



Baby-LIN

User manual V3.6

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 Imprint

Author	Lipowsky Industrie-Elektronik GmbH Römerstraße 57 64291 Darmstadt
Phone	+49 (0) 6151 / 93591 - 0
Fax	+49 (0) 6151 / 93591 - 28
E-Mail	info@lipowsky.de
Website	www.lipowsky.com
CEO	Andreas Lipowsky
Commercial register	Darmstadt HRB 5139
VAT-ID	DE 111647423
Quality Management	DIN EN ISO 9001:2008

Title	Baby-LIN - User manual
Version	V3.6
Date	12.02.2021
Valid for	Baby-LIN
Copyright	© 2021, Lipowsky Industrie-Elektronik GmbH, Darmstadt

This publication is copyright protected. All rights reserved, including those to translation, performance, use of illustrations and tables, broadcasting, microfilming or reproduction by other means, or electronic storage of all material contained herein.

All other brand names and trademarks used within this manual are unlimited subject to the applicable trademark laws and the ownership rights of their registered owners.

The hardware, firmware, software and documents of the Baby-LIN are subject to change without prior notice. Lipowsky Industrie-Elektronik GmbH thereby has no obligations.

2 Contents

1	Imprint.....	2
2	Contents.....	3
3	Glossary.....	8
4	Safety instructions.....	9
4.1	Basic information.....	9
4.2	Directives and standards.....	9
4.3	Intended use.....	9
4.4	Authorized Personnel.....	9
4.5	Hazards and protective devices.....	10
4.6	Warning signs.....	10
4.7	Safety precautions.....	11
5	Preface.....	12
5.1	Updates.....	12
5.1.1	Update philosophy.....	12
5.1.2	Downloads.....	12
5.1.3	Installation.....	13
5.1.4	Check versions.....	14
5.2	Support information.....	14
5.3	Declaration of conformity.....	16
5.4	Conflict Minerals Policy Statement.....	17
6	Datasheet.....	18
6.1	Product photo.....	18
6.2	Product description.....	18
6.3	Operation modes.....	18
6.4	Simulation modes.....	19
6.5	LIN-Bus properties.....	19
6.6	LINWorks suite.....	19
6.7	Technical Specifications.....	20
6.8	Use cases.....	21
6.9	Hardware requirements.....	21
6.10	Software requirements.....	21
6.11	LINWorks workflow.....	22
6.12	Scope of delivery.....	22
6.13	Ordering information.....	22
6.14	Distributors.....	24
7	Getting started.....	26
7.1	Introduction.....	26

7.2	Installation.....	26
7.3	Create a SDF.....	26
7.4	Connections.....	28
7.5	Start the bus communication.....	29
8	Hardware.....	31
8.1	Overview.....	31
8.2	Connectors.....	31
8.2.1	X1 - PC connector.....	31
8.2.2	X2 - LIN-Bus connector.....	31
8.3	Power supply.....	33
8.4	LEDs.....	33
8.4.1	LD1 - Run LED.....	33
8.4.2	LD2 - Error LED.....	34
9	Firmware.....	35
9.1	Firmware update.....	35
9.1.1	Introduction.....	35
9.1.2	Required software.....	35
9.1.3	Update the firmware.....	35
9.2	Stand-alone mode and autostart.....	36
9.2.1	Enable the stand-alone mode.....	36
9.2.2	Configure the autostart macro.....	36
9.2.3	Store a SDF persistently.....	37
9.2.4	Configure the device to automatically load a SDF and start a macro.....	37
9.3	Logging.....	39
9.3.1	Introduction.....	39
9.3.2	Log data targets.....	39
9.3.3	Log data formats.....	40
10	SDF - The session configuration file.....	41
10.1	Overview.....	41
10.2	Channels and sections.....	42
10.2.1	Introduction.....	42
10.2.2	Sections.....	42
10.2.3	Channels.....	43
10.2.4	Mapping of sections to channels.....	44
10.3	Section Properties.....	44
10.4	Device section.....	45
10.4.1	Target configuration and target-specific options.....	45
10.4.2	Sequences.....	47
10.5	Bus description.....	47

10.6	Emulation.....	48
10.6.1	Introduction.....	48
10.6.2	LIN emulation.....	49
10.7	Tables.....	49
10.8	Virtual Signals.....	50
10.8.1	Introduction.....	50
10.8.2	Sizes.....	50
10.8.3	Scope.....	51
10.8.4	Reset on bus start.....	51
10.9	System variables.....	51
10.9.1	Introduction.....	51
10.9.2	Quick reference.....	52
10.9.3	Timing.....	53
10.9.4	LIN header.....	56
10.10	Signalfunctions.....	57
10.10.1	Overview.....	57
10.10.2	Counter - Time based.....	58
10.10.3	Counter - Frame based.....	59
10.10.4	Checksum.....	60
10.10.5	XOR.....	61
10.10.6	CRC - 8 Bit.....	62
10.10.7	CRC - 16 Bit.....	63
10.11	Protocols.....	64
10.12	GUI-Elements.....	65
10.13	Macros.....	65
10.13.1	Overview.....	65
10.13.2	Command labels.....	65
10.13.3	Macro commands.....	66
10.14	Macroselections.....	70
10.15	Events.....	70
10.15.1	Introduction.....	70
10.15.2	Event conditions.....	71
10.15.3	Event action.....	73
11	LINWorks Software.....	76
11.1	LINWorks archive.....	76
11.2	Installation.....	77
11.3	Baby-LIN driver.....	77
11.4	LDFEdit.....	78
11.5	SessionConf.....	78
11.6	SimpleMenu.....	78
11.7	LogViewer.....	79

11.8	FrameBlaster.....	80
11.8.1	Introduction.....	80
11.8.2	Features.....	80
11.9	CustomPanel.....	80
11.9.1	Introduction.....	80
11.9.2	Features.....	80
11.10	Baby-LIN-DLL.....	81
11.11	Baby-LIN-DLL .NET Wrapper.....	82
11.12	Baby-LIN-DLL LabView Wrapper.....	82
11.13	Serial writer.....	82
11.14	BLProg.....	83
11.15	Switch to ASCII tool.....	83
12	Protocols and applications.....	84
12.1	Jumbo Frames: LIN frames with more than 8 data bytes.....	84
12.1.1	Introduction.....	84
12.1.2	Jumbo frames via inject commands.....	84
12.1.3	Testing jumbo frames using the SimpleMenu.....	85
12.1.4	Custom application using the Baby-LIN-DLL.....	86
12.2	Diagnostic modes.....	87
12.2.1	Supported diagnostic modes.....	87
12.2.2	Sleep Command (special Master Request Frame).....	104
12.2.3	Target configuration.....	104
12.3	ASCII mode.....	106
12.3.1	Introduction.....	106
12.3.2	Switch into the ASCII mode.....	106
12.3.3	Switch off the ASCII mode.....	107
12.3.4	ASCII Commands.....	107
12.3.5	Application example.....	110
13	FAQ - Frequently asked questions.....	111
13.1	General questions.....	111
13.1.1	Where can I download software, firmware, documentation, etc.?.....	111
13.1.2	How can I find out the device type and serial number?.....	112
13.2	SDFFile.....	112
13.2.1	Can a SDFFile be extracted from a device?.....	112
13.3	Voucher and activation codes.....	112
13.3.1	Are there voucher codes available? Can this device be enhanced using activation codes?.....	112
13.4	Stand-alone mode.....	112
13.4.1	How can I activate the stand-alone mode?.....	112
13.4.2	How can I configure the autostart macro?.....	113
13.4.3	How can I store a SDF persistently in a device?.....	113
13.4.4	How can I configure the Baby-LIN to automatically load a SDF and start a macro?..	114

13.5	Troubleshooting.....	115
13.5.1	How can I find out the device type and serial number?.....	115
13.5.2	How can I check, if the USB connection and/or firmware are working properly?.....	116
13.5.3	How can I check, if the LIN communication is working properly?.....	116
14	Troubleshooting.....	117
14.1	Introduction.....	117
14.2	Overview.....	117
14.3	Topics.....	118
14.3.1	Required software.....	118
14.3.2	Identification of the Baby-LIN device.....	118
14.3.3	USB connection and firmware check.....	120
14.3.4	Firmware update.....	121
14.3.5	SDF configuration.....	122
14.3.6	LIN-Bus communication.....	122
14.4	Shipping instructions.....	123
14.5	Error report.....	125
15	Embedded files.....	126
15.1	Access embedded files.....	126
15.2	LinCommunicationTest.sdf.....	126
15.2.1	LinCommunicationTest.....	126
15.3	DiagPure_19200.sdf.....	126
15.3.1	DiagPure_19200.....	126

3 Glossary

4 Safety instructions

4.1 Basic information

In this chapter the properties and use cases of the Baby-LIN are discussed. Thereof security measures are derived.

Read all parts of the technical documentation carefully, before operating the Baby-LIN.

Before connecting any device to the LIN-Bus read its technical documentation, to be aware of any security risks. Make sure that all required security measures are installed properly.

4.2 Directives and standards

The Baby-LIN was made in accordance with the current laws and regulations. The following laws and regulations have been considered:

EN 55024:1998 + A1:2001 + A2:2003

EN 55022:2006

4.3 Intended use

The Baby-LIN is intended to be used in controlled environments like laboratories, test departments and production facilities (EOL applications) to communicate with any LIN-Bus equipped devices.

The Baby-LIN is especially not intended to be used in uncontrolled environments like cars, trains, planes or any other transportation systems.

The Baby-LIN is intended to be used in compliance with the technical specifications in the datasheet. Check chapter "[Technical Specifications](#)" for more information.

4.4 Authorized Personnel

The Baby-LIN may only be used and installed by competent and trained personnel. The personnel needs to be trained regarding all devices connected directly or indirectly to any interface of the Baby-LIN as well as the whole installation.

The personnel especially needs knowledge about:

- Relevant safety rules
- Accident prevention regulations
- Relevant standards

Through this knowledge possible dangers can be recognized and prevented.

4.5 Hazards and protective devices

The Baby-LIN was manufactured according to the state-of-the-art and all safety relevant rules. Nevertheless, the usage may lead to hazards for the operating personnel or third party and may cause material damage.

Only use this unit if it is in technically perfect condition, utilized for the intended purpose and while keeping safety and potential dangers in mind. Any faults, that may affect the safety, have to be rectified instantly.

The workplace must be secured to meet all security precautions of each component connected directly or indirectly to the Baby-LIN.

4.6 Warning signs

The following warning signs are used for safety precautions:



DANGER indicates a hazardous situation which, if not avoided, will result in death or serious injury.



WARNING indicates a hazardous situation which, if not avoided, could result in death or serious injury.



CAUTION indicates a hazardous situation which, if not avoided, could result in minor or moderate injury.



NOTICE is used to address practices not related to physical injury.



Safety instructions signs indicate specific safety related instructions or procedures.

The following notice types are used to give you non safety precaution related information, e.g. software or configuration related problems:



Attention

This notice type signals possible problems, you should definitely pay attention to. Ignoring them probably lead to unexpected behaviour or data loss.



Version incompatibility

This notice type signals possible version incompatibilities and may lead to unexpected behaviour. These incompatibilities can be caused by old or incompatible software or firmware versions as well as missing activation codes.



Warning

This notice type signals possible problems, you should pay attention to. Ignoring them may lead to unexpected behaviour or data loss.



Advice

This notice type should inform you about useful information, that help you understand the Baby-LIN better.



Tip

This notice type should give you tips, that help to reduce your expense and time to implement.

4.7 Safety precautions

Despite compliance with the relevant laws and regulations, residual risks can not be excluded. The following safety precautions define the hazards that can occur when operating the Baby-LIN.

 DANGER	<p>Mortal danger by automatic start of connected devices.</p> <ul style="list-style-type: none"> ▶ Prepare for actions from connected devices. ▶ Study safety precautions of connected devices.
 WARNING	<p>Mortal danger by electric shock.</p> <ul style="list-style-type: none"> ▶ Operate the Baby-LIN only within dry conditions. ▶ Do not touch the Baby-LIN if powered. ▶ Do not touch the Baby-LIN if damaged. ▶ Do not touch the Baby-LIN if wet.

5 Preface

5.1 Updates

5.1.1 Update philosophy

The functionality and features of the Baby-LIN are defined by the installed firmware as well as the used versions of the LINWorks and Baby-LIN-DLL.

As we are permanently working on product improvements, the software and firmware are updated periodically. These updates make new features available and solve problems, which have been discovered by our internal tests or have been reported by customers with earlier versions.

All the firmware updates are done in a way, that the updated Baby-LIN will continue to work with an already installed, older LINWorks installation. So updating the Baby-LIN firmware does not mean, that you necessarily have to update your LINWorks installation as well.

Therefor it is highly recommended to always update your Baby-LIN to the latest available firmware version.

We also recommend to also update your LINWorks software and Baby-LIN-DLL, if new updates get available. Since new versions of the SessionConf may introduce new features to the SDF format, it is possible that older firmware, SimpleMenu, CustomPanel, FrameBlaster or Baby-LIN-DLL versions are not compatible. Therefor you should also update them.

If you update your LINWorks it is highly recommended updating the firmware of your Baby-LIN to the latest available firmware version as well as distributed the used versions of the Baby-LIN-DLL.

So the sole reason to stay with an older LINWorks version should be, that you use a Baby-LIN with outdated firmware version, which you can't upgrade for whatever reason.

It is highly recommended updating the Baby-LIN driver to the latest version.

5.1.2 Downloads

The latest versions of our software, firmware and documents are available in our client portal: portal.lipowsky.de. There you can find downloads like:

- Software, especially the LINWorks suite.
- Device firmware for the Baby-LIN. The BLProg update tool is included in the archive.
- USB drivers for the Baby-LIN.
- Manuals and datasheets.
- Application notes and examples how to create custom applications.



Advice

The LINWorks archive contains not only the LINWorks software but also the USB driver, manuals, datasheets, application notes and examples.

Only the device firmware packages are not included. The firmware is available as separate package.

Before you can download anything you have to register yourself in our customer portal. Here you also have to select which product you are using. After your account has been activated by us you will receive an e-mail. This can take up to some hours.

After you received the activation e-mail you will be able to login. In the login screen you should also select your preferred language. After the login you will be able to download all relevant files.

Login
Registration
Contact
Imprint

Select language /
Sprache wählen

English

change/ändern

Registration

Registration

You are interested in Software Downloads and further information?

Please fill out our registration form to become a user of our portal.

Companyinfo

Company *

Street *

ZIP *

City *

Phone

German

Default-language

Fax

Email *

Username (3-20 characters) *

Password (min. 6 characters) *

Passwordcheck

Contact

Mr.

Salutation *

Name *

Surname *

Register

Groups

Baby-Lin User

Distributors

HARP user

Posco Users

Groups *

(Notice: You are customer of one of our products or sales prospect? Please choose the relevant groups to get access to downloads and further information!)

Login
Registration
Contact
Imprint

Select language /
Sprache wählen

English

change/ändern

Login

Welcome to the Customerportal of Lipowsky

Username *

Password *

Login

not yet registered? [register here...](#)

After the registration you will also be automatically notified via e-mail when new versions are available.



Advice

If you have forgotten your password, please send us your E-Mail address, that you used for the registration. Then we can reset your password so you can login again.

5.1.3 Installation

The LINWorks suite is delivered with a handy setup application. If you already have installed an older version you can simply install the newer versions. The setup application will take care of overwriting the required files. Simply follow these steps:

- Start the "Setup.exe".
- Select the components you want to install.
- Follow the instructions.



Warning

Please stop all running LINWorks applications and disconnect all Baby-LIN devices before starting the setup.



Version incompatibility

If you have used the SessionConf and SimpleMenu with version V1.x.x, the new version will be installed parallel to the old ones. Therefore you have to use the new shortcuts to start the new versions.

5.1.4 Check versions

If you want to check the current version of the Baby-LIN firmware or a LINWorks component the following table shows you how it is done:

Component	How to check the version
Baby-LIN firmware	Start the SimpleMenu and connect to the Baby-LIN. Now the firmware version is visible in the device list.
LINWorks: <ul style="list-style-type: none"> • LDFEdit • SessionConf • SimpleMenu • LogViewer • FrameBlaster • CustomPanel • Baby-LIN-MB-Tool 	Select the menu option "Help"/"About"/"Info". The info dialog will show the software version.
Baby-LIN-DLL	Call BLC_getVersionString(). The version is returned as string.
Baby-LIN-DLL .NET Wrapper	Call GetWrapperVersion(). The version is returned as string.



Advice

If you need support please always tell us the firmware and software versions you are using.

5.2 Support information

In case of any questions you can get technical support by e-mail 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 therefore 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 information

Website:	www.lipowsky.com	E-Mail:	info@lipowsky.de
Customer portal:	portal.lipowsky.de	Telephone:	+49 (0) 6151 / 93591 - 0
Shipping address:	Lipowsky Industrie-Elektronik GmbH, Römerstraße 57, 64291 Darmstadt, Germany		

5.3 Declaration of conformity

Lipowsky Industrie-Elektronik GmbH
Römerstraße 57, 64291 Darmstadt
Quality management



Declaration of conformity

We Lipowsky Industrie-Elektronik GmbH
 Römerstraße 57
 64291 Darmstadt

declare under our sole responsibility, that the product

Baby-LIN

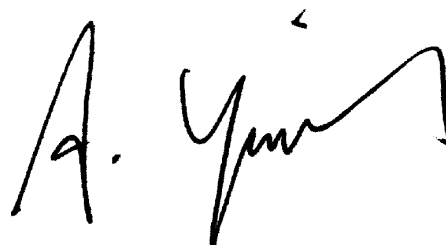
to which this declaration relates, are in conformity with the following standards

EN 55024:1998 + A1:2001 + A2:2003

EN 55022:2006

following the provisions of directives **2014/30/EU**.

We declare, that the products conform the requirements of the RoHS directive.



Darmstadt, 12.02.2021

Dipl.-Ing. Andreas Lipowsky
(CEO Lipowsky Industrie-Elektronik GmbH)

5.4 Conflict Minerals Policy Statement

Lipowsky Industrie-Elektronik GmbH
Römerstraße 57, 64291 Darmstadt
Quality management



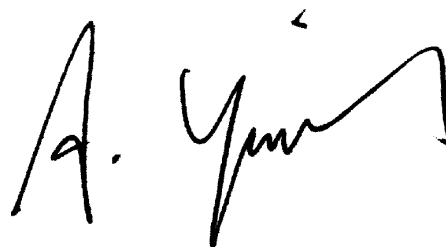
Conflict Minerals Policy Statement

The Lipowsky Industrie-Elektronik GmbH supports the ending of violence and human rights violations in the mining of minerals from the area known as the "Conflict Region" in the east of the Democratic Republic of Congo (DRC) and surrounding countries.

We have undertaken due diligence with our suppliers including the provision of written statements that our purchases of tin, tantalum, tungsten and gold originate from outside of the "Conflict Region".

The Lipowsky Industrie-Elektronik GmbH will not purchase products that contain conflict minerals that directly or indirectly finance or benefit armed groups in the DRC or adjoining countries. The Lipowsky Industrie-Elektronik GmbH expects its suppliers to undertake reasonable due diligence with their supply chains to assure that specified metals are being sourced only from:

- Mines and smelters outside the "Conflict Region" or
- Mines and smelters which have been certified by an independent third party as "conflict free" if sourced within the "Conflict Region".



Darmstadt, 12.02.2021

Dipl.-Ing. Andreas Lipowsky
(CEO Lipowsky Industrie-Elektronik GmbH)

6 Datasheet

6.1 Product photo



6.2 Product description

The Baby-LIN allows controlling LIN-Bus equipped devices by using a standard PC. After installation of the supplied **LINWorks** software, you can connect the Baby-LIN to a free USB port, and access the LIN-Bus devices via LINWorks or your own applications by using the Baby-LIN-DLL.

The power for the Baby-LIN can be supplied either from the USB or the LIN-Bus side. If the device is powered by the LIN-Bus it can be operated without the need for a PC.

The Baby-LIN can handle LIN-Bus voltages in the range of 8-36 VDC.

All communication interfaces (LIN-Bus, USB) are **galvanically isolated**, eliminating interferences between the PC and the board electronics.

The Baby-LIN unit includes its own 32-bit microcontroller, which takes care of all **time critical** tasks of the LIN-Bus protocol.

The device firmware is field updateable, so the changes of bus specification or upcoming new system features can be adapted easy.

6.3 Operation modes

Any situation that requires communication with a LIN device is a potential field of application for a Baby-LIN. It is a versatile tool that can be used in research laboratories, test departments and production (EOL applications).

The Baby-LIN allows for different operation modes to support typical use cases like:

- **Monitor** and log all frames on the bus without the need for a SDF. If a SDF is available signal values can also be monitored.
- **Control** the bus via the **LINWorks** software or customer specific applications by using the **Baby-LIN-DLL**.
- **Program** and store free programmable command sequences in the Baby-LIN to run it as a **stand-alone** device without the need for a PC. Thus you can run a bus driven ECU in a **durability test** or **EOL applications** without any PC connected.

6.4 Simulation modes

The Baby-LIN is able to simulate different configurations of LIN-Bus nodes. It is possible to **simulate any number of nodes** ranging from none to all. These are some typical configurations:

- Simulate the **LIN-Bus master** to operate slave nodes.
- Simulate any number of **LIN-Bus slave** nodes.
- Simulate **all nodes** and therefor the complete communication on the bus.
- Simulate all but one node and realize a **residual bus simulation**.
- Simulate no node to **monitor** the bus communication only.

6.5 LIN-Bus properties

The used LIN driver supports bus voltages of 8-36 VDC and can be used to up to 200 kBaud. That way even nodes that operate outside the standard limits of the LIN specifications can be controlled with the Baby-LIN. Supported LIN-versions are V.1.2, V.1.3,...V.2.2. The pull-up resistor of the LIN-Bus driver is switched to 30 kΩ, if the master node is emulated and to 1 kΩ, if only slave nodes are emulated.

The maximum supported signal cable length of the LIN-Bus is 30m.

6.6 LINWorks suite

The purchase of a Baby-LIN includes the license to download the **LINWorks** suite. This suite is a collection of PC software that supports you during the whole workflow.

The **LDFedit** allows the inspection, creation and edit of a LDFile (LIN Description File).

The **SessionConf** allows the inspection, creation and edit of a SDFile (Session Description File) and features a file import for LDFiles (for LIN-Bus simulation). It defines everything needed for a complete simulation of each available bus, e.g. which nodes on each bus are available and which nodes should be simulated by the Baby-LIN. Moreover it allows defining an application logic. This programming ability is available for each device out of the box.

The **SimpleMenu** is used to establish a connection to the Baby-LIN and upload SDFiles, change the device target configuration, control the bus and monitor the frames and signals on the bus. Even without a LDFile/SD-File the bus can be monitored and the frames can be logged.

The **Baby-LIN-DLL** allows customers to create their own application and use all features of the Baby-LIN like controlling and monitoring the LIN-Bus interfaces. The **Baby-LIN-DLL** is a native **C/C++** DLL. It is available for **Windows**, **Linux** and **RaspberryPi**. Wrapper for **.NET**, **Python**, **VB6** and **LabView** are available. Of course we provide examples for all supported languages.

The **LogViewer** can show and convert the log files of the SimpleMenu.

The **FrameBlaster** is a script interpreter, that gives you access to the features of the Baby-LIN from within a JavaScript. The Script can be developed and executed in an integrated development environment, but also executed by a command line tool to allow batch execution.

The **CustomPanel** is a graphical user interface, whose layout is stored in a configuration file. The controls allow you to show and control LIN based signals from a SDFFile. With this tool any customer can quickly create complex user interfaces based on your requirements. A graphical editor to create the configuration of the graphical user interface is included.

The **LINWorks** software runs on 32 and 64 bit Windows versions.

6.7 Technical Specifications

Device

- CPU: ARM-7, 60 MHz
- Memory: 32 kB RAM
- 2 LEDs: Signal bus and error states
- Power supply: via USB or LIN-Bus power
- Maximum current consumption: 70 mA @ 12 VDC
- Galvanic isolation of all communication interfaces (LIN-Bus, USB)

Interface: LIN

- 1 LIN-Bus interface available
- LIN-Bus connection via 3 pin connector (MC 1,5/ 3-ST-3,81)
- LIN-Bus supply voltage: 8-36 VDC
- LIN-Bus baud rate: up to 200 kBaud (Support of protocols outside of the LIN specification)
- Supported LIN versions: V1.2, V1.3,...V2.2
- Supported LIN related protocols: Cooling and SAE J2602
- Maximum signal cable length for LIN-Bus: 30 m

Interface: USB Device

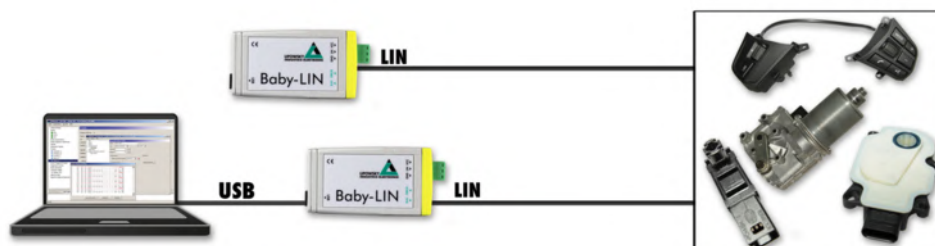
- USB 2.0 interface
- Connection via USB type B-Mini

Case

- Degree of protection: IP20
- Operating temperature: -20° - +60° Celsius
- Weight: 37 g
- Case dimensions [mm]: 78 x 45 x 18 (L x W x H)

Elements like connectors are not included.

6.8 Use cases



6.9 Hardware requirements

The following hardware is required to operate the Baby-LIN:

Requirement	Purpose
A PC with about 200 MB free hard drive space	Required for the installation of the LINWorks software. Please check the software requirements and use cases.
A free USB port	Required to transfer SDFiles to the Baby-LIN.
	Required for firmware updates.

6.10 Software requirements

The LINWorks software requires one of the following operating systems:

- Windows XP
- Windows Vista (32 and 64 Bit)
- Windows 7 (32 and 64 Bit)
- Windows 8 (32 and 64 Bit)
- Windows 10 (32 and 64 Bit)



Version incompatibility

The Baby-LIN-DLL is available for Linux. The exact requirements are available upon request.

