMDM: Flash init	-100022	A sendCommand message time out
3004	PMDM: Flash write	-100023	SDFFile can not be loaded to a the device due to incompatibility (incompatible SDFV3 to SDF-V2 device)
3005	PMDM: Flash read back	-100024	Value passed as a SDFFile handle is not valid
3006	PMDM:Flash channel locked	-100025	SDFFile can not be unloaded as the SDFFile is in use on a device
3007	PMDM: Flash channel unprepared	-100026	Can not execute command because SDFFile download is in progress

The following error codes are redirected from code, that is also used within the "Baby-LIN-DLL". Therefor the following error codes can occur. Please contact us, if you experience one of the following codes. Check chapter "Support information" for more information.

Error code	Example
0	Operation successful
-100027	Function can not be executed due to wrong mode or configuration
-100094	The number of parameters is not valid for this method
-100095	The value could not be read
-100096	One of the parameters in invalid
-100097	The property has no getter for the type
-100098	The property has no setter for the type
-100099	The value given was not set
-100100 ... -100200	The path parameter given to one of the BLC_UnifiedAccess functions could not be found. The index of that key is the return value - -100100. This index is 0 based.
-100201	The ISO-TP service is supposed to send a request but has no request data
-100202	During the reception of the response or the request a frame timeout occurred
-100203	A frame send by the library was not echoed by the BabyLIN device within a timeout. You might have to do a disframe/mon_on with that FrameID.
-100204	The response was not received within timeout_response milliseconds. Maybe the request is malformed?
-100205	A flow-control frame send by the library was not echoed by the BabyLIN device within a timeout. You might have to do a disframe/mon_on with that FrameID.
-100206	The flow-control state reported by the target is not one of the known states
-100207	The flow-control state was "wait"(0x1) in more then max_flow_wait flow-control frames
-100208	The flow-control state was "overflow"(0x2)
-100209	The flow-control was not issued by the other node
-100210	The data for a frame to send can not be put into a frame with the specified frame length.
-101100 ... -101200	The path parameter given to one of the BLC_UnifiedAccess functions could not be resolved. The index of the object, that could not be found, is the return value.

8 Support information

In case of any questions you can get technical support by email or phone. We can use TeamViewer to give you direct support and help on your own PC. This way we are able to sort out problems fast and direct. We have sample code and application notes available, which will help you to make your job.

Lipowsky Industrie-Elektronik GmbH realized many successful LIN and CAN related projects and therefor we can draw upon many years of experience in these fields. We also provide turn key solutions for specific applications like EOL (End of Line) testers or programming stations.

Lipowsky Industrie-Elektronik GmbH designs, produces and applies the Baby-LIN products, so you can always expect qualified and fast support.

Contact informations	Lipowsky Industrie-Elektronik GmbH, Römerstr. 57, 64291 Darmstadt		
Website:	www.lipowsky.com	Email:	info@lipowsky.de
Telephone:	+49 (0) 6151 / 93591 - 0		