Some additional tools available in the LINWorks software suite require an installed .NET Framework v4.0.

To install LINWorks components administration privileges are required.

6.11 LINWorks workflow

Baby-LIN Workflow

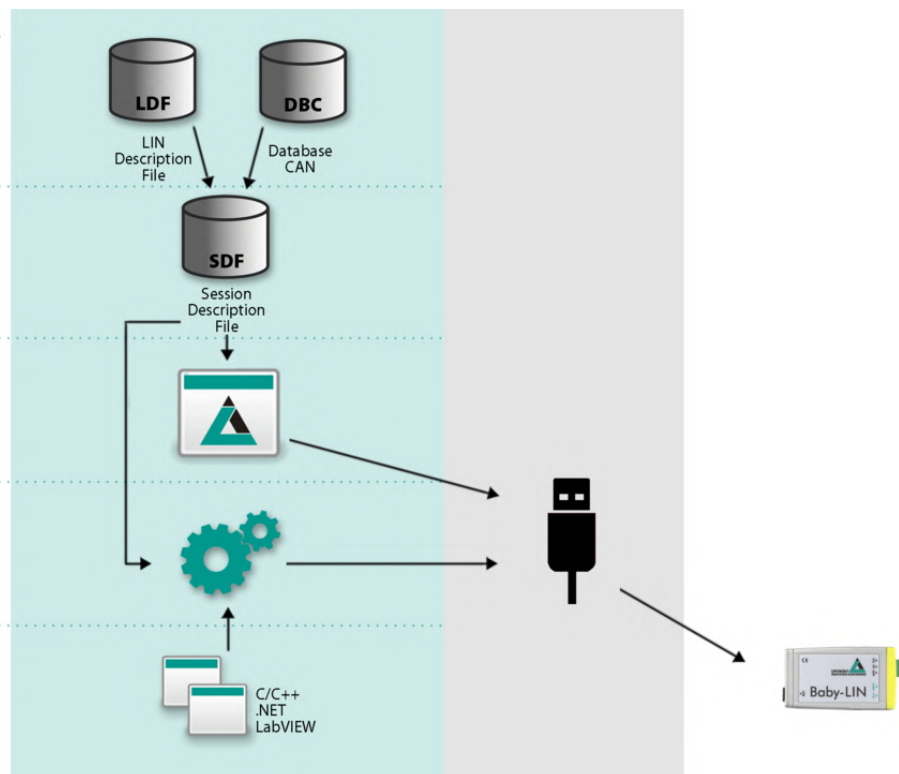
LDF-Editor

Session Configurator

Simple Menu

Baby-LIN-DLL

Custom Application



6.12 Scope of delivery

The delivery of a Baby-LIN systems includes the following components:

- Baby-LIN device
- USB 2.0 cable, 1.5m, Type A to type B-Mini
- Plug components for all terminals:
 - 1 3-pin plug with screw connection (MC 1,5/ 3-ST-3,81)
- Download license for the LINWorks Suite (includes LINWorks PC software, USB-driver, example files and documentations)

6.13 Ordering information



Attention

This device is replaced by a successor and can not be ordered anymore. Please refer to the datasheet of the Baby-LIN-II for updated ordering information.

Main device

Item number	Item	Description
8000565	Baby-LIN	LIN-Bus simulation device with USB interface



Advice

Each device includes a download license for the LINWorks application suite. This PC software can be downloaded using our client portal: portal.lipowsky.de



Tip

Country of origin: Germany
Customs tariff number: 90308930

Optional hardware components

Item number	Item	Description
3500701	USB 2.0 cable, 1.5m Type A to type B-Mini	This cable connects the Baby-LIN to a PC. Such a cable is already within the scope of delivery.
3021303	MC 1,5/ 3-ST-3,81	3-pin plug component, screw connection with tension sleeve. Cable outlet parallel to plugin direction. Screw direction vertical to plugin direction.



Advice

All devices are delivered with a full set of plug components. An extra order is necessary for replacement or configuration purposes only.

Optional software components

Item number	Item	Description
9004210	Customer specific installation.	Installation of customer specific SDFile version and/or installation of license activation key.
9103010	LINWorks CD	The LINWorks archive with PC software for all Baby-LIN products on a physical medium (CD).



Tip

You can order the Baby-LIN as test device. Try it for one week and convince yourself. Please contact us over info@lipowsky.de.

6.14 Distributors

Distributors in Asia

	China		Hongke Technology Co. LTD Room 701, No. 2, the 3rd Street of Kehui Jingu, No. 99 of Science Avenue, Huangpu District, Guangzhou, China	Phone: +86 20 3874 4538 E-Mail: sales@hkaco.com Web: www.hkaco.com
	Taiwan		Microport Computer Electronics Inc. Building A, No.143, Ln. 152, Sec. 1, Baoda Rd., Guiren, Dist. Tainan City, Taiwan	Phone: +886 6 330 3000 Fax: +886 6 330 2526 E-Mail: inquiry.microport@gmail.com Web: www.microport.com.tw
	South Korea		KMDATA Inc. 505-2, Daeryung Techno Town 1-cha 18, Gasan Digital 2-ro, Geumcheon-gu, Seoul, 08592, Korea	Phone: +82 2 3281 0333 Fax: +82 2 3281 3117 E-Mail: daniel@kmd.co.kr Web: www.kmd.co.kr
	Japan			

Distributors in North America

	USA		FEV North America Inc. 4554 Glenmeade Lane, Auburn Hills, MI 48326-1766, USA	Phone: +1 248 724 2830 Fax: +1 248 373 8483 E-Mail: marketing_fev@fev.com Web: www.fev.com
	Canada			
	Mexico		Círculo SEI S.A. de C.V. Guanajuato Tecno Parque VIDA UG, Planta Baja, Oficina 15 Círculo SEI, Autopista Federal Guanajuato, Silao Km 3.8., Colonia Capulín de la Cuesta, CP. 36292, Silao City, Guanajuato State, México	Phone: +52 473 1030459 E-Mail: sales@circulo-sei.com Web: www.circulo-sei.com

Distributors in Europe

	France		ISIT ZA La Menude 7 Rue André Marie Ampère 31830 Plaisance Du Touch France	Phone: +33 561 306 900 Fax: +33 561 165 063 E-Mail: contact@isit.fr Web: www.isit.fr
	UK		The Debug Store Berwyn House, Carrog, CORWEN, Denbigshire LL21 9AT UK	Phone: +44 1490 430526 Fax: +44 1490 430241 E-Mail: sales@TheDebugStore.com Web: www.thedebugstore.com
	Ireland			
	UK		LMH Engineering Services Ltd 1 James Avenue, Paignton, South Devon, TQ3 3LZ UK	Phone: +44 17542 725 765 E-Mail: info@lmh-engineering-services.co.uk Web: www.lmh-engineering-services.co.uk
	Ireland			

For all other regions or if you are unsure, contact us:

	<p>World- wide</p>		<p>Lipowsky Industrie-Elektronik GmbH Römerstraße 57 64291 Darmstadt Germany</p>	<p>Phone: +49 (0) 6151 / 93591 - 0 Fax: +49 (0) 6151 / 93591 - 28 E-Mail: info@lipowsky.de Web: www.lipowsky.com</p>
---	------------------------	---	--	---

7 Getting started

7.1 Introduction

This getting started guide will show you how to set up the Baby-LIN to communicate with or monitor the LIN-Bus. Simply follow the next steps.

**Tip**

This guide is made for new Baby-LIN users. If you already have experience with Baby-LIN products or you are an advanced LIN-Bus user then this guide probably is not suited for you.

**Advice**

This guide assumes you are using a Microsoft Windows operating system.

7.2 Installation

Before you can start using the Baby-LIN you have to install several components of the LINWorks software.

If you have not already downloaded the LINWorks software, please download it now from our customer portal: portal.lipowsky.de

**Advice**

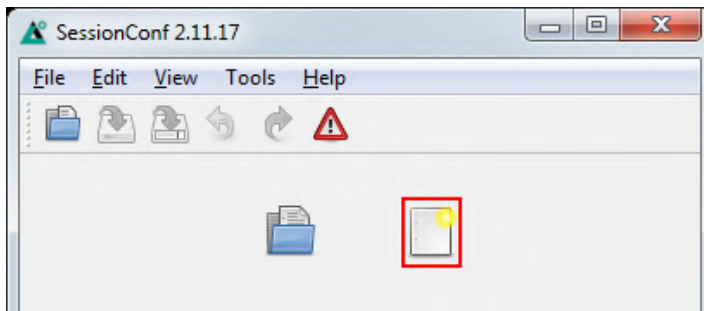
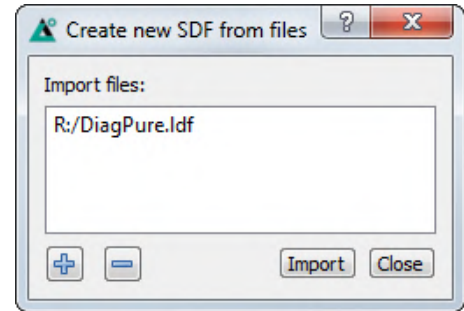
Check chapter "[Downloads](#)" for more information.

The following components are required for this getting started guide:

- Baby-LIN driver
- SessionConf
- SimpleMenu

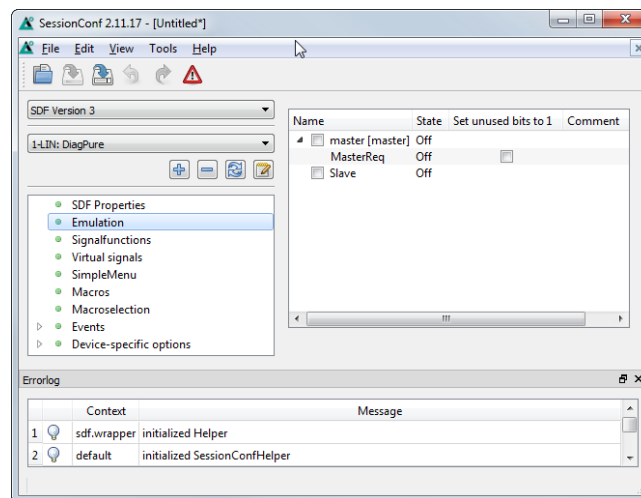
7.3 Create a SDF

First we have to create a SDF. This file contains all information required by the Baby-LIN to simulate nodes on the LIN-Bus or monitor it. Therefore please start the SessionConf and click on the right icon to create a new SDF file. Add your LDF file or DBC file and click on import. To import DBC files you additionally have to select the nodes, frames and signals you want to import.

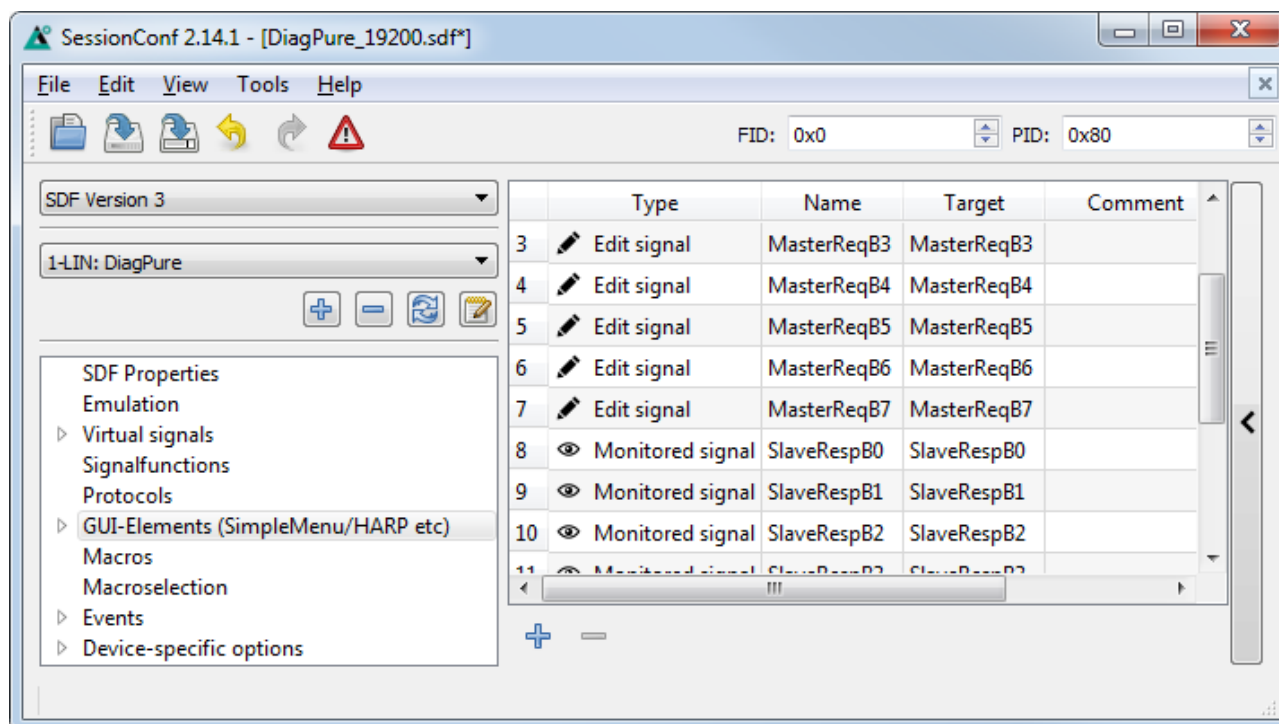


Select Emulation in the navigation menu on the left. Here you can select which nodes you want to be simulated by the Baby-LIN. If you only want to monitor the LIN-Bus, select nothing.

LIN-Bus simulation



Select GUI-Elements in the navigation menu on the left. Here you can add signals you want to monitor.



Advice

There are other ways to monitor frames and signals, but this is a good and configurable starting point.

7.4 Connections

Please establish the following connections:

- Connect the Baby-LIN to your LIN-Bus.



Keep the LIN-Bus voltage within the following range: 8-36 VDC.

- Injury by damaged Baby-LIN.
- The Baby-LIN may get damaged.



Check LIN-Bus node specifications before using voltages above 18 VDC.

If voltages in excess of 18 VDC are used as LIN-Bus supply voltage, it must be ensured that all connected nodes can cope with this voltage level. It is possible, that some nodes will function incorrectly in voltages exceeding 18 VDC, since the LIN specification states a maximum voltage of 18 VDC.

- Injury by damaged LIN-Bus nodes.
- LIN-Bus nodes may get damaged.





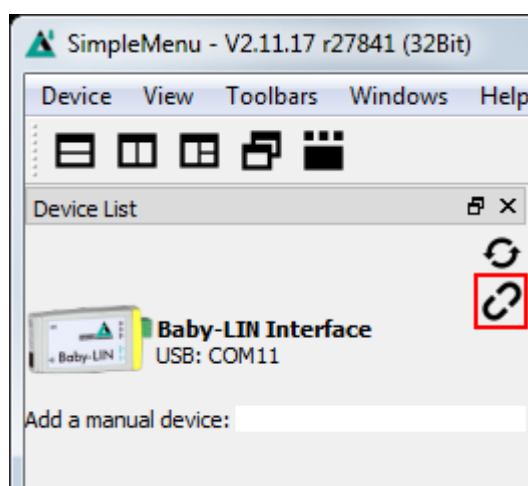
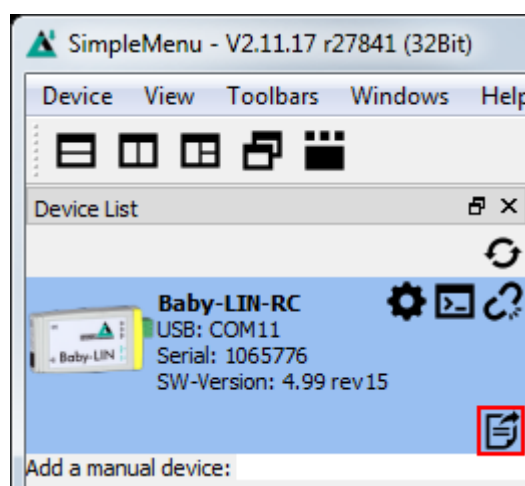
Tip

Make sure all other nodes are connected and running properly.

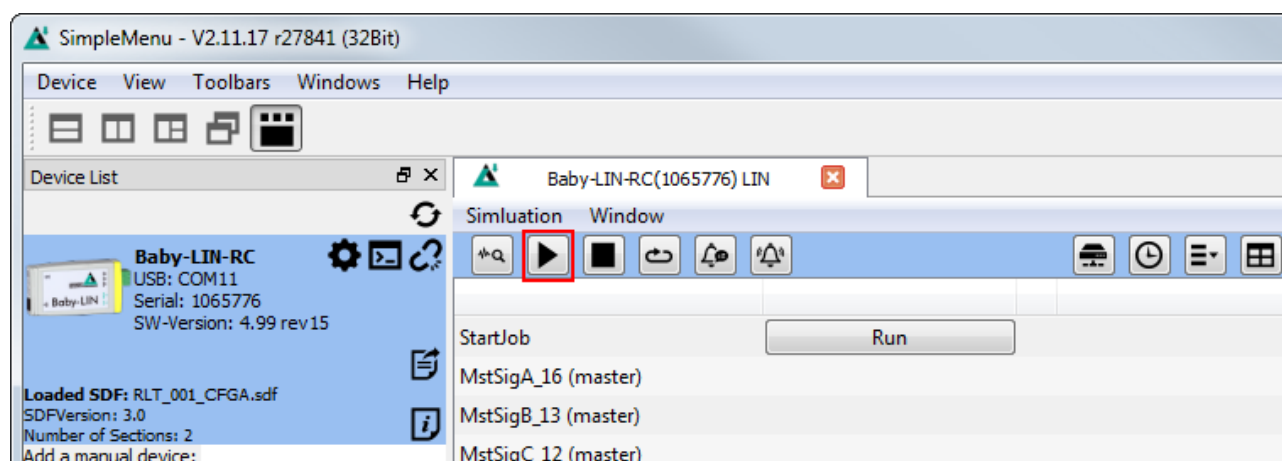
- Connect the Baby-LIN via the provided USB cable to a free USB port of your PC.

7.5 Start the bus communication

Start the SimpleMenu. You should be able to find your Baby-LIN in the device list on the left. Click the connect button and then load the SDF you created earlier.



Now you can see the variables you added to monitor. To start the simulation/monitoring click on the start button.



Now you will see the changes of these signals.



Tip

The Baby-LIN features a lot more features and possibilities and can be used for a wide selection of applications. Keep on reading this manual to learn more about the Baby-LIN.



Tip

The Baby-LIN is not the only LIN-Bus device from Lipowsky Industrie-Elektronik GmbH. Check out our other products at <http://www.lipowsky.com/lipo/products/lin-and-can-systems/>

8 Hardware

8.1 Overview

The following images show you what features the Baby-LIN has to offer. The following features will be shown:

Abbreviation	Description
X	Connectors to access the different interfaces.
LD	LEDs that symbol certain states.



Abbreviation	Type	Description
X1	USB 2.0 type B-Mini	PC connector. Can be used as logic power supply.
X2	Socket for MC 1,5/ 3-ST-3,81 and MCVR 1,5/ 3-ST-3,81	LIN-Bus connector. Can be used as logic power supply.
LD1	Green LED	LIN-Bus voltage indicator.
LD2	Red LED	LIN-Bus communication error.

8.2 Connectors

8.2.1 X1 - PC connector

This connector is a USB type B-Mini. It is used to connect the Baby-LIN to a PC. To use this interface the Baby-LIN USB driver has to be installed on the PC.



Description

The connector uses the default pin assignment of USB type B-Mini.

8.2.2 X2 - LIN-Bus connector

The LIN-Bus interface is available via a MC 1,5/ 3-ST-3,81 connector.

The LIN interface of the Baby-LIN can be operated between voltages of 8-36 VDC.



Pin	Signal	Description
X2-1	GND	LIN-Bus ground
X2-2	LIN	LIN-Bus signal
X2-3	VCC	



CAUTION

Keep the LIN-Bus voltage within the following range: 8-36 VDC.

- Injury by damaged Baby-LIN.
- The Baby-LIN may get damaged.



CAUTION

Check LIN-Bus node specifications before using voltages above 18 VDC.

If voltages in excess of 18 VDC are used as LIN-Bus supply voltage, it must be ensured that all connected nodes can cope with this voltage level. It is possible, that some nodes will function incorrectly in voltages exceeding 18 VDC, since the LIN specification states a maximum voltage of 18 VDC.

- Injury by damaged LIN-Bus nodes.
- LIN-Bus nodes may get damaged.



Advice

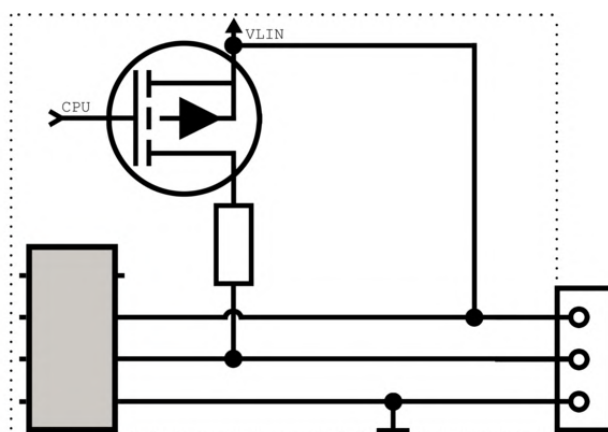
The LIN-Bus interface is galvanically isolated from the logic supply, the USB interface.



Attention

The LIN-Bus supply must be provided by an external power supply and must not be interrupted during the LIN communication.

Equivalent circuit of the LIN-Bus interface:



Electrical characteristics	Value	Unit
Maximum operational voltage	26	V

Electrical characteristics	Value	Unit
Maximum LIN voltage	32	V
Maximum LIN current	450	mA
Maximum LIN supply voltage	51	V
Maximum LIN supply current	550	mA

Electrical characteristics	Voltage threshold for LIN detection	Voltage threshold for LIN transceiver
Minimum	With USB connection: 3.9 V Without USB connection: 5.58 V	6 V

The pull-up resistor of the LIN-Bus driver is switched to 30 k Ω , if the master node is emulated and to 1 k Ω , if only slave nodes are emulated.

8.3 Power supply

The Baby-LIN can be powered by one of the following sources:

- Power supply over the LIN-Bus supply: "X2 - LIN-Bus connector"
Supported power supply voltage: 5V (Typical USB specifications)
- Power supply over USB: USB 2.0 type B-Mini
Supported power supply voltage: 8-36 VDC

The Baby-LIN has a typical power consumption of 70 mA @ 12 VDC.

8.4 LEDs

8.4.1 LD1 - Run LED

This green LED shows the state of the LIN-Bus voltage. It will blink once per second. The duration the LED is on gives you information about the LIN-Bus voltage:

The duration the LED is on	Information
50 ms	No LIN-Bus voltage available.
500 ms	LIN-Bus voltage is available.

If the Baby-LIN enters the transparent mode while the ASCII mode is active, the LED will blink alternatively with "LD2 - Error LED". Check chapter "ASCII mode" and "TRANSPARENT" for more information.

8.4.2 LD2 - Error LED

This red LED shows errors on the LIN-Bus. If it blinks an error like the following has happened on the LIN-Bus:

- Checksum error
- Missing responses

If the Baby-LIN enters the transparent mode while the ASCII mode is active, the LED will blink alternatively with "LD1 - Run LED". Check chapter "[ASCII mode](#)" and "[TRANSPARENT](#)" for more information.

9 Firmware

9.1 Firmware update

9.1.1 Introduction

The firmware is stored in the flash memory of the Baby-LIN and can easily be programmed from a PC. The update process uses an ISP (In-system programming) operation and can be executed in the field.

As we are permanently working on product improvements, the firmware is updated periodically. These updates make new features available and solve problems, which have been discovered by our internal tests or have been reported by customers with earlier versions. All the firmware updates are done in a way, that the updated Baby-LIN will continue to interwork with an already installed, older LINWorks installation. So updating the Baby-LIN firmware does not mean, that you necessarily have to update your LINWorks installation as well. Therefore it is highly recommended to always update your Baby-LIN to the latest available firmware version.

9.1.2 Required software

The following downloads are required to update the firmware of the Baby-LIN:

Download archive	Description
BabyLinDriverSetup.exe	This setup contains the Baby-LIN driver. It is required to update the firmware.
FirmwarePackage-Baby-LIN-Devices.zip	This package contains the firmware for the Baby-LIN as well as the update tool "blprog".

9.1.3 Update the firmware

To update the firmware of the Baby-LIN you have to follow these steps:

- Install the drivers for the Baby-LIN, if you have not already installed them.
- Connect the Baby-LIN with your PC.



Tip

If you have other Baby-LIN products aside the Baby-LIN connected with the PC you will later be asked to select one. This could easily lead to confusions and flashing the wrong device. Therefore we recommend connecting only one Baby-LIN product during a firmware update.

- Unpack the firmware archive.
- Start the "blprog.exe". A command prompt will be opened and guide you through the process.
- If you have other Baby-LIN products aside the Baby-LIN connected with the PC, you will be asked which Baby-LIN product you want to update.



Tip

If you do not know which virtual COM-Port corresponds to your Baby-LIN, you can press "CTRL +C" to stop the firmware update. Now you may disconnect other Baby-LIN products until only your desired Baby-LIN is connected. Now you can restart the "blprog.exe".

- Press the "y" key to confirm the correct firmware and start the flash process.
- Wait until the flashing finished.
- After the flashing has finished press the "Enter" key to exit the tool.

9.2 Stand-alone mode and autostart

9.2.1 Enable the stand-alone mode

The Baby-LIN is able to operate stand-alone without a PC, PLC or operator. To enable this mode several requirements need to be met:

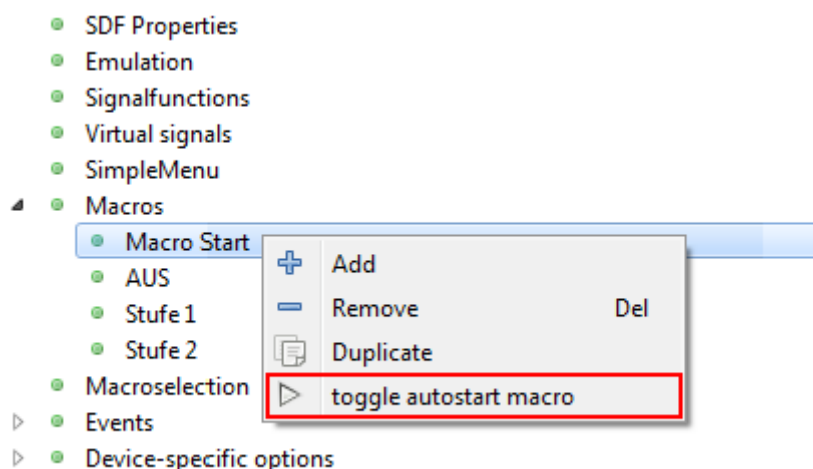
- You need a SDF with a macro that is marked as "autostart" macro.
- The SDF needs to be stored persistently on the Baby-LIN.
- The device needs to be configured to automatically start the autostart macro of a persistently stored SDF when powered up.

9.2.2 Configure the autostart macro

You can mark a macro in a SDF as autostart macro. This means that this macro is started automatically when the SDF is loaded. Usually this macro will start the LIN-Bus communication and perform necessary initialisations.

To mark a macro as autostart macro the following steps are necessary.

- Open your SDF using the SessionConf.
- If this SDF does not already have a macro that you want to use as autostart macro please create one.
- Right click on the macro and select "toggle autostart macro". The macro will now have the "[autostart]" marker.



**Advice**

Please note that only one macro can be marked as autostart macro.

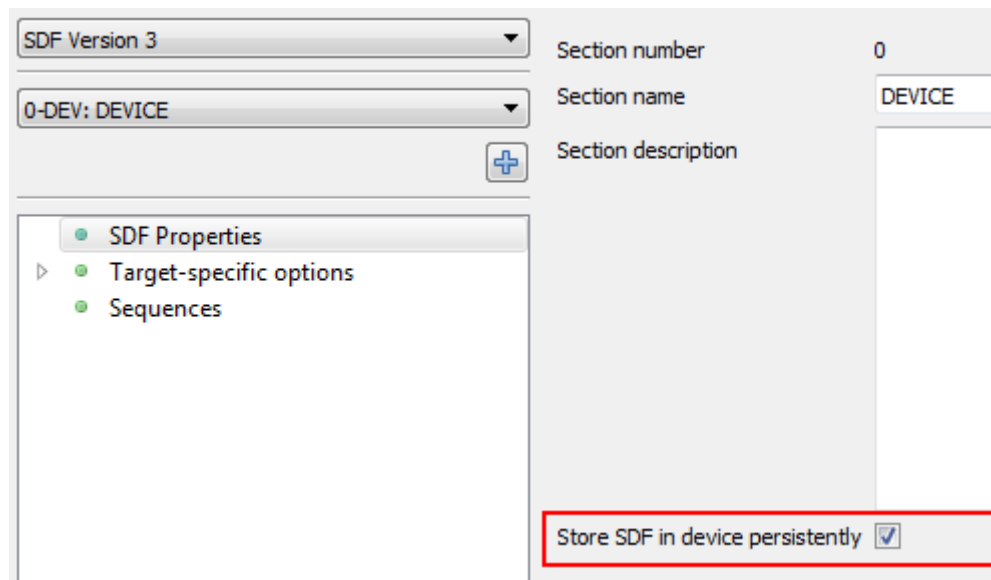
**Tip**

You probably want to start the LIN-Bus within the macro since it is not automatically started by loading the SDF.

9.2.3 Store a SDF persistently

It is possible to store a SDF persistently on the Baby-LIN. Therefore a special option in the SDF must be configured:

- Start the SessionConf and load your SDF.
- In the left menu click on "SDF Properties".
- Check the "Store SDF in device persistently" option.



- Save your SDF and reload it to a device using the SimpleMenu.

**Advice**

The option "Store SDF in device persistently" can always be found in the "SDF Properties". It is independent from the SDF version (SDF v2, SDF v3) or type of section currently selected. This option is global for the whole SDF even though it can be found in every section.

9.2.4 Configure the device to automatically load a SDF and start a macro

The Baby-LIN can be configured to automatically start a autostart macro. The Autostart can be triggered if the device is powered on or if the LIN-Bus is powered on. These target-specific options (also known as target configuration) are persistently stored on the Baby-LIN.

There are 2 important options that affect the autostart feature:

Target-specific option	Possible values	Description
Autostart	Off	No schedule or macro is started when the autostart is triggered.
	Schedule	The first schedule but not the autostart macro is started when the autostart is triggered.
	Schedule + Macro	The first schedule and the autostart macro are started when the autostart is triggered.
	Macro only	No schedule but the autostart macro is started when the autostart is triggered.
Autostart on LIN Power on	Off	An autostart is triggered only when the device is powered on.
	On	An autostart is triggered each time the LIN-Bus is powered on.



Advice

If you want to start another but the first schedule simply select the macro only option and start another than the first schedule from that macro.



Tip

Triggering the autostart each time the LIN-Bus is powered on is especially helpful for applications in which the node connected to the LIN-Bus is changed periodically, e.g. EOL applications. In such a case, the simulation may be started automatically every time a new node is connected and the LIN-Bus voltage is turned on.



Warning

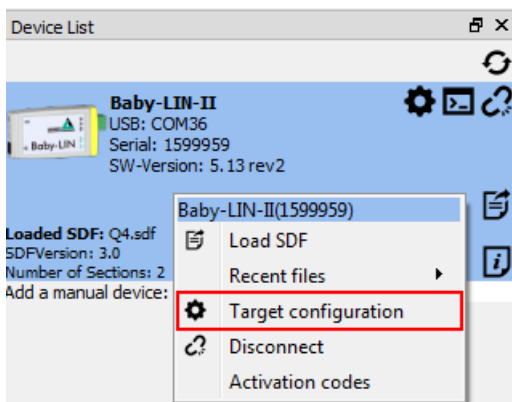
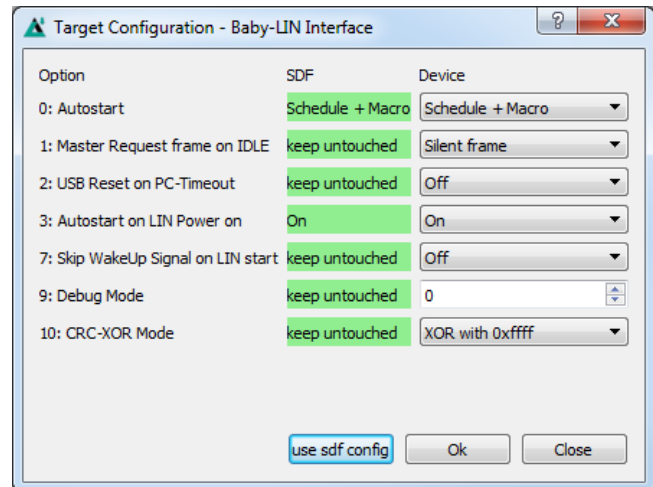
The option "Autostart on LIN Power on" is not required for the Baby-LIN if the LIN voltage is also used as the logic supply.



Version incompatibility

Newer revisions of the Baby-LIN support the new SDF v3 file format. One of the features of SDF v3 is the setting of preset values for target-specific options.

The target-specific options can be set using the SimpleMenu. Right click on the connected device and click on "Target configuration". In the following dialog you can change the values in the device. For comparison reasons the preset values of the SDF are shown.



9.3 Logging

9.3.1 Introduction

The Baby-LIN supports the logging of the frames on the LIN-Bus. The following possibilities to log the bus data are available:

- The bus data can be logged using the SimpleMenu if the Baby-LIN is connected to a PC.

The logging using the SimpleMenu is described here: Check chapter ["SimpleMenu"](#) for more information..

9.3.2 Log data targets

9.3.2.1 Overview

The Baby-LIN can write log data to the following targets:

Target	Description
"SimpleMenu"	The logging data can be viewed and written into a file by using the "SimpleMenu" .

9.3.2.2 SimpleMenu

The ["SimpleMenu"](#) can show the bus communication of the Baby-LIN in the frame monitor. Additionally that communication can be written into log files in the ["Binary format"](#).

9.3.3 Log data formats

9.3.3.1 Format overview

The Baby-LIN is writing log files in the following formats:

Format	Description	Frame size ^(*)
--------	-------------	---------------------------

(*) This is the the typical size that an 8 byte frame will require in the log file. Additional header and other information will increase the log file size furthermore.

9.3.3.2 Binary format

This format uses a proprietary binary data format to store the log data. It is optimized for speed and a low file size. This file can be viewed, edited and converted into other formats using the ["LogViewer"](#).

10 SDF - The session configuration file

10.1 Overview

A SDF file contains the configuration to setup a Baby-LIN. It contains among other things the following information:

Element	Description	Chapter
Sections	A section contains the configuration for a single LIN channel.	"Channels and sections"
Device section	A section not assigned to a bus interface but the Baby-LIN in common.	"Device section"
Section properties	Certain settings to configure a section.	"Section Properties"
Bus description	For LIN sections Bus description based on a LDFile or NCFFile.	"Bus description"
	For CAN sections Bus description based on a DBC file or ARXML file.	
Emulation	The emulation setup describing which nodes are emulated by the Baby-LIN.	"Emulation"
Virtual signals	Virtual signals can store values just like bus signals, but they do not appear on the bus. They can be used for many different tasks like: <ul style="list-style-type: none"> • Temporary values, like counters • Store constants • Operands and results from calculations • etc. 	"Virtual Signals"
System variables	System variables are virtual signals with predefined names and implement certain system functionality like e.g.: <ul style="list-style-type: none"> • Timer • Memory • Protocol settings • Digital I/O operations • System information • etc. 	"System variables"
Signalfunctions	Signalfunctions realize certain functionalities: <ul style="list-style-type: none"> • Signal and frame based counter • Checksums • Regular CRCs 	"Signalfunctions"
GUI-Elements	The GUI-Elements define, which items are shown, when the SDF file is downloaded using the SimpleMenu. The following elements can be added: <ul style="list-style-type: none"> • Show signal values of monitored signals • Start macros • Edit signal values of emulated bus signals and virtual signals • Select a macro from a macroselection 	"GUI-Elements"

Element	Description	Chapter
Macros	<p>Macros are sequences of commands, that can be used to implement custom functionalities. They can especially be used to implement temporal sequences. The macro commands cover many areas like:</p> <ul style="list-style-type: none"> • Signal manipulation • Flow control mechanisms • Table manipulation • Bus operations • Macro operations • etc. 	"Macros"
Macroselection	Macroselections are lists of macros and can be used for grouping and selective execution.	"Macroselections"
Events	<p>Events can be used to trigger actions.</p> <ul style="list-style-type: none"> • Frame events • Device specific events • Signals events • etc. 	"Events"



Warning

Such a stored SDFFile can not be extracted from the Baby-LIN again.
Therefor you should make sure you have archived the SDFFiles you download.

10.2 Channels and sections

10.2.1 Introduction

With the introduction of the second generation of Baby-LIN products and SDF-V3, the full support for multiple bus interfaces was added.

Each bus interface on a Baby-LIN product is now represented by a channel. Each channel has its own set of SDF features like virtual signals, signal functions, macros, events etc..

A SDFFile in the SDF-V2 format describes the configuration of a single bus. This configuration is now called a section and a SDFFile in the SDF-V3 format can contain more than one section.

The following chapters explain channels, section and the mapping from sections to channels when a SDFFile is loaded.



Warning

The Baby-LIN can only process a SDF-V2SDFFile. The following explanations should be read nonetheless since you may encounter these terms.

10.2.2 Sections

A channel is configured by a section from a SDFFile. Each section has its own set of SDF features like virtual signals, signal functions, macros, events etc..

A SDFFile in SDF-V2 format only has a single section. That section contains the configuration for the single LIN channel and the device channel.

A SDFFile in SDF-V3 format always has a device section. Additionally it can keep any number of LIN and CAN sections.

Section type	Channel type	Availability	Description
Device section	Device channel	Only in SDF-V3	This represents the device itself. It contains configurations that are independent from channels. In the SDF-V2 format these configurations could be found within the LIN section.
LIN section	LIN channel	SDF-V2 and SDF-V3	This channel represents a LIN-Bus interface. In the SDF-V2 format this section also contained the configurations that are independent from channels.
CAN section	CAN channel	Only in SDF-V3	This channel represents a CAN-Bus interface.

10.2.3 Channels

Each bus interface of a Baby-LIN product is represented by a channel. Each channel is configured with a single section from a SDFFile. The following channel types are available:

Channel type	Description
Device channel	This channel represents the device itself. It is used for bus independent functions. It is made available by using the second generation software (LINWorks v2, BLC interface of the Baby-LIN-DLL v8 or higher).
LIN channel	This channel represents a LIN-Bus interface. One LIN-Bus interface was always available for the Baby-LIN products of the first generation. More than one LIN-Bus interfaces are available for certain devices of the Baby-LIN products of the second generation.
CAN channel	This channel represents a CAN-Bus interface. Fully featured CAN-Bus interfaces are available only for certain devices of the Baby-LIN products of the second generation.

The following channels are available for the Baby-LIN:

Index	Channel	Connector
0	Device	The device channel has no special bus interface.
1	LIN	"X2 - LIN-Bus connector"



Warning

The channel index gives you the order of the channels.

This channel index is used by the "Baby-LIN-DLL".

10.2.4 Mapping of sections to channels

When a SDFFile is loaded, its sections must be mapped to the channels of the Baby-LIN. The following algorithm is applied:

The sections of the SDFFile are loaded in order from the beginning to the end. They are mapped in the order they appear in the SDFFile. A LIN section is mapped to the next unused and activated LIN channel. A CAN section is mapped to the next unused and activated CAN channel. If no more LIN or CAN channels are available, the remaining sections are discarded. The device section is always mapped to the device channel.

Here is an example for the Baby-LIN:

SDFFile		→	Baby-LIN	
Section type	Section name		Channel type	Mapped section
CAN	CAN-Section-1		LIN	LIN-Section-1
LIN	LIN-Section-1			
LIN	LIN-Section-2			
CAN	CAN-Section-2			
LIN	LIN-Section-3			



Version incompatibility

Older versions of the firmware may return errors, if the SDFFile contains sections, that can not be mapped to channels. You should check the following typical reasons:

- The SDFFile was developed for or used by another Baby-LIN product, that features more LIN or CAN channels.

10.3 Section Properties

Each section has some common properties and a set of individual properties and information, that describe that section. The following table shows these properties:

Property	Availability	Description
Section number	All	This is the index of the section within the SDFFile.
Section name	All	Each section can have a name, to easily identify it within the Session-Conf.
Section description	All	Here you can freely enter a description, e.g. comments, a version history or something similar.
Store SDF in device persistently	All	This global property is shown in all sections even though it is just one setting for the whole SDFFile. This is a tribute due to the importance of this setting.
LDF protocol version	LIN	This shows the LIN protocol version of the LDFFile, this section is based on.
LDF language version	LIN	This shows the LIN language version of the LDFFile, this section is based on.
Speed [Bit/s]	LIN & CAN	This is the baudrate of the bus. Here you can overwrite the value from the LDFFile/DBC file.

10.4 Device section

10.4.1 Target configuration and target-specific options

10.4.1.1 Introduction

The Baby-LIN has an persistent memory to save configuration settings. This target configuration consists out of target-specific options. There are two possibilities to set them:

- Directly change the target configuration within the Baby-LIN using the SimpleMenu.
- Set the target-specific options in an SDF using the SessionConf. If the SDF is loaded into the Baby-LIN the target-specific options of the SDF can overwrite the device configuration.

10.4.1.2 List of options

The following list shows all target-specific options available for the Baby-LIN:

Target-specific option	Description	
Autostart	This option determines if a schedule or macro should start when an autostart is triggered. This option is important if you want to use the Baby-LIN in the stand-alone mode. The trigger for an autostart depends on the option "Autostart on LIN Power on".	
	Value	Description
	Off	No schedule or macro is started when the autostart is triggered.
	Schedule	The first schedule but not the autostart macro is started when the autostart is triggered.
	Schedule + Macro	The first schedule and the autostart macro are started when the autostart is triggered.
	Macro only	No schedule but the autostart macro is started when the autostart is triggered.
	<div>Advice Check out chapter "Stand-alone mode and autostart" for more information.</div>	
Master Request frame on IDLE	This option determines what happens if the signals of a Master Request frame have not changed.	
	Value	Description
	Silent frame	No Master Request frame is sent by the Baby-LIN.
	Dummy frame (8 * 0xFF)	A Master Request frame with all data bytes set to 0xFF is sent by the Baby-LIN.

Target-specific option	Description						
USB Reset on PC-Timeout	This option determines if the USB chip is reset if the connection to the PC times out.						
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>Off</td><td>Nothing happens if the connection to the PC times out.</td></tr><tr><td>On</td><td>The USB chip of the Baby-LIN is reset if the connection to the PC times out.</td></tr></table>	Value	Description	Off	Nothing happens if the connection to the PC times out.	On	The USB chip of the Baby-LIN is reset if the connection to the PC times out.
	Value	Description					
Off	Nothing happens if the connection to the PC times out.						
On	The USB chip of the Baby-LIN is reset if the connection to the PC times out.						
Autostart on LIN Power on	This option determines when an autostart is triggered. The action that is triggered by an autostart depends on the option "Autostart".						
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>Off</td><td>An autostart is only triggered after the Baby-LIN has started.</td></tr><tr><td>On</td><td>An autostart is triggered after the Baby-LIN has started and each time the LIN-Bus power switches from off to on.</td></tr></table>	Value	Description	Off	An autostart is only triggered after the Baby-LIN has started.	On	An autostart is triggered after the Baby-LIN has started and each time the LIN-Bus power switches from off to on.
	Value	Description					
Off	An autostart is only triggered after the Baby-LIN has started.						
On	An autostart is triggered after the Baby-LIN has started and each time the LIN-Bus power switches from off to on.						
Skip WakeUp Signal on LIN start	This option determines if a wake up signal is sent at the LIN start.						
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>Off</td><td>The wake up signal is sent at the LIN start.</td></tr><tr><td>On</td><td>The wake up signal is skipped at the LIN start.</td></tr></table>	Value	Description	Off	The wake up signal is sent at the LIN start.	On	The wake up signal is skipped at the LIN start.
	Value	Description					
Off	The wake up signal is sent at the LIN start.						
On	The wake up signal is skipped at the LIN start.						
Debug Mode	<div><div>Possible values: 0 - 3</div><div><div></div><div><div>Attention</div><div>This option is only used for debugging purposes. Do not change this value unless explicitly instructed by Lipowsky Industrie-Elektronik GmbH.</div></div></div></div>						
CRC-XOR Mode	This option determines the CRC-XOR value for signal functions with mode "CRC-8" and "CRC-16".						
	<table><tr><th>Value</th><th>Description</th></tr><tr><td>XOR with 0xffff</td><td>The XOR value is 0xffff.</td></tr><tr><td>XOR with 0x0000</td><td>The XOR value is 0x0000.</td></tr></table>	Value	Description	XOR with 0xffff	The XOR value is 0xffff.	XOR with 0x0000	The XOR value is 0x0000.
	Value	Description					
XOR with 0xffff	The XOR value is 0xffff.						
XOR with 0x0000	The XOR value is 0x0000.						
<div><div></div><div><div>Version incompatitbility</div><div>This option is only used if SDF v2 are used.</div><div>SDF v3 keep the CRC-XOR value within the file.</div></div></div>							

10.4.1.3 Setting the target-specific options

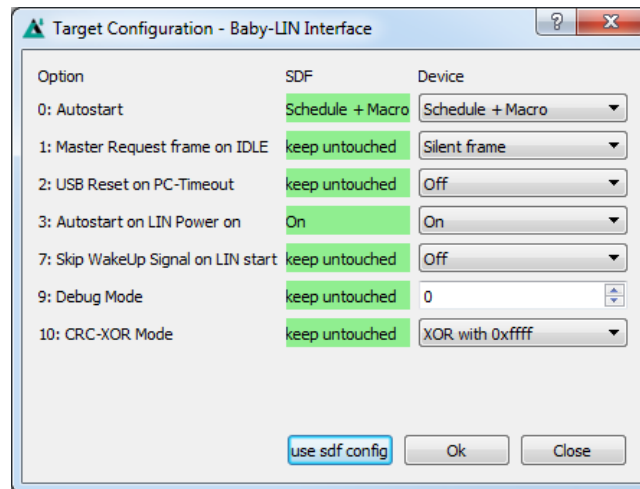
10.4.1.3.1 Setting the target-specific options directly on the device

You can change the target configuration of the Baby-LIN using the SimpleMenu:

- Connect your Baby-LIN with a PC.
- Start the SimpleMenu and connect to your Baby-LIN.
- Right click on the Baby-LIN in the device list and select "Target configuration".

A dialog will open and show you the current configuration of the Baby-LIN in the "Device" column. Here you can change each target-specific option.

If you downloaded a SDF to the Baby-LIN after you connected to it, the "SDF" column will show you the settings of that SDF.



10.4.1.3.2 Setting the target-specific options indirectly via a SDF v3

If a Baby-LIN requires certain target-specific options or you want to set the target configuration of a lot of Baby-LINs it is possible to set them in a SDF v3. Each time that SDF is downloaded to a device the target-specific options of the SDF are used and overwrite the target configuration of the device. Each option can also be set to not overwrite the device's option.

- Start the SessionConf and load a SDF v3 or create a new one.
- Select the device section.
- Expand the "Target-specific options" element within the tree on the left.

The target-specific options are divided into a global area with options that are available on each(*) Baby-LIN device and options that are only available on certain devices. Each option additionally has the setting "keep untouched". This setting is the default value and ensures that the corresponding value in the Baby-LIN device is not changed. Therefore it is possible to set only single target-specific options.

(*) Exception: The Baby-LIN-MB and Baby-LIN-MB-II devices are not using any target-specific options.

10.4.2 Sequences

Sequences can only be used on the Baby-LIN-MB and the Baby-LIN-MB-II.

10.5 Bus description

The bus description shows you the basic information about the bus. These information can come from the following sources:

LIN section	CAN section
<ul style="list-style-type: none"> • LDFFile • NCFFile 	<ul style="list-style-type: none"> • DBC file • ARXML file

When a DBC file or ARXML file is imported into a CAN section, not all elements must be imported. Therefore the bus description may contain only a subset of a bus description file.

The bus description shows all bus elements, that are imported:

Element	Description
Nodes	<p>Shows all imported nodes. Any number of these nodes can be emulated by the Baby-LIN. Each node has the following properties:</p> <ul style="list-style-type: none"> • Node name: The name of the node. Can be changed. • Node Nr.: The index of the node.
Signals	<p>Shows all signals in imported frames.</p> <ul style="list-style-type: none"> • Signal name: The name of the node. Can be changed. • Signal number: The index of the signal. • Length: The bit length of a signal. • Initial value: The initial value the signals has, when the bus is started. • Mappings: A list of frames, the signal is mapped in.
Frames	<p>Shows all imported frames.</p> <ul style="list-style-type: none"> • Frame name: The name of the node. Can be changed. • Frame number: The index of the frame. • Length: The length of the frame data. • Frame ID: The ID of the frame. • 29 Bit Frame ID: Flag, that shows, if the Frame ID is a 29 Bit Frame ID instead of a 11 Bit Frame ID. • Publishing node: The node, that is sending this frame. • Mappings: A list of all signal mappings.
Schedules	<p>Shows all imported schedules.</p> <ul style="list-style-type: none"> • Schedule name: The name of the node. Can be changed. • Schedule number: The index of the schedule. • A list of all frames in the schedule and the interval (delay) between them.

10.6 Emulation

10.6.1 Introduction

Here you can setup, which nodes should be emulated by the Baby-LIN. If the Baby-LIN emulates a node, it will take part in the communication, like the emulated node would do by sending the frames, the node publishes. By default the signals within these frames will contain the default signal values. By adding logic to the SDFFile, these values can be changed. Of course the Baby-LIN can also react to frames and signals it receives.

The following list shows typical emulation setups. Of course any other scenarios are possible.

Number of emulated nodes	Application
None	This setup can be used, if the Baby-LIN should react to the bus communication, but not take part. The Baby-LIN then acts as a monitor.
1	This setup can be used, if the Baby-LIN should emulate a single node, that is not available. This setup is often used in the development.
All but 1	This setup is called restbus simulation. The Baby-LIN emulates all nodes of the bus, but one. This is usually used to operate a single node for tests, EOL applications or installations, when no ECU is available.
All	This setup can be used to see, how the bus communication works.

10.6.2 LIN emulation

The following settings can be made to setup the emulation of a LIN section.

- Each node can be configured to have one of two states:

Node state	Description
Off	The frames of the node are not published by the Baby-LIN. The frames and their data will still be visible to the user if they appear on the LIN-Bus.
Emulated	The frames of the node will be published by the Baby-LIN.

- Each emulated frame in a LIN section can be configured to set unused bits to 1 instead of 0.

If the Baby-LIN emulates the master node, it will put the header of the frames on the LIN-Bus based on the schedule it executes right now. Additionally it will publish all frames of all emulated nodes on the LIN-Bus.

If the Baby-LIN emulates no master node, it can only publish the frame data, if another node is the master and puts the header of the frames on the LIN-Bus.

10.7 Tables

Tables can be used to store large amounts of data of the same kind. They can especially be used to replace virtual signals, that are used to store constant values.

Each Table consists of one or more columns. The type of data, that can be stored in that column can be defined using the first lines by the following properties:

Line	Name	Description
1		The index of the column.
2	Name	The name of the column.
3	Type	This is the basic type of data, that can be stored in this column. Possible values are: <ul style="list-style-type: none"> String: A text. Signed: A signed integer. Unsigned: An unsigned integer.

Line	Name	Description
4	Bit width	A number between 1 and 64. This value is only relevant for the types Signed and Unsigned.
5	Format	This is the format, the data will be shown as. Possible values are: <ul style="list-style-type: none"> Decimal Hexadecimal Binary

Each table can have any number of columns and rows. Additionally whole tables can be imported from and exported to CSV files.

Table cells can be evaluated and written using macro commands. Check chapter "[Macros](#)" for more information.

10.8 Virtual Signals

10.8.1 Introduction

Signals are small sections of LIN frames. They are defined in the underlying LDFFile.

The SDFFile provides the opportunity to define virtual signals. They are mostly handled like normal bus signals. The main difference is that they are defined in the SDFFile instead of the LDFFile.

The virtual signals can be used for many purposes. Here are some common use cases:

- They can be used as buffer or interim results for calculations.
- They can be used as auxiliary variables like counters or state variables.
- They can be used as trigger for signal events.
- They give access to system specific resources. Check chapter "[System variables](#)" for more information.

10.8.2 Sizes

The size of a virtual signal in the SDF-V2 format is for compatibility reasons always 16 Bit and unsigned. The SDF-V3 allows for sizes between 1 to 64 Bit and also signed values.

An exception to this scheme are system variables. They are always 32 Bit and unsigned. Check chapter "[System variables](#)" for more information.

	Size	Sign
SDF-V2	16 Bit	Unsigned
SDF-V3	1 Bit to 64 Bit	Signed or Unsigned
System variables	32 Bit	Unsigned

10.8.3 Scope

The scope of a virtual signal is limited to the section it is defined in. Therefore a channel has one instance for every virtual signal in the section, that was loaded to that channel.

It is not possible to share values of regular virtual signals over multiple sections, even if the name of a virtual signal is identical.

**Warning**

Since each channel has its own instance of system variables, single system specific resources may be accessed concurrently by multiple channels.

Please organize your system variable access accordingly. One solution could be, to access single system resources only from a single channel.

10.8.4 Reset on bus start

Virtual signals can react differently to a bus start. The following table gives you an overview:

SDF version	Behaviour
SDF-V2	All virtual signals are reset to their initial values, when the bus (re-)starts.
SDF-V3	The SDF-V3 allows the user to change the behaviour of virtual signals. Virtual signals can now keep their value, when the bus (re-)starts.

**Version incompatibility**

To keep the backward compatibility, the state of all virtual signals is set to reset on bus start, when converting a SDF-V2 to a SDF-V3.

Accordingly you can only convert a SDF-V3 to a SDF-V2, if the state of all virtual signals is set to reset on bus start.

**Warning**

Some system variables are ignoring the state and will always behave in a specific way. Check chapter "[System variables](#)" for more information.

10.9 System variables

10.9.1 Introduction

System variables are virtual signals with reserved names. They allow access to system specific resources like timers, inputs and outputs etc..

Before you can use these system variables, they need to be added to the virtual signals. You can easily identify the system variables since they always start with the "@@" string.

The size of system variables is always 32 Bit and they are always unsigned.



Warning

Some system variables ignore the default reset on bus start behaviour or setting.

The "[Quick reference](#)" will give you an overview.

There are 3 types of system variables:

Type	Description
Single	The name represents a single variable, that is available with exactly that name. Example: <code>SYSTIMER_RTC_HOUR</code> .
Prefix	The name of the variable uses a defined prefix and a custom postfix. The combination represents an individual variable. Example: <code>SYSTIMER_DOWN_MYTIME</code> .
Group	The name of the variable consists of the group name and a number. The combination represents an individual variable. Example: <code>SYSDIGOUT1</code>

System variables do not all react in the same way on bus starts. Some are reset to their initial value like normal virtual signals while others are never reset:

Reset behaviour	Description
Default	These system variables have the same behaviour like normal virtual signals. Check chapter " Reset on bus start " for more information.
Never reset	The value of these variables is never reset on the bus start. Even in SDF-V2 or if accordingly configured in SDF-V3.

Some system variables are read-only. Setting them will be ignored.



Warning

Since each channel has its own instance of system variables, single system specific resources may be accessed concurrently by multiple channels.

Please organize your system variable access accordingly. One solution could be, to access single system resources only from a single channel.

10.9.2 Quick reference

The following tables give you an overview over the available system variables.

Category	System variable	Description	Name type	Reset behaviour	Read-only
Timing	"@@SYSTIMER_UP"	Up counting timer.	Prefix	Default	
Timing	"@@SYSTIMER_FAST_UP"	Fast up counting timer.	Prefix	Default	
Timing	"@@SYSTIMER_DOWN"	Down counting timer.	Prefix	Default	
Timing	"@@SYSTIMER_FAST_DOWN"	Fast down counting timer.	Prefix	Default	
Timing	"@@SYSTIMER_RTC_HOUR"	The hour of the real-time clock.	Single	Never reset	

Category	System variable	Description	Name type	Reset behaviour	Read-only
Timing	"@@SYSTIMER_RTC_MIN"	The minute of the real-time clock.	Single	Never reset	
Timing	"@@SYSTIMER_RTC_SEC"	The seconds of the real-time clock.	Single	Never reset	
LIN header	"@@SYSBREAKTIME"	Sets the break pulse duration.	Single	Default	
LIN header	"@@SYSBREAKPAUSE"	Sets the pause between the break pulse and the next sync character.	Single	Default	
LIN header	"@@SYSINTERBYTETIME"	Sets a inter byte duration between characters.	Single	Default	

10.9.3 Timing

10.9.3.1 @@SYSTIMER_UP



Advice

This system variable name is just a prefix. To create a real system variable it is necessary to add a postfix. You can use any string as postfix. There are no limitations regarding the name or number of system variables of this type.

The @@SYSTIMER_UP system variable prefix will assign an up counter with the following properties:

- It will start counting if its value is unequal 0.
- It will count to a maximum number of 65535.
- The timer tick period is 1 second.

10.9.3.2 @@SYSTIMER_FAST_UP



Advice

This system variable name is just a prefix. To create a real system variable it is necessary to add a postfix. You can use any string as postfix. There are no limitations regarding the name or number of system variables of this type.

The @@SYSTIMER_FAST_UP system variable prefix will assign an up counter with the following properties:

- It will start counting if its value is unequal 0.
- It will count to a maximum number of 65535.
- The timer tick period is 10 milliseconds.

10.9.3.3 @@SYSTIMER_DOWN



Advice

This system variable name is just a prefix. To create a real system variable it is necessary to add a postfix. You can use any string as postfix. There are no limitations regarding the name or number of system variables of this type.

The @@SYSTIMER_DOWN system variable prefix will assign a down counter with the following properties:

- It will start counting if its value is unequal 0.
- It will count to a minimum number of 0.
- The timer tick period is 1 second.

10.9.3.4 @@SYSTIMER_FAST_DOWN



Advice

This system variable name is just a prefix. To create a real system variable it is necessary to add a postfix. You can use any string as postfix. There are no limitations regarding the name or number of system variables of this type.

The @@SYSTIMER_FAST_DOWN system variable prefix will assign a down counter with the following properties:

- It will start counting if its value is unequal 0.
- It will count to a minimum number of 0.
- The timer tick period is 10 milliseconds.

10.9.3.5 @@SYSTIMER_RTC_HOUR



Advice

This system variable name identifies a single variable. To use the functionality of this system variable the name of the virtual signal must match exactly without any postfixes.

This system variable contains the hour value of the internal real-time clock.



Warning

The real-time clock is not battery powered. It will lose its value when the device is unplugged from the supply voltage.



Advice

This variable will start with the value 0, when the Baby-LIN is powered on.

**Advice**

The hour value of the time can be changed by assigning a value to this variable.

**Advice**

The clock consisting of @@SYSTIMER_RTC_HOUR, @@SYSTIMER_RTC_MIN and @@SYSTIMER_RTC_SEC will run to 23:59:59 and then roll over to 00:00:00.

10.9.3.6 @@SYSTIMER_RTC_MIN

**Advice**

This system variable name identifies a single variable. To use the functionality of this system variable the name of the virtual signal must match exactly without any postfixes.

This system variable contains the minute value of the internal real-time clock.

**Warning**

The real-time clock is not battery powered. It will lose its value when the device is unplugged from the supply voltage.

**Advice**

This variable will start with the value 0, when the Baby-LIN is powered on.

**Advice**

The minute value of the time can be changed by assigning a value to this variable.

**Advice**

The clock consisting of @@SYSTIMER_RTC_HOUR, @@SYSTIMER_RTC_MIN and @@SYSTIMER_RTC_SEC will run to 23:59:59 and then roll over to 00:00:00.

10.9.3.7 @@SYSTIMER_RTC_SEC

**Advice**

This system variable name identifies a single variable. To use the functionality of this system variable the name of the virtual signal must match exactly without any postfixes.

This system variable contains the seconds value of the internal real-time clock.

**Warning**

The real-time clock is not battery powered. It will lose its value when the device is unplugged from the supply voltage.

**Advice**

This variable will start with the value 0, when the Baby-LIN is powered on.

**Advice**

The seconds value of the time can be changed by assigning a value to this variable.

**Advice**

The clock consisting of @@SYSTIMER_RTC_HOUR, @@SYSTIMER_RTC_MIN and @@SYSTIMER_RTC_SEC will run to 23:59:59 and then roll over to 00:00:00.

10.9.4 LIN header

10.9.4.1 Introduction

Sometimes it is necessary to change LIN timing parameters within a frame to values, different to the typical values defined in the LIN specification.

One example is the usage of a longer break time pulse length for nodes using SNPD (Slave Node Position Detection) with the shunt method.

10.9.4.2 @@SYSBREAKTIME

**Advice**

This system variable name identifies a single variable. To use the functionality of this system variable the name of the virtual signal must match exactly without any postfixes.

The @@SYSBREAKTIME system variable allows the definition of the break pulse duration in multiples of the bit duration.

Setting this system variable will change the break time to the value in bit time. The default value is 16 resulting in a break time of 16 * bit time. Setting a value of 0 will reset the break time to the default value of 16 * bit time.

Property	Description
Unit	Multiples of the bit time
Default value	16 => 16 * bit time
Setting to zero	Resets to default value (16)

10.9.4.3 @@SYSBREAKPAUSE

**Advice**

This system variable name identifies a single variable. To use the functionality of this system variable the name of the virtual signal must match exactly without any postfixes.

The @@SYSBREAKPAUSE system variable allows the definition of the idle time between the break pulse and the following sync character.

Setting this system variable will change the break pause to the value in 0.1 bit time. The default value is 15 resulting in a break pause of $1.5 * \text{bit time}$. Setting a value of 0 will reset the break pause to the default value of $1.5 * \text{bit time}$.

Property	Description
Unit	0.1 * bit time
Default value	15 => $1.5 * \text{bit time}$
Setting to zero	Resets to default value (15)
Max. value	100

10.9.4.4 @@SYSINTERBYTETIME

**Advice**

This system variable name identifies a single variable. To use the functionality of this system variable the name of the virtual signal must match exactly without any postfixes.

The @@SYSINTERBYTETIME system variable allows the definition of the inter byte time between the characters sent by the Baby-LIN.

Setting this system variable will change the inter byte time to the value in μs . The default value is 0 resulting in an inter byte time of 0 μs .

Property	Description
Unit	μs
Default value	0 => 0 μs

10.10 Signalfunctions

10.10.1 Overview

Signalfunctions are algorithms that calculate the value of signals. Since these algorithms are calculated by the Baby-LIN, they are usable for time critical functionalities, like implementing frame counters or CRCs.

The signalfunctions are triggered by different conditions.

Trigger	Description	Signalfunction
Time based	These signalfunctions are triggered periodically after a given interval.	• "Counter - Time based"
Incoming frame	These signalfunctions are triggered, when a certain frame is received.	• "Counter - Time based"
Outgoing frame	These signalfunctions are triggered, before a certain frame is sent.	• "Checksum" • "XOR" • "CRC - 8 Bit" • "CRC - 16 Bit"

10.10.2 Counter - Time based

This signalfunction realizes a counter that is incremented periodically.

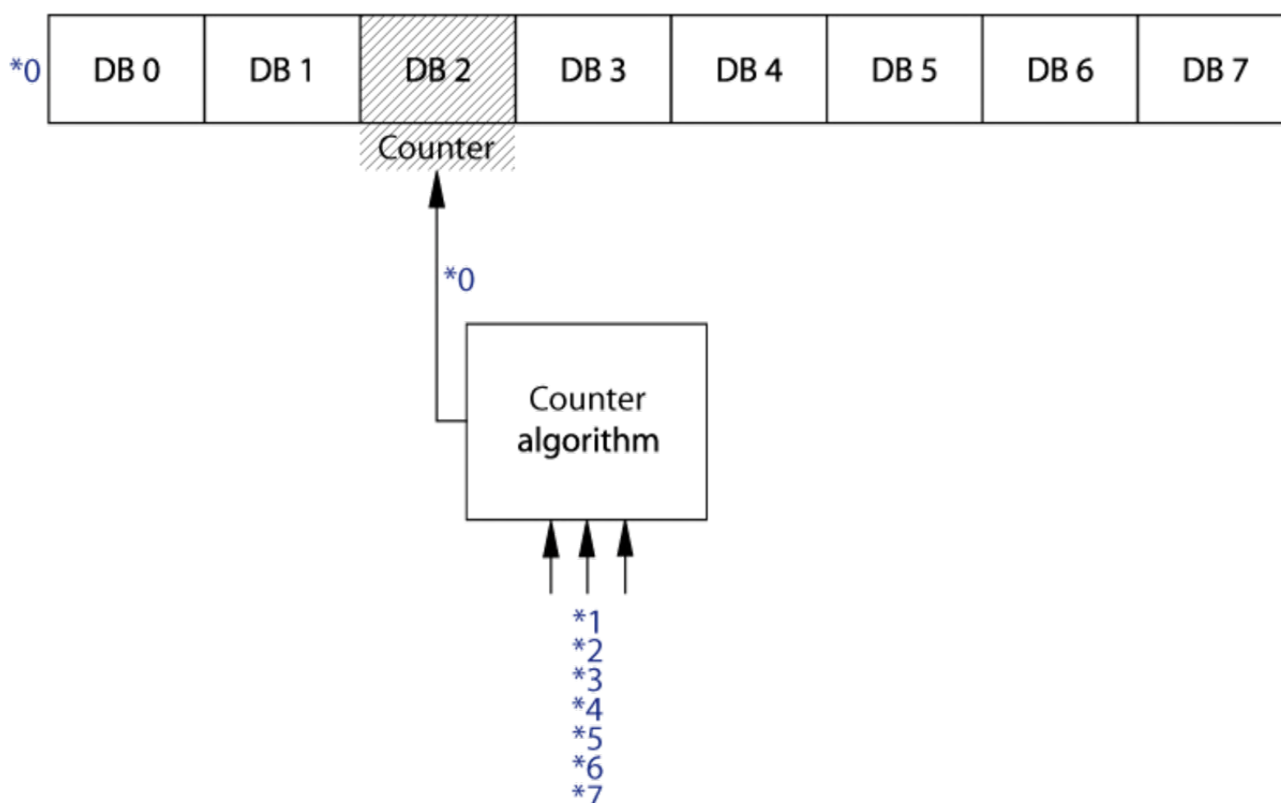


Advice

This functionality can also be realized using timer system variables.

Check chapter "Timing" for more information.

The following diagram shows the properties of the algorithm:



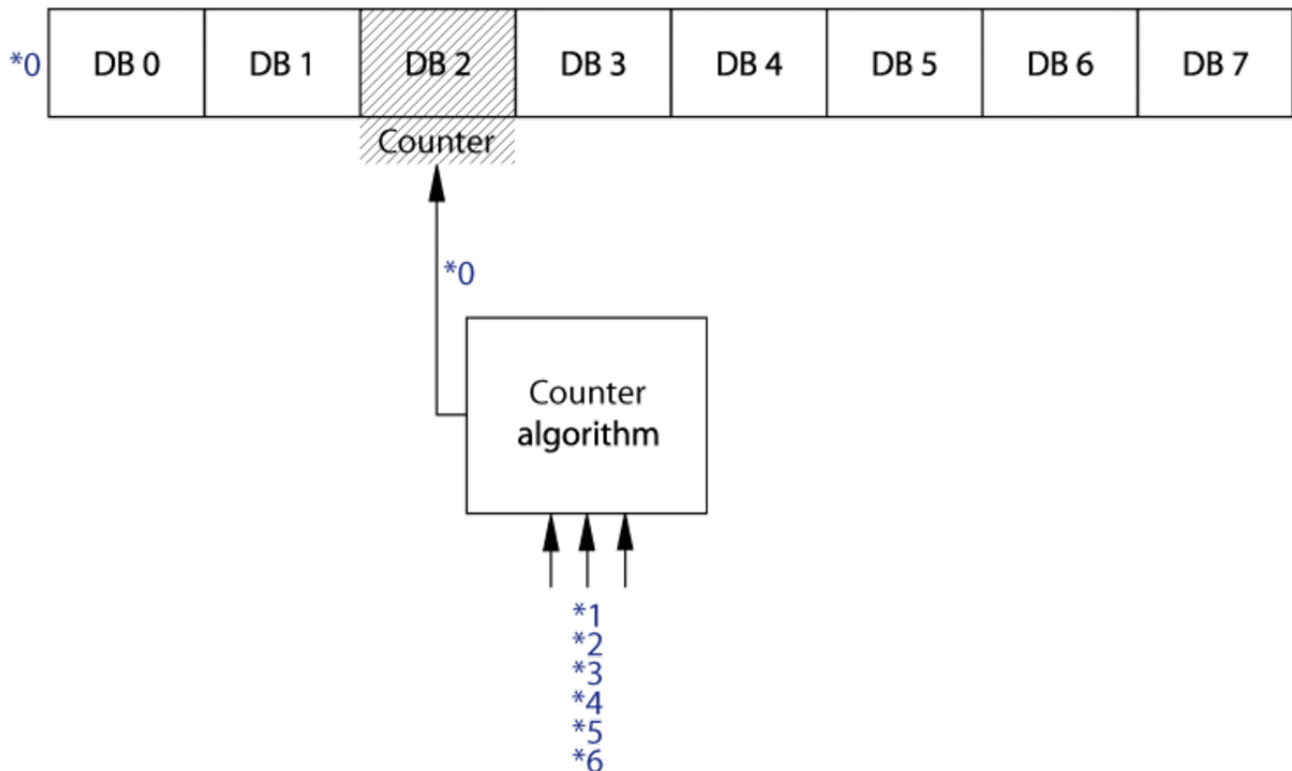
The following table explains the properties of the algorithm:

Index	Name	Description	
0	Target signal	The signal into which the new value is written.	
1	Mode	Mode	Description
		Periodic	The value is calculated new periodically with a period of <code>Time base (ms)</code> .
		Single shot	The value is calculated new only once after a time of <code>Time base (ms)</code> .
2	Minimum	This is the minimum value, that will be assigned, if the new value was lesser.	
3	Maximum	This is the maximum value, that will be assigned, if the new value was greater	
4	Increment/Decrement	This is the value, that will be added or subtracted from the last value.	
5	Time base (ms)	The time base defines the interval, in which a new value is calculated.	
6	Byteswap		

10.10.3Counter - Frame based

This signalfunction realizes a counter, that is calculated right before the frame is sent. This allows to easily implement message counters within frames.

The following diagram shows the properties of the algorithm:



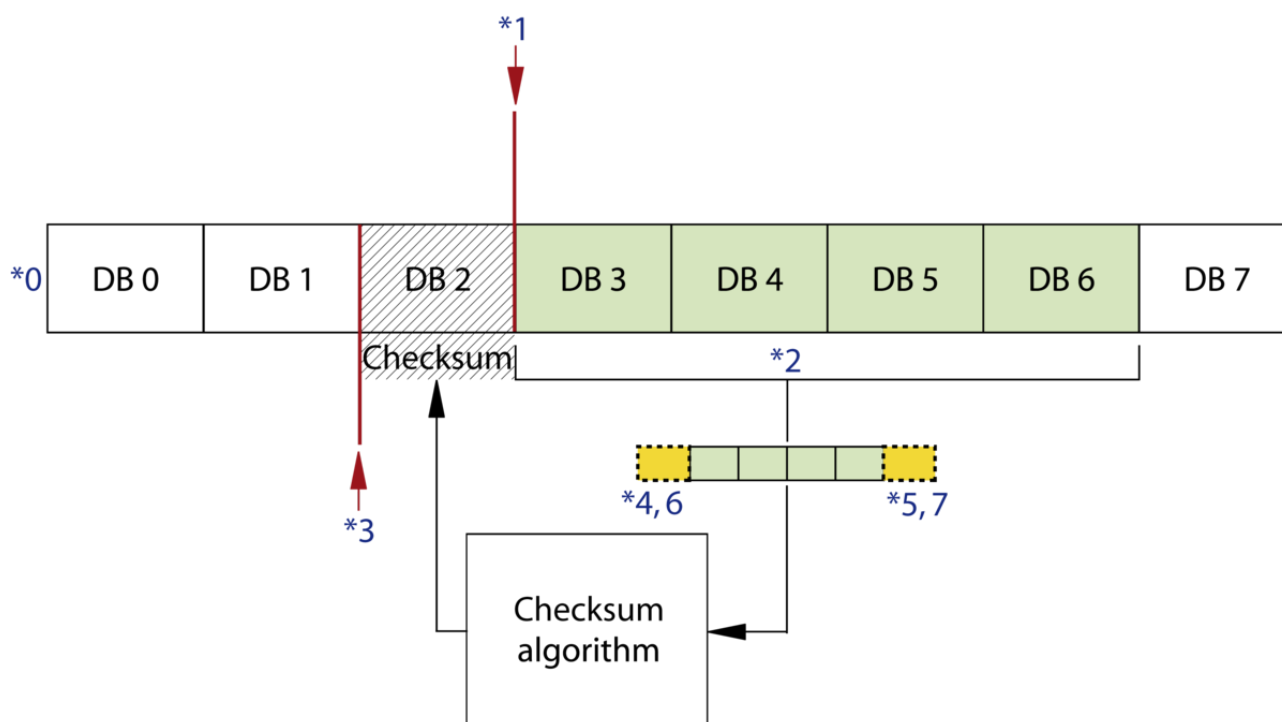
The following table explains the properties of the algorithm:

Index	Name	Description	
0	Target signal	The signal into which the new value is written.	
1	Mode	Mode	Description
		Periodic	The value is calculated new periodically with a period of <code>Time base (ms)</code> .
		Single shot	The value is calculated new only once after a time of <code>Time base (ms)</code> .
2	Minimum	This is the minimum value, that will be assigned, if the new value was lesser.	
3	Maximum	This is the maximum value, that will be assigned, if the new value was greater	
4	Increment/Decrement	This is the value, that will be added or subtracted from the last value.	
5	Byteswap		

10.10.4Checksum

This signalfunction realizes a 8 Bit Modulo checksum algorithm, that is calculated right before the frame is sent.

The following diagram shows the properties of the algorithm:



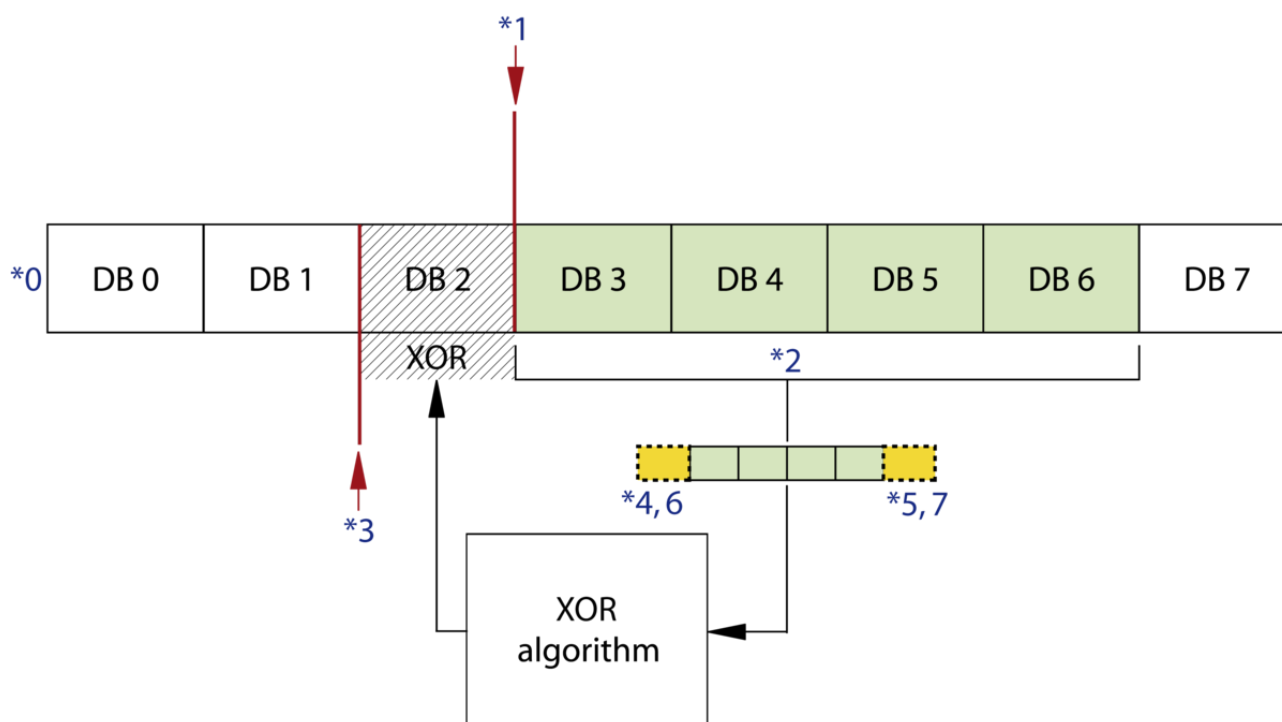
The following table explains the properties of the algorithm:

Index	Name	Description
0	Frame	This is the frame into which the calculated value will be mapped.
1	Start byte of input block within the frame	This is the first byte of the input block area, that will be used to calculate the value.
2	Byte length of the input block within the frame	This is the length in bytes of the input block area, that will be used to calculate the value.
3	Start byte of value within the frame	This is the byte position of the target value.

10.10.5XOR

This signalfunction realizes a 8 Bit XOR checksum algorithm, that is calculated right before the frame is sent.

The following diagram shows the properties of the algorithm:



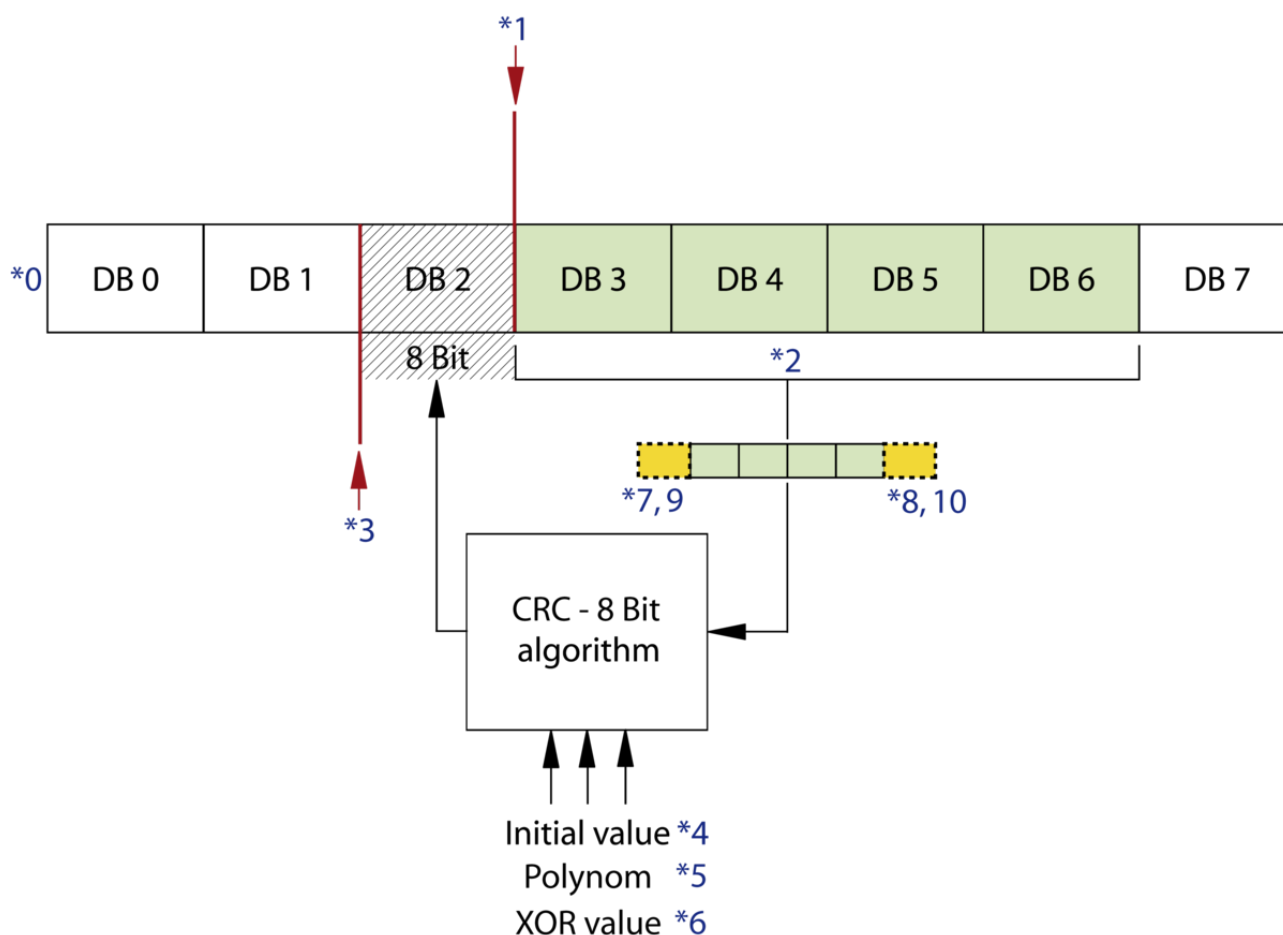
The following table explains the properties of the algorithm:

Index	Name	Description
0	Frame	This is the frame into which the calculated value will be mapped.
1	Start byte of input block within the frame	This is the first byte of the input block area, that will be used to calculate the value.
2	Byte length of the input block within the frame	This is the length in bytes of the input block area, that will be used to calculate the value.
3	Start byte of value within the frame	This is the byte position of the target value.

10.10.6CRC - 8 Bit

This signalfunction realizes a 8 Bit CRC algorithm, that is calculated right before the frame is sent.

The following diagram shows the properties of the algorithm:



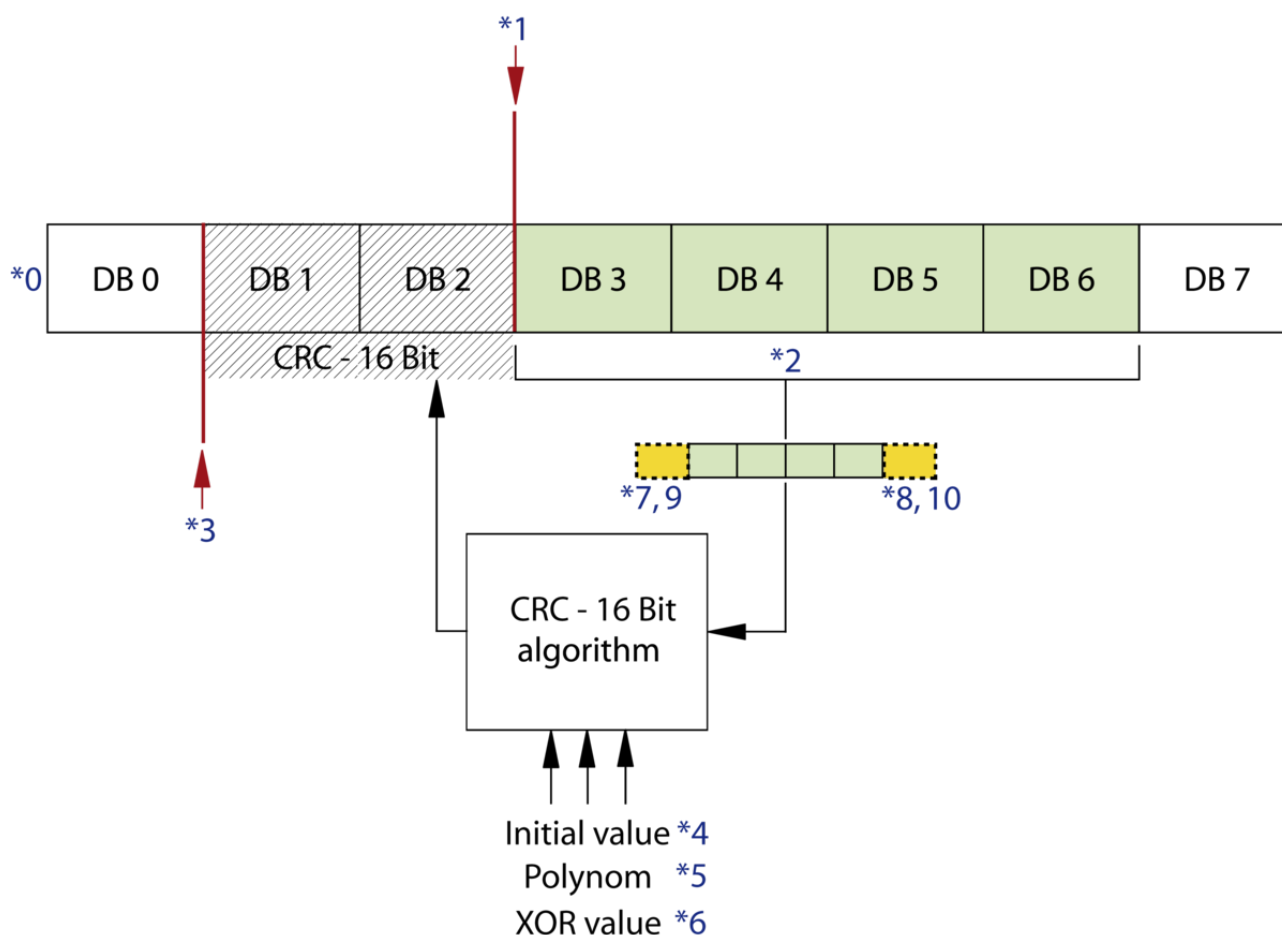
The following table explains the properties of the algorithm:

Index	Name	Description
0	Frame	This is the frame into which the calculated value will be mapped.
1	Start byte of input block within the frame	This is the first byte of the input block area, that will be used to calculate the value.
2	Byte length of the input block within the frame	This is the length in bytes of the input block area, that will be used to calculate the value.
3	Start byte of value within the frame	This is the byte position of the target value.
4	Initial value	This is the initial value used by the CRC algorithm.
5	Polynom	This is the polynom used by the CRC algorithm.

10.10.7CRC - 16 Bit

This signalfunction realizes a 16 Bit CRC algorithm, that is calculated right before the frame is sent.

The following diagram shows the properties of the algorithm:



The following table explains the properties of the algorithm:

Index	Name	Description
0	Frame	This is the frame into which the calculated value will be mapped.
1	Start byte of input block within the frame	This is the first byte of the input block area, that will be used to calculate the value.
2	Byte length of the input block within the frame	This is the length in bytes of the input block area, that will be used to calculate the value.
3	Start byte of value within the frame	This is the byte position of the first byte of the target value.
4	Initial value	This is the initial value used by the CRC algorithm.
5	Polynom	This is the polynom used by the CRC algorithm.

10.11 Protocols

By using the protocol feature it is possible to define custom protocols as well as use standardized protocols such as:

- LIN diagnostics
- ISO-TP
- UDS

These protocols are then executed directly by the Baby-LIN without the need for a PC or PLC. This is especially useful for time critical applications.

10.12 GUI-Elements

Here you can add elements, that automatically will be shown, when the SDFFile is loaded with the "SimpleMenu".

The following elements can be added:

Element	Description		
Signals	Signals can be shown in two kind of ways:		
		Presentation	Selectable signals
	Monitored signal	The value of the signal will be shown.	All signals can be monitored.
	Edit signals	The value of the signal will be shown and can be edited.	Only emulated bus signals and virtual signals can be edited.
Macros	A macro can be started by the user. The state and return value will be displayed.		
Macroselections	One of the macros of the macroselection can be choosen and started.		

For each element an alternative name can be defined, that will be shown.

10.13 Macros

10.13.1 Overview

A macro consists of one or more macro commands. Each of these macro commands are single instructions, that allow you to modify signals, bus states, the macro flow and much more. Each macro command has its own parameters.



Version incompatitbility

Not all macro commands are available in SDF-V2, but only in SDF-V3.

10.13.2 Command labels

Each macro command can get a label. This label is just a textual marker, but is very helpful to mark certain sections within a macro. A label can also be used to clarify the target command of a jump command. Check chapter "Flow control commands" for more information.

10.13.3 Macro commands

10.13.3. Signal commands

Command	Description	Parameter							
Set signal	Set a signal with a constant value. Requirements: <ul style="list-style-type: none">Target signal must be emulated.	Signal The signal, that is set. Value The constant value, that is assigned to the signal.							
Add signal	Add a constant value to a signal. Requirements: <ul style="list-style-type: none">Target signal must be emulated.	Signal The signal, that is set. Value The constant value, that is added to the signal. Minimum The minimum value, the signal can be assigned to. Maximum The maximum value, the signal can be assigned to. Wrap around The maximum value, the signal can be assigned to. The maximum value, the signal can be assigned to. The maximum value, the signal can be assigned to. <table><tr><th>Wrap around</th><th>Description</th></tr><tr><td>enabled</td><td>If the result is lesser than the minimum/greater than the maximum, the rest will be added to the maximum/minimum value.</td></tr><tr><td>disabled</td><td>If the result is greater than this maximum, the maximum will be assigned to the signal.</td></tr></table>		Wrap around	Description	enabled	If the result is lesser than the minimum/greater than the maximum, the rest will be added to the maximum/minimum value.	disabled	If the result is greater than this maximum, the maximum will be assigned to the signal.
Wrap around	Description								
enabled	If the result is lesser than the minimum/greater than the maximum, the rest will be added to the maximum/minimum value.								
disabled	If the result is greater than this maximum, the maximum will be assigned to the signal.								
Set from signal	Set a signal with the value of another signal. Requirements: <ul style="list-style-type: none">Target signal must be emulated.SDF-V3	Signal target The signal, that is set. Signal source This signal value will be assigned to the target signal.							
Set bit	Set or clear a single bit within a signal. Requirements: <ul style="list-style-type: none">Target signal must be emulated.SDF-V3	Signal target The signal, that is set. Bit position [0..n) The position of the bit, starting with the LSB. Set bit Decides if a bit is set or reset.							

Command	Description	Parameter
Set minimum	Set a signal to its minimum value. Requirements: <ul style="list-style-type: none"> Target signal must be emulated. SDF-V3 	Signal target The signal, that is set.
Set maximum	Set a signal to its maximum value. Requirements: <ul style="list-style-type: none"> Target signal must be emulated. SDF-V3 	Signal target The signal, that is set.
Set using mathematical operation	Set a signal by applying a mathematical operation to a signal value and a constant or signal. Requirements: <ul style="list-style-type: none"> Target signal must be emulated. SDF-V3 	Set signal The signal, that is set. Signal The signal, that is used as first operand. Operation The possible operations are shown in the following table. Signal The signal, that is used as second operand, if the second operand is a signal. Constant The signal, that is used as second operand, if the second operand is a constant.

There are several mathematical operations available. The following table gives an overview:

Mathematical operation	Description
+ const	Adds a constant to a signal.
+ signal	Adds a signal to a signal.
- const	Subtracts a constant from a signal.
- signal	Subtracts a signal from a signal.
* const	Multiply a constant with a signal.
* signal	Multiply a signal with a signal.
/ const	Divides a signal by a constant.
/ signal	Divides a signal by a signal.
& const	Applies a logical AND to a signal and a constant.
& signal	Applies a logical AND to a signal and a signal.
const	Applies a logical OR to a signal and a constant.
signal	Applies a logical OR to a signal and a signal.
^ const	Applies a logical XOR to a signal and a constant.
^ signal	Applies a logical XOR to a signal and a signal.
<< const	Left shifts a signal by a constant.
<< signal	Left shifts a signal by a signal.

Mathematical operation	Description
>> const	Right shifts a signal by a constant.
>> signal	Right shifts a signal by a signal.
Set array value using signal index	Sets an array value to a signal value using the index from a constant.
Set array value using constant index	Sets an array value to a signal value using the index from a signal.
Get array value using signal index	Gets a signal value from a signal array using the index from a constant.
Get array value using constant index	Gets a signal value from a signal array using the index from a signal.

10.13.3. Bus commands

Command	Description	Parameter
Start	Starts the bus. If the Baby-LIN emulates a LIN master, a schedule can be chosen, that should be executed.	Start previously selected schedule (LIN-Bus only) The bus is started with the last executed schedule. Start schedule (LIN-Bus only) The bus is started with the selected schedule. Schedule The schedule, that should be executed.
Stop	Stops the bus.	
Restart	Stops the bus and start it again without resetting any signal.	
Sleep	Initiate a sleep event. Requirements: • Section must be a LIN section	
Wakeup	Initiate a wakeup event. Requirements: • Section must be a LIN section	Automatic timing The period of the wakeup pulse is 150 µs. Manual timing This defines the period of the wakeup pulse.
Set speed	Set the baudrate of the bus.	Speed The new baudrate.

10.13.3.3 LIN commands

Command	Description	Parameter
Select schedule	<p>Change the schedule, that should be executed by a LIN master. Optionally you can set the schedule mode for this execution.</p> <p>Requirements:</p> <ul style="list-style-type: none"> LIN section Baby-LIN simulates Master 	<p>Schedule</p> <p>The schedule, that should be executed.</p> <p>Mode</p> <ul style="list-style-type: none"> <code>keep schedule mode</code>: The schedule will be started with the last set mode. <code>Cyclic</code>: The schedule is executed cyclic until another one is started. <code>Single run</code>: The schedule will run only once. <code>Exit on complete</code>: The schedule is executed cyclic until another one is started. But the schedule will finish its cycle before.
Force checksum	<p>Force a checksum type.</p> <p>Requirements:</p> <ul style="list-style-type: none"> LIN section 	<p>Checksum type</p> <p>Forces the LIN checksum generator to use a specific model independant from SDF information.</p> <ul style="list-style-type: none"> <code>automatic</code>: The checksum type from the SDF will be used. <code>v1</code>: All frames will use the LIN V1 checksum type. <code>v2</code>: All frames will use the LIN V2 checksum type.
Send Master Request	<p>Send a Master Request (Frame ID 3C) on the LIN bus.</p> <p>Requirements:</p> <ul style="list-style-type: none"> LIN section Baby-LIN simulates Master 	<p>Master request</p> <p>The data bytes of the MasterRequest.</p> <p>Response count</p> <p>The number of SlaveResponses expected.</p>
Send DTL Request	<p>Disabled command. It is only for compatibility reasons visible.</p> <p>Requirements:</p> <ul style="list-style-type: none"> LIN section Baby-LIN simulates Master 	

10.13.3.4 Flow control commands

Command	Description	Parameter
Delay	Delays the execution for the specified time.	<p>Delay</p> <p>The delay time in milliseconds.</p>
Jump	Jump to another command within the macro. Can be used to realize loops or larger conditional command blocks. Often used with a condition.	<p>Target command</p> <p>The target of the jump. It is helpful, if "Command labels" are used.</p>

Command	Description	Parameter
Event	<p>Disable or enable events.</p> <p>Requirements:</p> <ul style="list-style-type: none"> An event must exist 	<p>Mode</p> <ul style="list-style-type: none"> Disable: Disable a certain event. Enable: Enable a certain event. Disable all events: Disable all events. Enable all events: Enable all events. Enable grouping: This mode starts the grouping. All following event commands will not be executed, until the grouping is disabled. Disable grouping: This mode stops the grouping. All event commands, that were queued after the grouping began, will now be executed. <p>Event selection</p> <p>The selected device specific event.</p>

10.14 Macroselections

A macroselection is a list of macros. They can be used to group macros, that belong together.

A macroselection always has a macro, that is selected. By default the first macro of a macroselection is selected.

Using macros or "Macro actions" it is possible to execute a macro relative to the currently selected macro and to change the selection. Hereby all macros of a macroselection can be executed one after another.

10.15 Events

10.15.1 Introduction

Events can be used to react to certain situations on the LIN-Bus. An event consists of two parts:

- Event condition: This condition determines if and when the event is triggered.
- One or more event actions: The event actions will be executed, as soon as the event condition is .

The events are categorized by the Baby-LIN product. An event can only be triggered, if the SDFile is loaded into a matching Baby-LIN product. The following Baby-LIN product categories exist:

Baby-LIN product category	Baby-LIN products, on which events are triggered
BabyLIN/-II/-MB/-MB-II	<ul style="list-style-type: none"> Baby-LIN Baby-LIN-II Baby-LIN-MB Baby-LIN-MB-II
BabyLIN-RC/-II	<ul style="list-style-type: none"> Baby-LIN-RC Baby-LIN-RC-II

Baby-LIN product category	Baby-LIN products, on which events are triggered
BabyLIN-RM/-II	<ul style="list-style-type: none"> • Baby-LIN-RM • Baby-LIN-RM-II
HARP-2/4/5	<ul style="list-style-type: none"> • HARP-4 • HARP-5

10.15.2 Event conditions

An event has a event condition. The event is triggered, as soon as the condition is fulfilled. The availability of different event conditions depends on the device. All events are grouped in event types.

10.15.2.1 Frame event conditions

Condition	Trigger description	Parameter
Frame on BUS	The selected frame appears on the bus.	<p>Trigger on all known Frames</p> <p>If checked, the event is triggered, if a known frame is received. If unchecked, a frame must be selected.</p> <p>Frame</p> <p>The event is triggered, if this frame is received.</p>
Frame error	The selected frame caused an error.	<p>Abort all following frame events</p> <p>If this event is triggered and this flag is checked, all following frame events will not be triggered, even, if they have selected the same frame.</p> <p>Trigger on all known Frames</p> <p>If checked, the event is triggered, if a known frame is received. If unchecked, a frame must be selected.</p> <p>Frame</p> <p>The event is triggered, if this frame is received.</p>
Frame okay	The selected frame appeared on the bus without any errors.	<p>Trigger on all known Frames</p> <p>If checked, the event is triggered, if a known frame is received. If unchecked, a frame must be selected.</p> <p>Frame</p> <p>The event is triggered, if this frame is received.</p>
Changes	The selected frame appears on the bus and its data have changed.	<p>Trigger on all known Frames</p> <p>If checked, the event is triggered, if a known frame is received. If unchecked, a frame must be selected.</p> <p>Frame</p> <p>The event is triggered, if this frame is received.</p>

10.15.2. Signal event conditions

Condition	Trigger description	Parameter
< Const	The signal value is lesser than the given constant.	<p>Signal</p> <p>This signal value will be compared to a constant.</p> <p>Compare value</p> <p>The constant value, which is compared to the signal value.</p>
<= Const	The signal value is lesser than or equal to the given constant.	
> Const	The signal value is greater than the given constant.	
>= Const	The signal value is greater than or equal to the given constant.	
= Const	The signal value is equal to the given constant.	
!= Const	The signal value is not equal to the given constant.	
< Signal	The signal value is lesser than the compare signal value.	<p>Signal</p> <p>This signal value will be compared to another signal value.</p> <p>This signal value stands on the left side of the relation.</p> <p>Compare to signal</p> <p>This signal value will be compared to another signal value.</p> <p>This signal value stands on the right side of the relation.</p>
<= Signal	The signal value is lesser than or equal to the compare signal value.	
> Signal	The signal value is greater than the compare signal value.	
>= Signal	The signal value is greater than or equal to the compare signal value.	
= Signal	The signal value is equal to the compare signal value.	
!= Signal	The signal value is not equal to the compare signal value.	
In range	The signal value is within the given range.	<p>Signal</p> <p>This signal value is checked, if it is within the given range.</p> <p>Minimum</p> <p>This is the minimum value of the range.</p> <p>Maximum</p> <p>This is the maximum value of the range.</p>
Out range	The signal value is outside the given range.	
Bitmask	The signal equals a constant after a bitmask was applied via a logical AND operation.	<p>Signal</p> <p>The signal value, that will be compared to the compare value, after the bitmask was applied.</p> <p>Compare value</p> <p>A constant.</p> <p>Bitmask</p> <p>The bitmask will be applied via a logical AND operation.</p>
Changes	The signal value changed.	<p>Signal</p> <p>The signal to check.</p>

10.15.3 Event action

An event has one or more actions, that are executed, if the event is triggered.

**Attention**

It is highly recommended, to implement the application logic within macros.

Therefor an event should have only one action, which starts a macro.

10.15.3. Signal actions

Action	Description	Parameter						
Set signal	Sets the signal value from a constant value.	Signal The signal, that is set. Value The constant value, that is assigned to the signal.						
Add signal	Adds a constant value to a signal value.	Signal The signal, that is set. Value The constant value, that is added to the signal. Minimum The minimum value, the signal can be assigned to. Maximum The maximum value, the signal can be assigned to. Wrap around The wrap around option defines, what happens, if the minimum or maximum is reached: <table><tr><th>Wrap around</th><th>Description</th></tr><tr><td>enabled</td><td>If the result is lesser than the minimum/greater than the maximum, the rest will be added to the maximum/minimum value.</td></tr><tr><td>disabled</td><td>If the result is greater than this maximum, the maximum will be assigned to the signal.</td></tr></table> If the result is greater than this maximum, the maximum will be assigned to the signal.	Wrap around	Description	enabled	If the result is lesser than the minimum/greater than the maximum, the rest will be added to the maximum/minimum value.	disabled	If the result is greater than this maximum, the maximum will be assigned to the signal.
Wrap around	Description							
enabled	If the result is lesser than the minimum/greater than the maximum, the rest will be added to the maximum/minimum value.							
disabled	If the result is greater than this maximum, the maximum will be assigned to the signal.							
Set from signal	Sets the signal value from another signal.	Signal target The signal, that is set. Signal source This signal value will be assigned to the target signal.						

Action	Description	Parameter
Set bit	Sets or resets a single bit of a signal value.	<p>Signal</p> <p>The signal, that is set.</p> <p>Bit position [0..n)</p> <p>The position of the bit, starting with the LSB.</p> <p>Set bit</p> <p>Decides if a bit is set or reset.</p>
Set minimum	Sets the signal value to the minimum value of the signal.	<p>Signal</p> <p>The signal, that is set.</p>
Set maximum	Sets the signal value to the maximum value of the signal.	<p>Signal</p> <p>The signal, that is set.</p>



Warning

To add a signal action, at least one bus signal or virtual signal must exist in the SDFFile.

10.15.3.2 Macro actions

Action	Description	Parameter
Start	Start a macro.	<p>Macro</p> <p>The macro, that is started.</p>
Stop	Stops all macros or just the selected macro.	<p>Stop all macros</p> <p>If checked, all running macros are stopped.</p> <p>Macro</p> <p>The macro, that is stopped, if not all macros should be stopped.</p>
Macro-selection	Changes the selected macro and/or executes the selected macro.	<p>Selection</p> <p>The macroselection, that is processed.</p> <p>Mode</p> <p>The mode is described in the table below.</p>



Warning

To add a macro action, at least one macro must exist in the SDFFile.



Warning

To add a macroselection action, at least one macroselection must exist in the SDFFile.

The macroselection action can have a number of different parameter sets depending on the mode parameter:

Mode	Description
Select and execute	Selects a macro from the macroselection and executes it. Parameter: <ul style="list-style-type: none"> Macro: The macro that should be selected and executed.
Select next and execute it	Selects the next macro and executes it. Parameter: <ul style="list-style-type: none"> no wrap around and re-execute limits: If the selected macro already was the last macro, it will be selected again and executed again. wrap around: If the selected macro already was the last macro, the first macro will be selected and executed. no wrap around and execute limits only once: If the selected macro already was the last macro, it will be selected again, but not executed again.
Select previous and execute it	Selects the previous macro and executes it. Parameter: <ul style="list-style-type: none"> no wrap around and re-execute limits: If the selected macro already was the first macro, it will be selected again and executed again. wrap around: If the selected macro already was the first macro, the last macro will be selected and executed. no wrap around and execute limits only once: If the selected macro already was the first macro, it will be selected again, but not executed again.
Select first and execute it	Selects the first macro from the macroselection and executes it.
Select last and execute it	Selects the last macro from the macroselection and executes it.
Select without executing	Selects a macro from the macroselection, but does not execute it. Parameter: <ul style="list-style-type: none"> Macro: The macro that should be selected..
Select next without executing	Selects the next macro, but does not execute it. Parameter: <ul style="list-style-type: none"> no wrap around: If the selected macro already was the last macro, it will be selected again. wrap around: If the selected macro already was the last macro, the first macro will be selected.
Select previous without executing	Selects the previous macro, but does not execute it. Parameter: <ul style="list-style-type: none"> no wrap around: If the selected macro already was the first macro, it will be selected again. wrap around: If the selected macro already was the first macro, the last macro will be selected.

11 LINWorks Software

11.1 LINWorks archive

The LINWorks is a collection of software to operate the Baby-LIN. The LINWorks archive is the downloadable file, that contains the following components:

Component	Archive subfolder	Description
Datasheets User manuals Application notes	Documentation	The datasheets show a quick overview about a Baby-LIN product and its features. The user manuals contain the main documentation. The application notes contain some older information, that have not been added to the user manuals yet.
"Baby-LIN driver"		The Baby-LIN driver is necessary to connect a Baby-LIN to a windows PC via USB. The Baby-LIN will be available as virtual COM port .
"LDFEdit"	LINWorks	The LDFEdit allows the inspection, creation and edit of a LDFFile (LIN Description File).
"SessionConf"	LINWorks	The SessionConf allows the inspection, creation and edit of a SDFFile (Session Description File) and features a file import for LDFFiles (for LIN-Bus simulation). It defines everything needed for a complete simulation of each available bus, e.g. which nodes on each bus are available and which nodes should be simulated by the Baby-LIN. Moreover it allows defining an application logic. This programming ability is available for each device out of the box.
"SimpleMenu"	LINWorks	The SimpleMenu is used to establish a connection to the Baby-LIN and upload SDFFiles, change the device target configuration, control the bus and monitor the frames and signals on the bus. Even without a LDFFile/SDFFile the bus can be monitored and the frames can be logged.
"LogViewer"	LINWorks	The LogViewer can show and convert the log files of the SimpleMenu.
"FrameBlaster"	LINWorks	
"CustomPanel"	LINWorks	
"Baby-LIN-DLL"	Development	The Baby-LIN-DLL allows customers to create their own application and use all features of the Baby-LIN like controlling and monitoring the LIN-Bus interfaces. The Baby-LIN-DLL is a native C/C++ DLL. It is available for Windows , Linux and RaspberryPi . Wrapper for .NET , Python , VB6 and LabView are available. Of course we provide examples for all supported languages.

Component	Archive subfolder	Description
"Baby-LIN-DLL .NET Wrapper"	Development	The Baby-LIN-DLL .NET Wrapper allows using all the functionality of the Baby-LIN-DLL within a .NET application . The Baby-LIN-DLL .NET Wrapper therefor offers the same interface as the Baby-LIN-DLL with minimal adjustments of the data types.
"Serial writer"	Tools	The Serial writer is used to change the serial number , that is stored within the persistent memory of a Baby-LIN. This serial number influences the allocation of the virtual COM port number, the Baby-LIN is available under.
"BLProg"	Tools	The BLProg is used to update the firmware of a Baby-LIN. If you download a firmware package from our customer portal (portal.lipowsky.de) a current version of the BLProg will always be included.
"Switch to ASCII tool"	Tools	The Switch to ASCII tool allows switching the Baby-LIN into the ASCII mode. Within this mode ASCII commands can be sent to use the Baby-LIN as mere level-translator or sending characters via the LIN hardware layer without the LIN frame packing conventions.

11.2 Installation

The LINWorks archive contains all components required to configure and operate the Baby-LIN.

Only the Baby-LIN driver needs a real installation with Administrator privileges. The other components like the LINWorks software can be executed without an installation.

For convenience we recommend to use the `Setup.exe` to install all components. This will install the Baby-LIN driver, copy all applications and create start menu entries.

11.3 Baby-LIN driver

The **Baby-LIN driver** is necessary to connect a Baby-LIN to a windows PC via USB. The Baby-LIN will be available as **virtual COM port**.

The Baby-LIN driver will provide the Baby-LIN as virtual COM port. Like a regular COM port the Baby-LIN will be available under a name like "COM1". This name will be used when you establish a connection using the "SimpleMenu" or the "Baby-LIN-DLL".

All required installation files will be installed by a setup file. Therefor it will check your windows version and install the correct version for your operating system.

**Advice**

The driver version for Windows XP will be v6.7.

The driver version for later operating systems will be higher.

11.4 LDFEdit

The LDFEdit is an editor for LDFiles. It can be used to inspect, create and edit LDFiles, which contains the full description of a LIN-Bus. That includes e.g. the nodes, frames, signals and schedules as well as their relation.

The LDFEdit also features background checks, that prevent you from saving invalid or incompatible LDFiles.

11.5 SessionConf

The SessionConf is an editor for SDFiles. It can be used to inspect, create and edit SDFiles, which contains the configuration to simulate any bus node.

To create a SDFile the SessionConf allows importing LDFiles and DBC files. The configuration and features of SDFiles are described in chapter ["SDF - The session configuration file"](#).

The SessionConf also features background checks and compatibility helper, that prevent you from saving invalid or incompatible SDFiles.

If you want to program custom applications, the SessionConf can export header files for some programming languages. Those headers contain constants named as the item name and the SDFile index. The following item indexes will be exported:

- | | | |
|-------------|------------|-----------|
| • Nodes | • Frames | • Signals |
| • Schedules | • Sections | • Macros |

11.6 SimpleMenu

The SimpleMenu gives you an easy access to your Baby-LIN and lets you control it. It is basically a comfortable and user friendly GUI for the Baby-LIN-DLL.

The SimpleMenu gives you access to the following features:

- Manage all connected Baby-LIN products.
- Manage all channels of a Baby-LIN product.
- Load SDFiles and control the bus.
- Operate the Baby-LIN product without a SDFile in the monitor mode to watch the bus communication.
- Watch and set signals as well as start macros using a customizable graphical user interface.
- Watch the bus communication in a frame monitor.
- Set the target configuration.
- View and add activation codes.

- Send commands directly to any Baby-LIN product.
- And many more...

11.7 LogViewer

The LogViewer allows viewing log data and to export signal data. It can open log files in the "Binary format", which contain frame based data.

To access the signal values, the LogViewer requires the SDFFile, on which the communication was based. Thus it is possible to filter and show relevant data by selecting a signal and a time grid. These data can also be exported as CSV file.

The following table shows you a typical operating sequence:

Step	User interaction	Output
Load log data	<ul style="list-style-type: none"> • Select <code>File</code> and then <code>Open Log</code> from the menu. • Now select a log file in the "Binary format". 	<ul style="list-style-type: none"> • The <code>Logfile</code> section in the <code>Log Info</code> tab shows you a summary of the log file. • The <code>Raw log framedata</code> tab shows you the raw frame based log data.
Load SDFFile	<ul style="list-style-type: none"> • Select <code>File</code> and then <code>Open SDF</code> from the menu. • Now select a SDFFile. 	<ul style="list-style-type: none"> • The <code>SDFFile</code> section in the <code>Log Info</code> tab shows you a summary of the SD-File. • The <code>Reports</code> menu is now available.
Create a signal report	<ul style="list-style-type: none"> • Select <code>Reports</code> and then <code>New Signal report</code> from the menu. • Select an option: <ul style="list-style-type: none"> ◦ List selected signals on each frame occurrence ◦ List values of selected signals with an interval 	<ul style="list-style-type: none"> • A <code>New Report</code> tab will open.
Configure the signal report	<ul style="list-style-type: none"> • Select a <code>New Report</code> tab. • Select the signals on the left side, that you are interested in. • Only for interval based reports: Edit the interval. • Click the update button on the left bottom. 	<ul style="list-style-type: none"> • The signal values and changes are shown on the right side.
Export the signal report	<ul style="list-style-type: none"> • Select <code>Reports</code> and then <code>Export current report (CSV)</code> from the menu. • Now enter the target CSV file. 	<ul style="list-style-type: none"> • The signal report will be saved as CSV file.

11.8 FrameBlaster

11.8.1 Introduction

The **FrameBlaster** is a script interpreter, that gives you access to the features of the Baby-LIN from within a JavaScript. The Script can be developed and executed in an integrated development environment, but also executed by a command line tool to allow batch execution.

11.8.2 Features

Typical applications for the FrameBlaster are:

- Monitor a LIN-bus, filter the communication and log in a standard or custom format.
- Send and receive Frames without a LDFile or SDFile and execute custom logics.
- Record and replay bus communication.
- Emulate a master node

Input data can come from different sources:

- From the script code itself.
- Loaded from custom files.
- Imported from log data.
- More sources are planned in the future.

Received data can be filtered and logged:

- Filter directly in the script code using the powerful JavaScript string processing.
- Log in standard formats or custom binary or text formats.

11.9 CustomPanel

11.9.1 Introduction

The **CustomPanel** is a graphical user interface, whose layout is stored in a configuration file. The controls allow you to show and control LIN based signals from a SDFile. With this tool any customer can quickly create complex user interfaces based on your requirements. A graphical editor to create the configuration of the graphical user interface is included.

11.9.2 Features

Typical applications for the CustomPanel are:

- Show a user-defined graphical user interface based on a SDFile.
- Create the layout for a graphical user interface based on a SDFile.

Individual designs can be realized with hierarchical graphical elements:

- Tab pages. They can be used to differentiate between different operating modes and realize entire different views.
- Group boxes. They can be used to group different controls.
- Controls. They can be pure graphical elements, show and edit signal values and trigger macros.

There is a wide variety of controls available:

- Controls to show and edit signal values, like text boxes, sliders, check boxes, radio buttons, combo boxes, spin boxes and knobs.
- It is also possible to show images depending on the value of a signal, e.g. to visualize error states or show small animations.
- A control to start a macro and/or show its state.
- Graphical elements like labels or generic buttons.

There are many configurations possibilities, e.g.:

- Signal based controls can be configured to be read-only, write-only or both readable and writable.
- All controls can be shown or hidden depending on a signal value.
- Timeouts can be configured for all controls showing bus signals.

The CustomPanel also features a customizable logging system:

- Multiple log files can be created.
- Each log file can have different signal based states.
- In each of these states custom texts and signal values can be logged.

11.10 Baby-LIN-DLL

The Baby-LIN-DLL is a native dynamic link library written in C/C++, that allows to control the Baby-LIN from a custom application.

The Baby-LIN-DLL can be used in any native customer application, primarily C/C++ applications. Wrapper for .NET applications and LabView are available as well. Check chapter "[Baby-LIN-DLL .NET Wrapper](#)" and "[Baby-LIN-DLL LabView Wrapper](#)" for more information.

The Baby-LIN-DLL allows to rebuild all the features of the SimpleMenu and much more:

- Manage all connected Baby-LIN products.
- Manage all channels of a Baby-LIN product.
- Load SDFiles and control the bus.
- Operate the Baby-LIN product without a SDFile in the monitor modus to watch the bus communication.
- Watch and set signals.
- Start macros and evaluate the result.
- Receive the full bus communication.
- Send commands directly to any Baby-LIN product.
- Inject frames without any LDFile or SDFile.

- Use higher protocols like DTL and others.
- And many more...

The Baby-LIN-DLL is available as x86 and x64 Windows DLL. A Linux version is available upon request.

11.11 Baby-LIN-DLL .NET Wrapper

The Baby-LIN-DLL .NET Wrapper is a .NET library, that allows easy access to the functionality of the Baby-LIN-DLL. It keeps the API of the Baby-LIN-DLL with just minor language dependent adjustments.

11.12 Baby-LIN-DLL LabView Wrapper

The Baby-LIN-DLL LabView Wrapper allows you accessing the Baby-LIN-DLL from within LabView.

11.13 Serial writer

The **Serial writer** is used to change the **serial number**, that is stored within the persistent memory of a Baby-LIN. This serial number influences the allocation of the **virtual COM port** number, the Baby-LIN is available under.



Advice

The serial number will never prevent the usage of multiple Baby-LINs on a single PC. It will just influence, which virtual COM ports are allocated.

The following example shows how the virtual COM ports are allocated if Baby-LINs use the same and different serial numbers. Please note, that the virtual COM port numbers are just exemplary and will differ on each PC.

	Baby-LIN A	Baby-LIN B	Baby-LIN C	Baby-LIN D
Serial number	0001	0001	0002	0003
Virtual COM port, if only A is connected	COM1			
Virtual COM port, if only B is connected		COM1		
Virtual COM port, if only C is connected			COM10	
Virtual COM port, if only D is connected				COM15
Virtual COM ports, if the Baby-LINs are connected in the following order: A, B, C, D	COM1	COM2	COM10	COM15
Virtual COM ports, if the Baby-LINs are connected in the following order: D, C, B, A	COM2	COM1	COM10	COM15

Using the tool is pretty straight forward. When it is started it will show you the current serial number and allow you to program a new one. All you have to do, is to make sure, only one Baby-LIN product is connected to your PC.

**Warning**

The Serial writer will only let you change the serial number if exactly one Baby-LIN product is connected to your PC. If none or at least two are found, the Serial writer will show you an error message.

**Advice**

The new virtual COM port will be allocated after the Baby-LIN has been disconnected from the USB port and then reconnected.

If you execute the procedure for all of your Baby-LIN, each one will always receive its individual virtual COM port.

11.14 BLProg

The BLProg is a small command line tool, that lets you update the firmware of the Baby-LIN. It is always included if you download a firmware package.

11.15 Switch to ASCII tool

The Switch to ASCII tool is a small command line tool, that lets you switch the Baby-LIN into the ASCII mode. Check chapter "[ASCII mode](#)" for more information.

12 Protocols and applications

12.1 Jumbo Frames: LIN frames with more than 8 data bytes

12.1.1 Introduction

The LIN specification defines a maximum number of 8 data bytes. Certain applications require or benefit from the usage of frames with more data bytes. Therefore the Baby-LIN can be enabled to process LIN-Bus frames with more than 8 data bytes. These frames will be referred to as jumbo frames.



Advice

The jumbo frame support needs to be activated with a special voucher code.

Please ask for the optional voucher code "Option-BL-HARP-Jumbo Frames" (Item number: 8000831).

(*) The following products support jumbo frames: Baby-LIN-II, Baby-LIN-RC-II, Baby-LIN-RM-II, Baby-LIN-MB-II, HARP-4, HARP-5

12.1.2 Jumbo frames via inject commands

Jumbo frames are LIN-Bus frames with more than 8 data bytes. The maximum number of data bytes supported by the Baby-LIN is 1024 Bytes.

Jumbo frames can be sent by the Baby-LIN as master by using so called inject commands. They allow the user to freely assemble a custom frame and send it on the LIN-Bus. Inject frames can even be sent without being defined in a schedule. The inject frame will be sent at the next possible moment in time, which usually is at the end of the slot time of the current scheduled frame.



Attention

Some LIN-Bus nodes may have problems if a schedule table is interrupted and an inject frame is inserted. The Baby-LIN has the ability to run a schedule table just once instead of in a loop. Therefore it is possible to finish a schedule table before injecting a frame.

Jumbo frames can be injected in custom applications using the Baby-LIN-DII. It is also possible to inject jumbo frames using the SimpleMenu for testing purposes.



Version incompatibility

Right now it is not possible to use jumbo frames in SDFs and therefore they can not be used in stand-alone applications.

Using inject commands custom frames can be assembled. To send a jumbo frame the following information are required:

- The checksum type (Classic, extended, undefined)
- The frame ID

- The data bytes itself
- The slot time for the inject frame

To receive a jumbo frame the following information are required:

- The frame ID
- The number of the data bytes
- The slot time for the inject frame

Up to 8 inject commands can be issued to be send or received consecutive.

12.1.3 Testing jumbo frames using the SimpleMenu

The most simple way to test the sending or receiving of a jumbo frame is by using the SimpleMenu and the so called monitor mode. The monitor mode lets the LIN-Bus simulation device with USB interface simulate a master on the LIN-Bus without the need for a SDF. In this mode the schedule table has to be created manually by injecting frames.

Please follow these steps to send or receive a jumbo frame:

- Prepare the device and PC:
 - Connect the LIN-Bus simulation device with USB interface to a PC.
 - Start the SimpleMenu and connect to the LIN-Bus simulation device with USB interface.
- Optional: Activate or check the jumbo frame option:
 - Right click on the LIN-Bus simulation device with USB interface in the device list.
 - Select "Activation codes".
 - Enter an activation code to enable the jumbo frame support or check that "Jumbo-Frame Support: enabled" is listed.
- Enable the monitor mode as master:
 - Select the LIN channel window/tab.
 - Open the channel shell by choosing "Channelshell ..." from the "Window" menu.
 - Enter `mon_init 20000 1` in the command prompt (20000 is the baud rate and the 1 signals that the LIN-Bus simulation device with USB interface is the master).
- Open the frame monitor. Here you can see all frames on the LIN-Bus.
 - Open the frame monitor by choosing "Frame monitor..." from the "Window" menu.
- Inject a jumbo frame:
 - Enter `inject 1 0x15 [ 0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xA 0xB 0xC 0xD 0xE 0xF ] 0 0` to inject a jumbo frame with the following properties:
 - Checksum: 1 (Classic checksum)
 - Frame ID: 0x15 (Decimal 21)
 - 16 data bytes: [0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xA 0xB 0xC 0xD 0xE 0xF]
 - Slot time in [ms]: 0 (0: Automatic slot time. ≠0: The next frame will start immediately when this time is over.)
 - No more inject frames issued to wait for: 0
- Inject a jumbo frame request:

° Enter `inject 4 0x15 16 [ ] 0 0` to inject a jumbo frame request with the following properties:

- Request a frame from a slave: 4
- Frame ID: 0x15 (Decimal 21)
- Expected data bytes: 16
- Slot time in [ms]: 0 (0: Automatic slot time. ≠0: The next frame will start immediately when this time is over.)
- No more inject frames issued to wait for: 0

• Check the frame monitor for the jumbo frames on the LIN-Bus:

Timestamp	FrameId	FrameData	Checksum	State
+377364	0x05 [0x85]	0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07 0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f	0x87	Other (injected)

12.1.4 Custom application using the Baby-LIN-DII

A custom application is the best way to use jumbo frames since you are able to programmatically load and process the high amount of data. The communication between the LIN-Bus simulation device with USB interface and the custom application is handled by the Baby-LIN-DII.



Advice

The Baby-LIN-DII can be used from custom applications written in C/C++. Wrapper for the .NET framework as well as LabView are available.

There are two possible modes that can be used to inject frames:

Mode	Description
SDF based mode	In this mode a SDF is loaded. The SDF contains the information from the LDF including the schedule tables. Schedules can be executed once or in a loop. The jumbo frames can be injected while a schedule is running.
Monitor mode as master	In this mode no SDF is required. The schedule has to be created manually by injecting frames.

In both cases the following table lists the functions a customer application would usually start with:

Baby-LIN-DII function	Description
<code>BLC_getVersionString()</code>	This function returns the version of the Baby-LIN-DLL. The version should always be mentioned when asking for support.
<code>BLC_getBabyLinPorts()</code>	This function returns information about the Baby-LIN products connected to the PC.
<code>BLC_open()</code>	This function starts the connection with the LIN-Bus simulation device with USB interface.
<code>BLC_getChannelCount()</code> <code>BLC_getChannelHandle()</code> <code>BLC_getChannelInfo()</code>	These function give you information about the channels that are usable from your LIN-Bus simulation device with USB interface.
<code>BLC_registerJumboFrameCallback()</code>	This function registers a callback to receive jumbo frames.

If you want to use a SDF the following functions need to be called:

Baby-LIN-DII function	Description
<code>BLC_loadSDF()</code>	This function loads a SDF into the DLL and down to the LIN-Bus simulation device with USB interface.
<code>BLC_sendCommand("disframe 0xC0000000 1;")</code>	This command subscribes to frame reports for certain frame IDs (0xC0000000 for all IDs).
<code>BLC_sendCommand("start;")</code>	This command starts the LIN-Bus and the first schedule.

If you do not want to use a SDF but the monitor mode the following functions need to be called:

Baby-LIN-DII function	Description
<code>BLC_sendCommand("mon_init 20000 1;")</code>	This function starts the monitor mode as master.
<code>BLC_sendCommand("mon_on 0xC0000000 1;")</code>	This command subscribes to frame reports for certain frame IDs (0xC0000000 for all IDs).

Now you can inject jumbo frames or jumbo frame requests:

Baby-LIN-DII function	Description
<code>BLC_sendCommand("inject 1 0x15 [0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xA 0xB 0xC 0xD 0xE 0xF] 0 0;")</code>	<p>This command injects a jumbo frame with the following properties:</p> <ul style="list-style-type: none"> • Classic checksum: 1 • Frame ID: 0x15 (21) • 16 data bytes: [0x0 0x1 0x2 0x3 0x4 0x5 0x6 0x7 0x8 0x9 0xA 0xB 0xC 0xD 0xE 0xF] • Slot time in [ms]: 0 (0: Automatic slot time. ≠0: The next frame will start immediately when this time is over.) • No more inject frames issued to wait for: 0
<code>BLC_sendCommand("inject 4 0x15 16 [] 0 0;");</code>	<p>This command injects a jumbo frame request with the following properties:</p> <ul style="list-style-type: none"> • Request a frame from a slave: 4 • Frame ID: 0x15 (21) • Expected data bytes: 16 • Slot time in [ms]: 0 (0: Automatic slot time. ≠0: The next frame will start immediately when this time is over.) • No more inject frames issued to wait for: 0

12.2 Diagnostic modes

12.2.1 Supported diagnostic modes

The LIN specification includes the definition of some specific frames and protocols, which are intended for diagnostic purposes.

Basically there are two frames reserved for diagnostics:

Frame-ID	Description	Abbreviation
0x3C	Master Request Frame	MRQ
0x3D	Slave Response Frame	SRP

These frame have some special properties, which you have to take into account:

- They always carry 8 data bytes.
- They always use the classic checksum method.
- A MRQ frame with first data byte set to zero, represents a sleep command for the bus.

Like all other frames, the MQR and SRP frames have to be included into the schedule table to appear on the bus.

So all the functions described in this manual, need a valid SDF file configuration being installed in the Baby-LIN. This SDF must define at least the 2 Frames MasterRequest (0x3C) and a Slave Response (0x3D) and must have a schedule table to use this frames.

Such a simple SDF file is available within this manual. Check chapter "[DiagPure_19200.sdf](#)" for more information.

Due to the fact, that these frames typically will be used in conjunction with some kind of specific diagnostic protocol, the data bytes of these frames are not available via the standard LINWorks GUI interface (SimpleMenu).

Instead some API functions are presented, which allow sending and receiving these frames, via API call to the Baby-LIN-DLL.

There are different Diagnostic modes, which offer different levels of protocol functionality:

Diagnostic modes	Protocol functionality
Diagnostic Off	Basically no data transfer.
Diagnostic RAW	Sending MRQ's and receiving a configurable number of SRP's.
Diagnostic DTL	Implements the Diagnostic Transport Layer according LIN Spec. V.2.x.

In all these modes, there is a command available, which allows to send the MRQ frame as a sleep command.

On Power Up the Baby-LIN will start with diagnostic mode set to Diagnostic OFF.

The selection of a Diagnostic Mode happens through the usage of the appropriate API function calls.

So the API functions `BLC_sendRawMasterRequest`/`BLC_sendRawSlaveResponse` will start the Diagnostic RAW mode, whereas the API calls `BLC_sendDTLRequest` or `BLC_sendDTLResponse` will start the Diagnostic DTL mode.



Version incompatibility

It is not possible to use different diagnostics modes at the same time!

With firmware version V3.60 of the Baby-LIN a new command was introduced, which allows the definition of the requested Diagnostic mode without using one of the above API calls.

This was necessary because in some situations, the diagnostic mode has to be defined and established before the above mentioned API-calls will be issued.

12.2.1.1 Command "dtlmode"

The command `dtlmode` allows to define the diagnostic mode of the Baby-LIN explicit.

<code>dtlmode 0;</code>	Will switch off diagnostic functions.
<code>dtlmode 1;</code>	Will init the diagnostic functions to support diagnostic raw mode.
<code>dtlmode 2;</code>	Will init the diagnostic functions to support diagnostic cooked mode.

12.2.1.2 Diagnostic Off

Basically this mode does not support any data transfers with the MRQ and/or SRP frames.

But the user can specify, how the Baby-LIN will handle the master request frame, when simulating a master and a master request appears in the schedule table.

This configuration is done by "Target Configuration" entry in the Baby-LIN menu of the SimpleMenu. The configuration is stored non volatile in the Baby-LIN.

12.2.1.2.1 Diagnostic Off Master Request frame handling

If configured to dummy data	Send a master request frames with all databytes set to <code>0xFF</code> .
If configured to silent data	Send no frame, but process timeslot (silent frame).

The factory default of this value is silent frame.

12.2.1.2.2 Diagnostic Off Slave Response frame handling

If a Slave response frame is found in the schedule table, a slave response header will be sent. If there will be no response from the bus, no error is generated. This is an exception from the normal Baby-LIN behaviour, where a frame without response would be reported as an error.

12.2.1.3 Diagnostic RAW mode

This mode allows sending a Master Request Frame with all data bytes being defined from the API. The API-calls for receiving slave Requests also will return all databytes contained in the slave response data frames (raw data => raw mode).

12.2.1.3.1 BLC_sendRawMasterRequest

The API call to send a Master Request in the Raw mode is:

```
int BLC_sendRawMasterRequest (BL_HANDLE handle, char* data, int count);
```

Sends the given (non-DTL) MasterReq with the specified 8 bytes as data.

The internal raw-SlaveResponse-buffer is being reset and the Baby-LIN gets instructed to report the next 'count' SlaveResp-frames, which in turn are accumulated into the SlaveResponse-buffer, which can be queried by `BL_getRawSlaveResponse()`.

param handle	Handle representing the channel; returned previously by <code>BLC_getChannelhandle()</code> .
param data	MasterReq frame data (exactly 8 bytes).
param count	Number of expected SlaveResponse frames.
return	Status of operation; '!=0' means successful, '!=0' otherwise.

The data bytes given with this call will be sent with the next MRQ frame schedule entry.

After transmission of the MRQ frame, count consecutive slave responses will be caught and arranged in a char array.



Advice

The processing of the MasterRequest and SlaveResponses in the Baby-LIN device happens asynchronous according to the schedule table selected there.



Version incompatitbility

So you have to make sure, that

- 1 A schedule table containing a MasterRequest and Slave Response Frame is running on the target.
- 2 If issuing multiple MasterRequests, you have to check for a `BL_OK (=0)` return value of the `BLC_sendRawMasterRequest (...)` call.

All return values codes != 0 indicate that a previously issued MasterRequest command has not yet been processed, and thus the new MasterRequest command was rejected. These cases should not be treated as an error in general, but should be handled by retrying the `BLC_sendRawMasterRequest`.

In such a case the application should call `BLC_sendRawMasterRequest()` as long as the return value is !=0.

On the other side, the loop doing this retry operation, should have some kind of time out condition, so the application can handle the situation, that the MasterRequest will never be send, e.g. because a wrong schedule table was selected.

In such a case the state machine in the target can be reset by issuing a `dtlmode 0;` command followed by a `dtlmode 1;` command.

12.2.1.3.2 BLC_getRawSlaveResponse

The values of the slave responses can be retrieved with the API call:

```
int BLC_getRawSlaveResponse (BL_HANDLE handle, unsigned char* data, int length);
```

Sends the given (non-DTL) MasterReq with the specified 8 bytes as data.

The internal raw-SlaveResponse-buffer is filled continuously with the data bytes of reported SlaveResp-frames and is being reset upon every call of `BLC_sendRawMasterRequest()`.

param handle	Handle representing the channel; returned previously by <code>BLC_getChannelhandle()</code> .
param data	Pointer to char array which gets filled (must hold min. 'length' bytes).
param length	How many bytes should get returned.
return	Status of operation; '=0' means successful, '!=0' otherwise.

12.2.1.3.3 BLC_sendRawSlaveResponse

If the Baby-LIN is used to simulate a slave node, it is possible to set it up to respond to a specific Master Request Frame.

This is done with the API call:

<code>int BLC_sendRawSlaveResponse (BL_HANDLE handle, unsigned char * reqdata , unsigned char * reqmask, ...)</code>	
Prepares the Baby-LIN to respond to a specific master request frame (filter parameters given unsigned char * reqdata and unsigned char * reqmask).	
param handle	Handle representing the channel; returned previously by <code>BLC_getChannelhandle()</code> .
param reqdata	Points to an 8 Byte Array with the Master Request values (e.g. <code>MRQVal[8]</code>).
param reqmask	Points to an 8 Byte Array with the Master Request mask (e.g. <code>MRQMask[8]</code>).
param data	Points to a <len> Byte Array with the data which will be send as slave response. len should be a multiple of 8. (8, 16, 24...). If len is bigger then 8, the response will be send in consecutive slave response frames.
param len	Length of the slave response (should be a multiple of 8).
return	Status of operation; '=0' means successful, '!=0' otherwise.

Both pointers reqdata and reqmask point to a 8 Byte Array. The data provided in these arrays will be used as an filter for the master requests.

Example:

```
char MRQVal[8] = {0x55, 0, 0, 0, 0, 0, 0, 0};
char MRQMask[8] = {0xff, 0, 0, 0, 0, 0, 0, 0};
```

With this filter settings, the Baby-LIN will accept every master request frame with first data byte = 0x55 to trigger the slave response.

```
char MRQVal[8] = {1, 2, 3, 4, 5, 6, 7, 8};
char MRQMask[8] = {0xff, 0, 0, 0, 0, 0, 0, 0xff};
```

With this filter settings, the Baby-LIN will accept every master request frame with first data byte = 1 and the last data byte = 8 to trigger the slave response.

The response which the Baby-LIN will deliver, if a Master Request match occurs, is defined in the array data. The length of the response is given with the parameter len.

The behaviour of the Baby-LIN concerning the master request frame as long as no API call is pending is defined by a target configuration command (silent frame or dummy frame).

12.2.1.3.4 Example

Here, two Master Requests with corresponding Slave Responses will be send and monitored with an emulated Master and emulated Slave.

**Advice**

Return values of API-Calls are not checked here to make examples easier, in a real application you should check the return values to make sure the operation was successful.

```
unsigned char masterreq[8] = {0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
// --> Databytes of first Master Request

unsigned char masterreq1[8] = {0x09, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08};
// --> Databytes of second Master Request

unsigned char response[8];
// --> buffer to store slave response

unsigned char MRQMask[8]= {0xff,0,0,0,0,0,0,0xff};
// --> Mask for masterrequest[]

unsigned char responsedata [8] = {0x08,0x07,0x06,0x05,0x04,0x03,0x02,0x01};
// --> Response of Slave for masterrequest[]

unsigned char responsedata1 [8] = {0x0A,0x0B,0x0C,0x0D,0x0E,0x0F,0x10,0x11};
// --> Response of Slave for masterrequest1[]
BL_HANDLE conhandle [2];
// --> Connectionhandle of 2 Baby-LIN devices

BL_HANDLE chanhan[2];
// --> Channelhandle of the LIN-Channel of each device

int comport[2] = {8, 9}; // COM-port of devices

int masterrequest; // Return Value for BLC_sendRawMasterrequest()

conhandle[0] = BLC_open(comports[0]); // open both Baby-LIN devices and get its handle
conhandle[1] = BLC_open(comports[1]);
chanhan[0] = BLC_getChannelHandle(conhandle[0], 1); // Get the handle of the dedicated.
chanhan[1] = BLC_getChannelHandle(conhandle[1], 1); // channel of each device

BLC_loadSDF (chanhandle[0], " DiagnosticSample.sdf", 1); // load SDF in the dedicated
BLC_loadSDF (chanhandle[1], " DiagnosticSample.sdf", 1); // channel of each device

BLC_sendCommand(chanhan[0],"setnodesimu 0 0"); // set Masternode
BLC_sendCommand(chanhan[0],"setnodesimu 1 1"); // for first device
BLC_sendCommand(chanhan[1],"setnodesimu 0 1"); // set Slavenode for
BLC_sendCommand(chanhan[1],"setnodesimu 1 0"); // second device.

// start schedule for both devices
BLC_sendCommand(chanhan[0],"start;");
```

```
BLC_sendCommand(chanhan[1], "start;");

// set devices to rawmode
BLC_sendCommand(chanhan[0], "dtlmode 1;");
BLC_sendCommand(chanhan[1], "dtlmode 1;");

// send RAW Slave Response
BLC_sendRawSlaveResponse(chanhan[0], masterreq, MRQMask, responsedata, 8);

// send RAW Master Request
masterrequest = BLC_sendRawMasterRequest(chanhan[1], masterreq, 1);

// GET raw Slave Response
do {
    std::this_thread::sleep_for(std::chrono::milliseconds(20));
    BLC_getRawSlaveResponse(chanhan[1], response, 8);
    printf("get RAW slave response: %d \n", sc);
}while(sc!=0);

printf("RESPONSE: ");
for(int i = 0; i < 8; i++) {
    printf("%02x ", response[i]);
}
printf("\n ");
```

Same procedure for the second Master request:

1 BLC_sendRawSlaveResponse

The answer for a dedicated masterrequest.

2 BLC_sendRawMasterRequest

Send Master request.(check if the return value of variable „masterrequest“ is zero).

3 while(BLC_getRawSlaveResponse(chanhan[1], response, 8) !=0)

Here, it is important that this function is executed until the masterrequest has reached the slave and the response is sent successfully.

4 Print response

After each slaveresponse, read buffer from BLC_GetRawSlaveResponse().

Again, it is important that the order is kept. Otherwise, timing problems may occur.

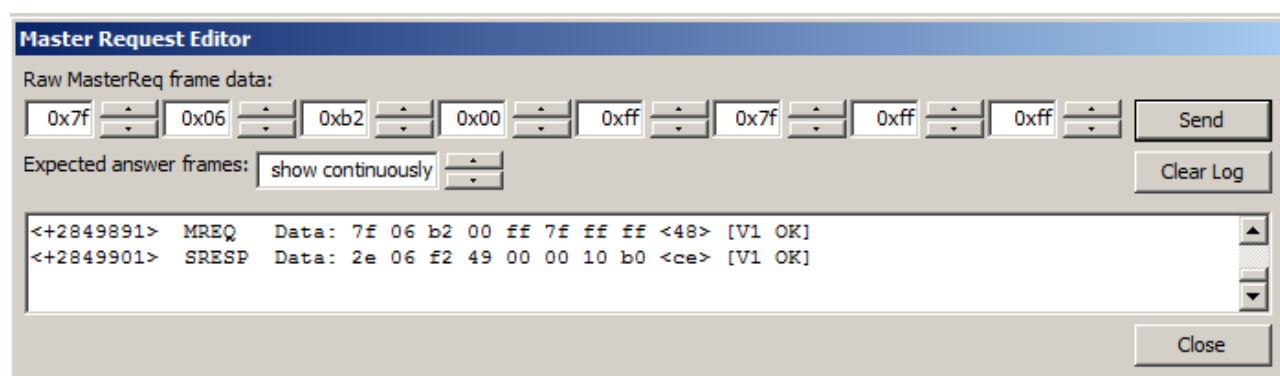
The Print out can be like this:

```
get RAW slave response: -100009 // The Return value will be executed until
get RAW slave response: -100009 // it has a valid slave response
get RAW slave response: -100009
get RAW slave response: -100009
get RAW slave response: -100009
get RAW slave response: -100009
get RAW slave response: -100009
get RAW slave response: -100009
get RAW slave response: -100009
get RAW slave response: 0 // responsedata[]:
RESPONSE: 08 07 06 05 04 03 02 01 // Slaveresponse data of first master request
get RAW slave response: -100009
```

```
get RAW slave response: -100009
get RAW slave response: -100009
get RAW slave response: -100009
get RAW slave response: -100009
get RAW slave response: -100009
get RAW slave response: -100009
get RAW slave response: 0 // responsedata1[]:
RESPONSE: 0a 0b 0c 0d 0e 0f 10 11 // Slave response of second master request.
```

12.2.1.3.5 Master request via SimpleMenu

A raw Master request can also be send in the Linworks SimpleMenu by Simulation / Send Masterrequest.



Master Request Editor

Raw MasterReq frame data:

0x7f 0x06 0xb2 0x00 0xff 0x7f 0xff 0xff

Expected answer frames: show continuously

Log:

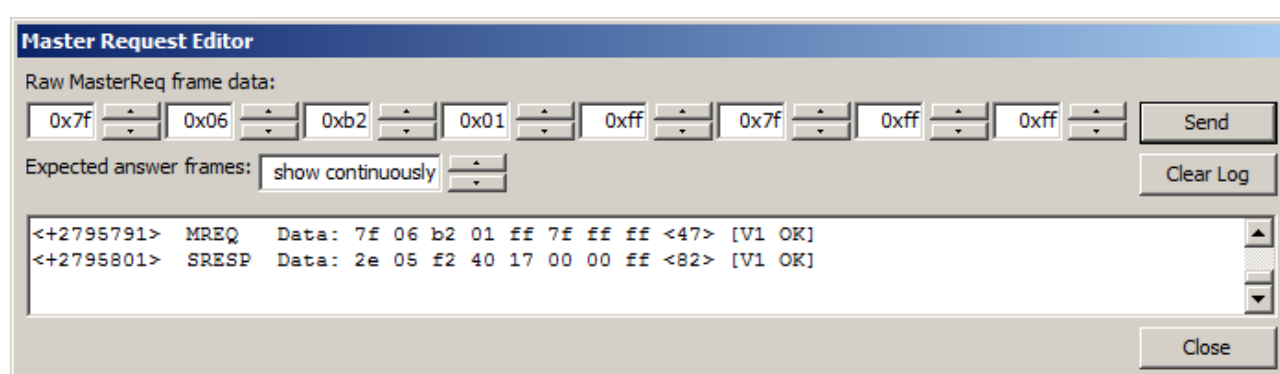
```
<+2849891> MREQ Data: 7f 06 b2 00 ff 7f ff ff <48> [V1 OK]
<+2849901> SRESP Data: 2e 06 f2 49 00 00 10 b0 <ce> [V1 OK]
```

Buttons: Send, Clear Log, Close

Use wildcard if NAD, SupplierId or FuncId are unknown.

Wildcards:

NAD	0x7F
SupplierId	0x7FFF
FuncId	0xFFFF



Master Request Editor

Raw MasterReq frame data:

0x7f 0x06 0xb2 0x01 0xff 0x7f 0xff 0xff

Expected answer frames: show continuously

Log:

```
<+2795791> MREQ Data: 7f 06 b2 01 ff 7f ff ff <47> [V1 OK]
<+2795801> SRESP Data: 2e 05 f2 40 17 00 00 ff <82> [V1 OK]
```

Buttons: Send, Clear Log, Close

12.2.1.4 Diagnostic Cooked Mode (DTL - Diagnostic Transport Layer)

The Baby-LIN supports the transmission and reception of frames according to the Diagnostic Transport Layer specified in the LIN specification. (Firmware version equal V.3.20 or newer).

This specification defines, how the data which is to be transmitted, is packed into the master request or the slave response frame. There are 3 frame types defined:

SF	Single Frame	Used to transmit up to 6 bytes payload (SID + data bytes D1..D5).
FF	First Frame	Used to start the transmission of multiple frames to transport 7...4095 bytes payload, carrying 5 bytes payload (SID + data bytes D1...D4).
CF	Continuation Frame	A multi frame transmission will have several CF's after an initial FF, each CF carrying 6 data bytes.

MasterRequest and Slave response are the transport frames. Data object up to 4095 Byte can be transferred in both directions.

The NAD is the Node Address of the Slave.

The PCI Byte indicates the frame type (SF, FF, CF) and codes some additional information.

There are three different PCI types, which are distinguished by the 4 higher bits of the PCI value.

In the First Frame (FF) the length is composed of 4 bit in the PCI and 8 bits in the following LEN byte which makes up to 12 bits = 4095 maximum available length.

For more details check the LIN Specification (Diagnostic and Configuration-PDU structure).

The SID is the Service Identifier and specifies the request that shall be performed by the slave node addressed, which is included in LEN in a Singleframe (SF) and as an extra-byte in FF. The Continuation Frame does not need the identifier due to belonging to a First Frame.

Type	Coding of PCI type				Additional information			
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SF	0	0	0	0	Length (0...6) SID is included			
FF	0	0	0	1	Length/256			
CF	0	0	1	0	Frame counter (0...15)			

Diag Frametypes

SF - Single Frame
FF - First Frame
CF - ConsecutiveFrame

NAD	PCI = SF	D0	D1	D2	D3	D4	D5
NAD	PCI = FF	LEN	D0	D1	D2	D3	D4
NAD	PCI = CF	D0	D1	D2	D2	D4	D5

PCI composition

PCI-SF

B7...B4	B3...B0
0	Length
1	Length/256
2	Framecounter

Length 0...6, which is maximum payload length in SingleFrame Message

PCI-FF

In a FirstFrame (FF) PCI is always followed by additional LEN Byte.

Maximum Payload length = 4095 (12 Bit)

PCI-CF

Framecounter in first CF = 1, in second CF = 2 etc. If more than 15 frames, frame counter wraps around and continues with 0, 1, 2,...

Example: SF-Request with SF-Response

Request	0x3C	0x0A	0x03	0x22	0x06	0x2E	0xFF	0xFF	0xFF
Response	0x3D	0x0A	0x06	0x62	0x06	0x2E	0x80	0x00	0x00

Example: SF-Request with MultiFrame-Response (FF + 2*CF)

Request	0x3C	Nad	0x03	0x22	0x06	0x5E	0xFF	0xFF	0xFF
Response	0x3D	0x0A	0x10	0x0E	0x62	0x06	0x5E	"3"	"C"
Response	0x3D	0x0A	0x21	"8"	"9"	"5"	"9"	"5"	"3"
Response	0x3D	0x0A	0x22	"7"	" "	" "	0xFF	0xFF	0xFF

The RSID is the Response Service Identifier and specifies the contents of the response. This allows messages with a length of up to 4096 data bytes to be send with one command given from the API. The segmentation of the message into SF, FF and CF frames is done by the Baby-LIN DLL resp. firmware as well as buffering of the data.



Version incompatitbility

The Baby-LIN firmware versions prior V.3.40 were limited to messages with a maximal size of 1024 Bytes. From version V.3.40 on, the firmware supports the processing of messages with up to 4095 Bytes, which is the maximal allowed vale according to the LIN specification.



Version incompatitbility

To allow the DTL communication to work, there must be a SDF loaded, which includes Master Request and Slave Response Frames in its schedule table and this schedule has to be selected to run.

The Baby-LIN-DLL offers API functions to pass the contents of a message to be sent by the master repsonse frame or the Slave Respnse frame, when simulation a slave node.

The Baby-LIN API expects the SID resp. RSID bytes to be given as first bytes in the data buffer, which are given with the calls `BL_sendDTLRequest` and `.BL_sendDTLResponse`.

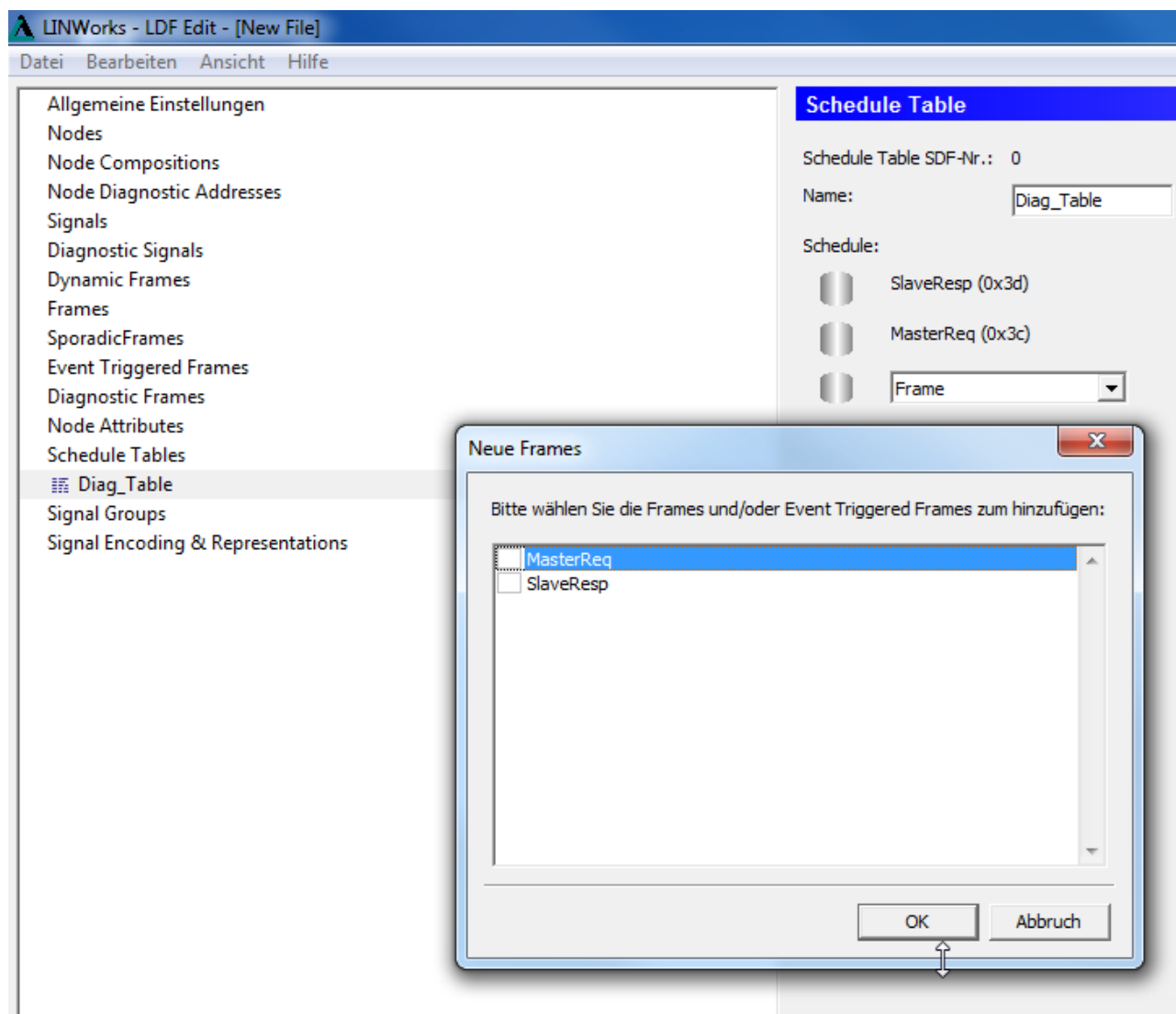
To use the DTL feature, you actually need to implement a program, which calls this API functions, as the standard LINWorks application programs do not support the DTL mechanism.

12.2.1.4.1 Cooked Request via SimpleMenu

The following example shows how to send a Cooked Request in LINWorks SimpleMenu:

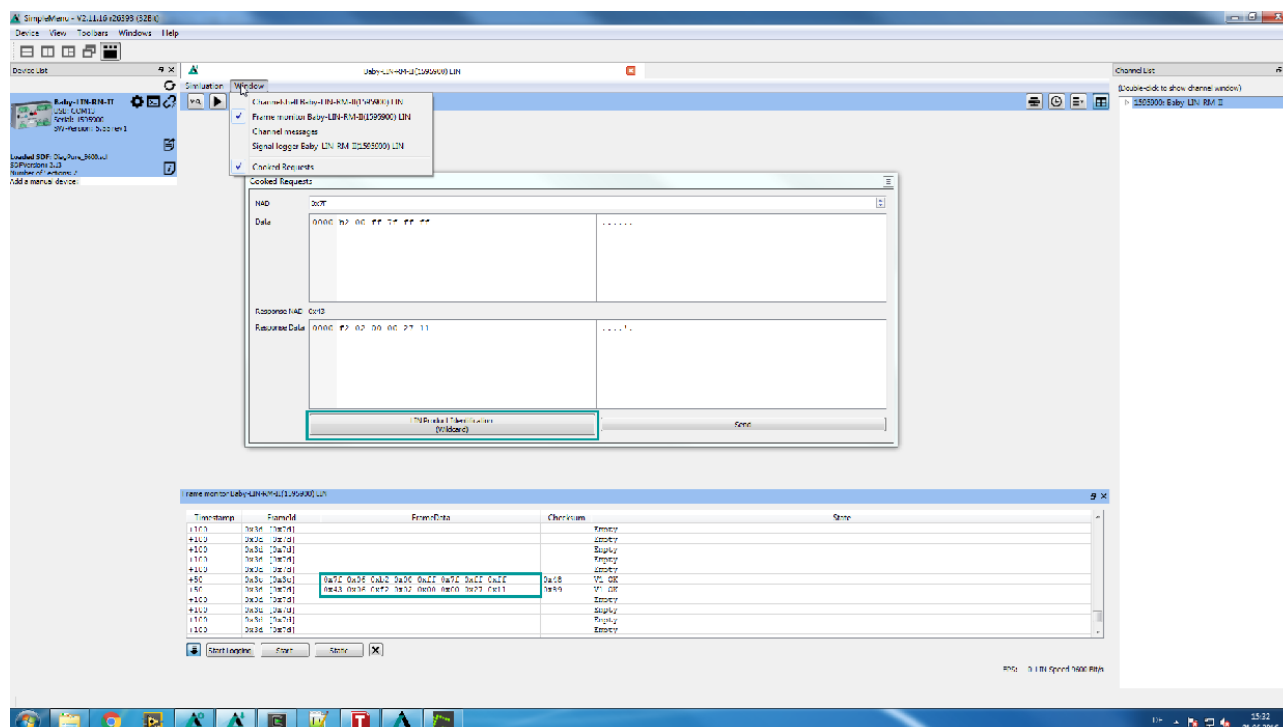
Here, a device (working as a slave) which is able to respond to a DTL Request with a Baby-LIN is used. It is important, that the SDFFile contains a schedule with diagnostic frames. In the LINWorks LDFEdit click on "Diagnostic frames" and then "activate".

Now, Frames for a request and respond can be added to the Schedule table.



After creating a SDFFile with the SessionConf out of the LDFFile, this can be used for a Cooked request in SimpleMenu.

After running the scheduler, a cooked request can be sent via Windows -> Cooked Request -> LIN Product Identification (wildcard) OR type in a dedicated request. If you do not know the NAD of your slave device you can simply use the wildcard option to get a response of all slaves connected to the bus. As you will see on the frame monitor, an empty respond frame (ID: 0x3D) is sent until a cooked request is executed and the request frame (ID: 0x3C) is sent. Then, the respond frame (0x3C) will be filled with data by the slave.



12.2.1.4.2 API

These are the API calls to use diagnostic cooked mode:

12.2.1.4.3 BLC_sendDTLRequest

To send a Master Request message with up to 4095 data bytes and utilizing the diagnostic transport layer use the API call:

<pre>int BLC_sendDTLRequest (BL_HANDLE handle, unsigned char nad, int length, char * data);</pre>	
Sends the given DTL MasterReq to the node identified by 'nad'.	
param handle	Handle representing the channel; returned previously by BLC_getChannelhandle().
param nad	NAD of the node the request gets send to.
param length	Length of the following data array.
param data	DTL frame data (begins with SID, followed by up to 4095 data byte).
return	Status of operation; '=0' means successful, '!=0' otherwise.

12.2.1.4.4 BLC_sendDTLResponse

<code>int BLC_sendDTLResponse (BL_HANDLE handle, unsigned char nad, int length, char* data);</code>	
Sends the given DTL SlaveResp for the node identified by "nad".	
param handle	Handle representing the channel; returned previously by <code>BLC_getChannelhandle()</code> .
param nad	NAD of the node the request gets send for.
param length	Length of the following data array.
param data	DTL frame data (begins with RSID, followed by up to 4095 data bytes).
return	Status of operation; '=0' means successful, '!=0' otherwise.

12.2.1.4.5 BLC_getDTLRequestStatus

<code>int BLC_getDTLRequestStatus (BL_HANDLE handle);</code>	
Returns the status of the last request send operation.	
param handle	Handle representing the channel; returned previously by <code>BLC_getChannelhandle()</code> .
return	Status of last request operation if ≥ 0 ; see <code>BL_DTL_STATUS</code> for values. For < 0 , see standard return values for error, or for textual representation (for return values < -1000) <code>BLC_getLastError()</code> .

12.2.1.4.6 BLC_getDTLResponseStatus

<code>int BLC_getDTLResponseStatus (BL_HANDLE handle);</code>	
Returns the status of the last response send operation.	
param handle	Handle representing the channel; returned previously by <code>BLC_getChannelhandle()</code> .
return	Status of last response operation if ≥ 0 ; see <code>BL_DTL_STATUS</code> for values. For < 0 , see standard return values for error, or for textual representation (for return values < -1000) <code>BLC_getLastError()</code> .

12.2.1.4.7 BL_DTL_STATUS

`BL_DTL_STATUS` values:

0	<code>LD_COMPLETED</code>
1	<code>LD_FAILED</code>
2	<code>LD_IN_PROGRESS</code>

12.2.1.4.8 BLC_getNextDTLResponse

If simulating a master, the Slave DTL responses can be retrieved with the API call:

```
int BLC_getNextDTLResponse (BL_HANDLE handle, BL_dtl_t *frame);
```

Fetches the next complete DTL response from the receiver queue.

param handle	Handle representing the channel; returned previously by <code>BLC_getChannelhandle()</code> .
param frame	Pointer to a <code>BLC_DTL</code> structure, to store the response information.
return	Status of operation; '=0' means successful, '!=0' otherwise.

The response data is written into the struct `BLC_DTL`, which is pointed to by `BLC_DTL *frame`.

As long as there is no complete DTL slave response is available in the receiver queue, this API call will return the value: -100009.

12.2.1.4.9 BLC_getNextDTLRequest

If simulating a slave node, a pending DTL master request can be retrieved with the API call:

```
int BLC_getNextDTLRequest (BL_HANDLE handle, BL_dtl_t *frame);
```

Fetches the next complete DTL request from the receiver queue.

param handle	Handle representing the channel; returned previously by <code>BLC_getChannelhandle()</code> .
param frame	Pointer to a <code>BLC_DTL</code> structure, to store the request information.
return	Status of operation; '=0' means successful, '!=0' otherwise.

The request data is written into the struct `BLC_DTL`, which is pointed to by `BLC_DTL *frame`.

As long as there is no complete DTL master request is available in the receiver queue, this API call will return the value: -100009.

12.2.1.4.1 BLC_DTL

Contents of a `BLC_DTL` structure:

```
typedef struct _BLC_DTL{
    // ! Status of protocol frame
    BL_DTL_STATUS status;

    // ! NAD of protocol frame
    unsigned char nad;

    // ! Length of the data-array.
    int length;

    // ! frame data, beginning with the (R)SID.
    unsigned char data[4*1024];
} BLC_DTL;
```

12.2.1.4.1 Example: Simulated master

In the following code example a simulated master (Baby-LIN) is used to send a Cooked request to a real slave device. To get the slave response, the corresponding function is polled until the slave has filled the response-frame with data.

```
BL_HANDLE chanhan, conhan;
BLC_DTL Request;
BL_DTL_STATUS status;
BLC_FRAME frame;
unsigned char data [6] = {0xb2,0x00,0xff,0x7f,0xff,0xff};
int i,j,sc =0;

// Initialization
conhan = BLC_open("/dev/ttyUSB0");
BLC_loadSDF(conhan,"DiagPure.sdf",1);
chanhan = BLC_getChannelHandle(conhan, 1);
sc = BLC_loadSDF(conhan,"DiagPure_9600.sdf",1);
BLC_sendCommand(chanhan,"disframe 255 1;");
BLC_sendCommand(chanhan,"start;");
BLC_setDTLMode(chanhan,2);

// Send the Cooked Request
BLC_sendDTLRequest(chanhan,0x7F,sizeof(data),&data);
// Wait until the Status for the request is done.
do{
    sc= BLC_getDTLRequestStatus(chanhan);
    printf("RequestStatus? %d\n",sc);
} while((status!=0) || (sc!=0));
sleep(1); // A waiting period is required to be able to receive the respond

// Read out the response
BLC_getNextDTLResponse(chanhan,&Request);
printf("Size DTL_Respond: %d \n",sizeof(Request));
printf("Infos about Respond:\n");
printf("Status: %d \n",Request.status);
printf("NAD: %d \n",Request.nad);
printf("Length: %d \n",Request.length);
for (int i =0; i<Request.length;i++) {
    printf("Data[%d]: c= %02x \n",i,Request.data[i]);
}

BLC_sendCommand(chanhan,"stop");
```

Print out:

```
RequestStatus? 2
RequestStatus? 2
RequestStatus? 2
RequestStatus? 0
Size DTL_Respond: 4108
Infos about Respond:
Status: 0
```

```
NAD: 67
Length: 6
Data[0]: c= f2
Data[1]: c= 02
Data[2]: c= 00
Data[3]: c= 00
Data[4]: c= 27
Data[5]: c= 11
```

12.2.1.4.1 Callbacks

Another way of receiving the respond (instead of using `BLC_getNextDTLResponse`) is to use the following callback functions:

12.2.1.4.1.1 BLC_registerDTLRequestCallback

```
int BLC_registerDTLRequestCallback (BL_HANDLE handle, BL_dtl_request_callback_func *cb);
```

Registers a callback function, which is called on every reception of a DTL request, but only if at least one Slave is emulated.

Issuing a NULL-pointer de-registers the callback function. As the function is called from another thread context, take care of thread-safety (i.e. using mutexes, etc.).

param handle	Handle representing the channel; returned previously by <code>BLC_getChannelhandle()</code> .
param cb	Pointer to a function call-compatible to <code>BLC_dtl_request_callback_func</code> .
return	Status of operation; '!=0' means successful, '!=0' otherwise.

12.2.1.4.1.2 BLC_registerDTLResponseCallback

```
int BLC_registerDTLResponseCallback (BL_HANDLE handle, BL_dtl_response_callback_func cb);
```

Registers a callback function, which is called on every reception of a DTL response, but only if the Baby-LIN emulates the master node.

Issuing a NULL-pointer de-registers the callback function. As the function is called from another thread context, take care of thread-safety (i.e. using mutexes, etc.).

param handle	Handle representing the channel; returned previously by <code>BLC_getChannelhandle()</code> .
param cb	Pointer to a function call-compatible to <code>BLC_dtl_response_callback_func</code> .
return	Status of operation; '!=0' means successful, '!=0' otherwise.

The callback functions are both declared as a void function with one calling parameter of the type `BL_dtl_t`. So you have to implement and register appropriate callback functions, if you want to use the callback interface.

```
typedef void (BLC_dtl_request_callback_func) (BL_HANDLE, BLC_DTL frame);
typedef void (BLC_dtl_response_callback_func) (BL_HANDLE, BLC_DTL frame);
```

Every time a complete DTL response or request will be available, the appropriate callback function will be called, importing the DTL message data by the calling parameter of by the type `BLC_DTL`.

**Version incompatibility**

Due to the memory requirements for the necessary buffers, it is not possible to emulate a DTL Master and a DTL Slave within the same Baby-LIN at the same time.

**Advice**

If a Baby-LIN is used to simulate a slave in a DTL session, the cooked mode must be activated by issuing the command `dtlmode 2;`.

12.2.1.4.1 Callback example

The same code as for the `BLC_getNextDTLResponse` is used here, except for using `BLC_registerDTLResponseCallback` with a corresponding callback function.

```
#include <stdio.h>
#include <BabyLIN.h>
#include <BabyLINCAN.h>

static void dtl_response_callback(BL_HANDLE bh, BLC_DTL frame) {
    int i;
    fprintf(stderr, "nad = %d\n", frame.nad);
    for (i=0; i<frame.length; i++) {
        fprintf(stderr, "%s Data[%d]: c= %02x \n", __func__, i, frame.data[i]);
    }
    fprintf(stderr, "\n");
}

int main(void) {
    BL_HANDLE chanhan, conhan;
    BLC_DTL Request;
    BL_DTL_STATUS status;
    BLC_FRAME frame;

    unsigned char data [6]= {0xb2, 0x00, 0xff, 0x7f, 0xff, 0xff};
    int i, j, sc =0;

    conhan = BLC_open("/dev/ttyUSB0");
    sc = BLC_loadSDF(conhan, "DiagPure_9600.sdf", 1);
    chanhan = BLC_getChannelHandle(conhan, 1);
    sc = BLC_setDTLMode(chanhan, 2);

    BLC_sendCommand(chanhan, "start;");
    BLC_sendCommand(chanhan, "disframe 255 1;");
    BLC_sendDTLRequest(chanhan, 0x7F, sizeof(data), &data);
    BLC_registerDTLResponseCallback(chanhan, dtl_response_callback);

    do{
        sc = BLC_getDTLRequestStatus(chanhan);
        printf("RequestStatus? %d\n", sc);
    } while((status!=0) || (sc!=0));
}
```

```
sleep(1);

BLC_sendCommand(chanhan, "stop");
BLC_registerDTLResponseCallback(chanhan, NULL);
}
```

Print output:

```
RequestStatus? 2
RequestStatus? 2
RequestStatus? 0
nad = 67
dtl_response_callback Data[0]: c= f2
dtl_response_callback Data[1]: c= 02
dtl_response_callback Data[2]: c= 00
dtl_response_callback Data[3]: c= 00
dtl_response_callback Data[4]: c= 27
dtl_response_callback Data[5]: c= 11
```

12.2.2 Sleep Command (special Master Request Frame)

The sleep command allows putting the LIN-Bus into the sleep mode.

The sleep command delivers the specific Master Request Frame with all data bytes set to zero. The Baby-LIN will stop the bus emulation after sending this frame.

The API call to do that is:

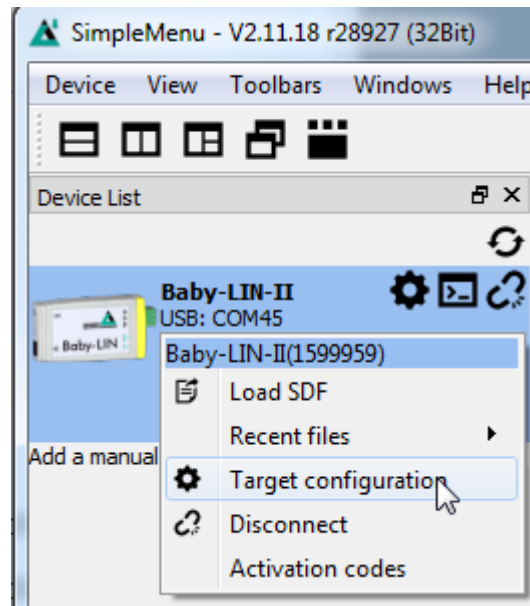
```
BLC_sendCommand (handle, "sleep ;");
```

There will be a negative return value, if the Baby-LIN does not simulate a Master Node.

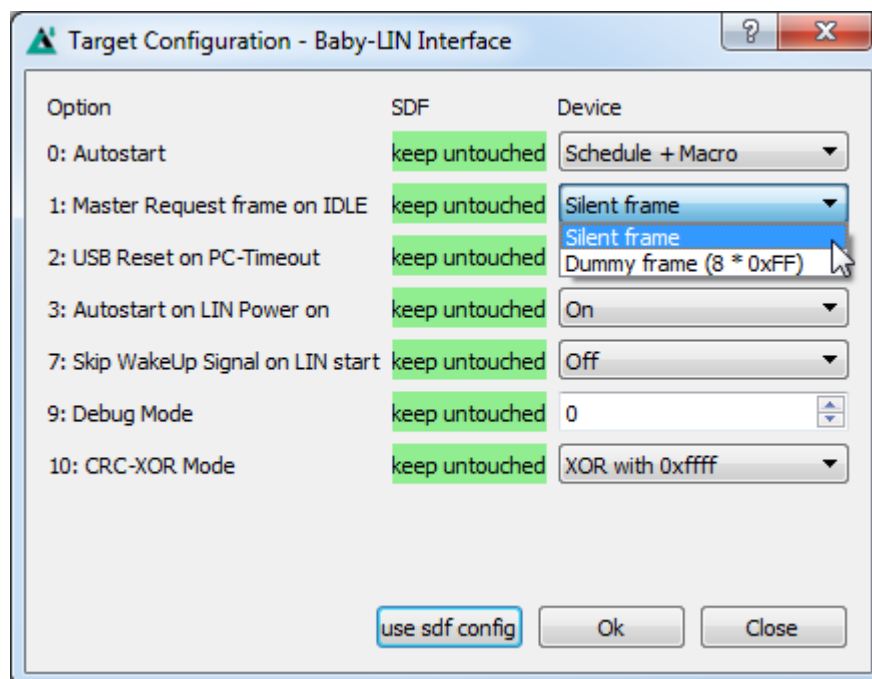
12.2.3 Target configuration

The Baby-LIN allows configuring the kind of master request frames to be send as long as no data for the master requests is supplied by API calls.

The Target configuration is available in the LINWorks component SimpleMenu.



If your Baby-LIN is connected to the PC, a menu will appear, showing the available parameters and the actual settings.



The item Master Request frame on IDLE has two choices.

Silent frame	No frame will be sent, but the timeslot will be executed, which will result in a silent frame.
Dummy frame	as long as no other master request data is applied by the application, a frame with all data bytes set to 0xFF will be sent on every master request frame entry found in the schedule table.

The Target configuration is stored in the Baby-LINs onboard EEPROM memory. This will only work, if the Baby-LIN is connected to the PC during this procedure.

12.3 ASCII mode

12.3.1 Introduction

The ASCII mode is a special mode the Baby-LIN supports. It allows using the LIN-Bus of the Baby-LIN as mere level-translator or sending characters via the LIN hardware layer without the LIN frame packing conventions.

Within this mode the Baby-LIN will no longer communicate with the PC using a proprietary binary format but ASCII commands. Therefor it is no longer required to use the "Baby-LIN-DLL".

Within this mode you can only use the LIN-Bus channel. All other channels can't be operated in this mode.

The Baby-LIN can be switched into the ASCII mode by using a command line tool or by sending a few special bytes via the virtual COM port of the USB connection.

The ASCII modes stays activated until the Baby-LIN is powered off and on again or a special command is sent.

12.3.2 Switch into the ASCII mode

There are two ways to switch the Baby-LIN into the ASCII mode:

- Use the Switch to ASCII tool that is available as part of the LINWorks.
You should use this tool, if you want to test the ASCII mode or manually want to switch into the ASCII mode.
Check chapter "Switch to ASCII tool" for more information.
- Send a special byte sequence to the virtual COM port of the USB connection.
You should use this method, if you want to use the ASCII mode in a custom application.
Check the following steps for a tutorial.

Please follow these steps to switch the Baby-LIN into the ASCII mode from a custom application:

- Open the virtual COM port of the USB connection of the Baby-LIN using the following settings:
 - COM port: The virtual COM port of the Baby-LIN
 - Baud rate: 921,600 Bd



Advice

On older windows operating systems it may not be possible to set this baud rate. Therefor it is also possible to set a baud rate of 300 Bd. The "Baby-LIN driver" will then automatically change it to 921,600 Bd.

- Data bits: 8
- Stop bits: 1
- Parity: None
- Flow control: None

- Write the following bytes to the connection:
Hex: 0x80, 0x95, 0x61, 0x73, 0x63, 0x69, 0x69, 0x8f
0x80, 0x95, 0x61, 0x73, 0x63, 0x69, 0x69, 0x8f

	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Hex:	0x80	0x95	0x61	0x73	0x63	0x69	0x69	0x8f

- If the switch to the ASCII mode was successful, the Baby-LIN will answer with the following byte sequence:

	Byte 1	Byte 2	Byte 3
Hex:	0x21	0x0D	0x0A
ASCII:	!	<CR>	<LF>

12.3.3 Switch off the ASCII mode

The Baby-LIN stays within the ASCII mode until one of the following events happen:

- The Baby-LIN is powered off and back on again.
- The `RESET` command is sent.

12.3.4 ASCII Commands

12.3.4.1 Commands Overview

The ASCII commands consists of a command string, an optional and variable list of parameters and a closing <CR>.

The parameters are integer numbers and can be entered as decimal and hexadecimal numbers:

Number format	Description	Example
Decimal	Decimal numbers are entered without prefix or postfix.	255
Hexadecimal	Hexadecimal numbers are entered with a trailing H.	FFH

The following commands are available:

Command	Description
ECHO	Switches on debug outputs.
VERSION	Requests the firmware version.
TARGETSPEED	Sets the baud rate of the LIN-Bus driver.
LINFRAME	Send a frame on the LIN-Bus.
TRANSPARENT	Directly send characters to and receive from the LIN-Bus.
RESET	Switch off the ASCII mode.

**Attention**

The following commands exist, but are reserved for internal usage. Do not send one of these commands to your Baby-LIN. These commands may switch the Baby-LIN into a state, from which you can not recover it.

- ERASE
- FIRMWARE
- PRODTST

12.3.4.2 ECHO

This command enables or disables the `ECHO` mode. If the `ECHO` mode is enabled, additional debug texts and line feed characters are returned from the Baby-LIN.

The characters from the PC are never repeated, since that would cause problems due to the half duplex nature of the LIN-Bus signal line.

Syntax:	ECHO [flag]<CR>	
Parameter [flag]:	0	ECHO mode disabled
	1	ECHO mode enabled
Return value:	!<CR>	Success
	?<CR>	Failure
Example:	<pre><< ECHO 1<CR> >> !<CR></pre>	

12.3.4.3 VERSION

The `VERSION` command queries the Baby-LIN for its firmware version. The Baby-LIN will return its firmware version.

Syntax:	VERSION<CR>	
Return value:	Baby-LIN V.4.99<CR>	The firmware version of the Baby-LIN.
Example:	<pre><< VERSION<CR> >> Baby-LIN V.4.99<CR></pre>	

12.3.4.4 TARGETSPEED

This command sets the baud rate of the LIN-Bus interface.

**Warning**

The `SETSPEED` command is deprecated and replaced by the `TARGETSPEED` command. Do no longer use the `SETSPEED` command.

Syntax:	TARGETSPEED [baudrate]<CR>
----------------	----------------------------

Parameter [baudrate]:	1 Baud - 200 kBaud	The baud rate of the LIN-Bus interface.
Return value:	!<CR>	Success
	?<CR>	Failure
Example:	<pre><< TARGETSPEED 19200<CR> >> !<CR></pre>	

12.3.4.5 LINFRAME

This command will send a LIN frame via the LIN-Bus interface.

The checksum type can be selected and the checksum itself will be created by the Baby-LIN.

The identifier of the frame needs to be entered as unprotected ID. This is the way it is used within the LDFile. The protected ID will be added, when the frame is sent on the LIN-Bus by the Baby-LIN by adding the two protection bits.

The length of the frame will be evaluated from the number of data bytes entered.

Syntax:	LINFRAME [cstype] [id] [data1] [data2] ... [dataN]<CR>	
Parameter [cstype]:	0	Classic checksum according to LIN v1.x.
	1	Enhanced checksum according to LIN v2.x.
Parameter [id]:	0 - 63	The unprotected ID of the frame.
Parameter [data1] to [dataN]:	0 - 255	The value of a data byte.
Return value:	!<CR>	Success
	?<CR>	Failure
Example:	<pre><< LINFRAME 1 10 1 2 3 4 5 6 7 8<CR> >> !<CR></pre>	

12.3.4.6 TRANSPARENT

This command will switch the Baby-LIN into a transparent data exchange mode. All ASCII characters sent to the Baby-LIN will be passed directly to the LIN-Bus interface and vice versa.

Syntax:	TRANSPARENT [echo] [timeout]<CR>	
Parameter [echo]:	0	Echo mode is disabled. Characters sent by the Baby-LIN will not be read back.
	1	Echo mode is enabled. Characters sent by the Baby-LIN will be read back.
Parameter [timeout]:	1 - FFFFFFFEH	The timeout in [ms]. If no characters will be received within this period, the Baby-LIN will leave the transparent mode automatically and is ready to receive new ASCII commands.
	0 or FFFFFFFFH	No timeout is set. The Baby-LIN will stay in the transparent mode until it is powered off and on again.
Return value:	!<CR>	Success
	?<CR>	Failure

Example:	<< TRANSPARENT 1 10000<CR> >> !<CR>
-----------------	--

While the device is in the transparent mode, the LEDs "LD1 - Run LED" and "LD2 - Error LED" will blink alternatively.

12.3.4.7 RESET

This command will cause a reset of the Baby-LIN. The result will be a termination of the ASCII mode.

Syntax:	RESET<CR>	
Return value:	!<CR>	Success
	?<CR>	Failure
Example:	<< RESET<CR> >> !<CR>	

12.3.5 Application example

Here are the steps to create a custom application that sends and receive characters via the LIN-Bus interface:

Step	Format	Data	Description
1	Binary	80H 95H 61H 73H 63H 69H 69H 8FH	Enter the ASCII mode.
2	ASCII	TARGETSPEED 115200	Set the baud rate of the LIN-Bus interface to 115200 Baud.
3	ASCII	TRANSPARENT 0 FFFFFFFFH	Enter the transparent mode without echo and timeout.
4...	ASCII characters		Send and receive ASCII characters via the LIN-Bus.

This initial procedure has to be done only once after the Baby-LIN is powered on.

13 FAQ - Frequently asked questions

13.1 General questions

13.1.1 Where can I download software, firmware, documentation, etc.?

The latest versions of our software, firmware and documents are available in our client portal: portal.lipowsky.de. There you can find downloads like:

- Software, especially the LINWorks suite.
- Device firmware for the Baby-LIN. The BLProg update tool is included in the archive.
- USB drivers for the Baby-LIN.
- Manuals and datasheets.
- Application notes and examples how to create custom applications.



Advice

The LINWorks archive contains not only the LINWorks software but also the USB driver, manuals, datasheets, application notes and examples.

Only the device firmware packages are not included. The firmware is available as separate package.

Before you can download anything you have to register yourself in our customer portal. Here you also have to select which product you are using. After your account has been activated by us you will receive an e-mail. This can take up to some hours.

After you received the activation e-mail you will be able to login. In the login screen you should also select your preferred language. After the login you will be able to download all relevant files.

Login
Registration
Contact
Imprint

Select language /
Sprache wählen
English
change/ändern

Registration

Registration

You are interested in Software Downloads and further information?

Please fill out our registration form to become a user of our portal.

Companyinfo

Company *

Street *

ZIP *

City *

Phone

German
Default-language

Fax

Email *

Username (3-20 characters) *

Password (min. 6 characters) *

Passwordcheck

Contact

Mr.
Salutation *

Name *

Surname *

Register

Groups

Baby-Lin User

Distributors

POSCO Users

Groups *

(Notice: You are customer of one of our products or sales prospect? Please choose the relevant groups to get access to downloads and further information!)

Login
Registration
Contact
Imprint

Select language /
Sprache wählen
English
change/ändern

Login

Welcome to the Customerportal of Lipowsky

Username *

Password *

Login

not yet registered? [register here...](#)

After the registration you will also be automatically notified via e-mail when new versions are available.

**Advice**

If you have forgotten your password, please send us your E-Mail address, that you used for the registration. Then we can reset your password so you can login again.

13.1.2 How can I find out the device type and serial number?

Each device from the Lipowsky Industrie-Elektronik GmbH has certain properties like the following:

- Device type
- Hardware revision
- Article number
- Serial number

The easiest way to figure out these data is to check the label on the device. But there are other options as well. Check chapter "[Identification of the Baby-LIN device](#)" for more information.

13.2 SDFFile

13.2.1 Can a SDFFile be extracted from a device?

No. If a SDFFile was downloaded using the SimpleMenu or the Baby-LIN-DLL, it can not be extracted from the device.

13.3 Voucher and activation codes

13.3.1 Are there voucher codes available? Can this device be enhanced using activation codes?

There are no voucher codes available for the Baby-LIN. It can not be enhanced via activation codes.

13.4 Stand-alone mode

13.4.1 How can I activate the stand-alone mode?

The Baby-LIN is able to operate stand-alone without a PC, PLC or operator. To enable this mode several requirements need to be met:

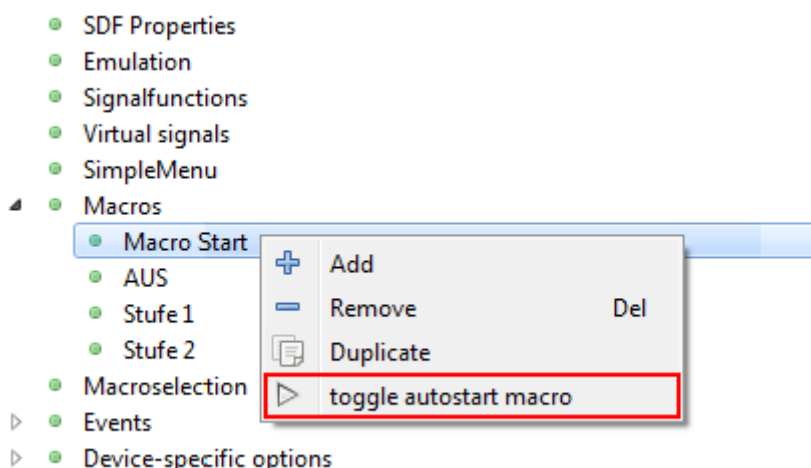
- You need a SDF with a macro that is marked as "autostart" macro.
- The SDF needs to be stored persistently on the Baby-LIN.
- The device needs to be configured to automatically start the autostart macro of a persistently stored SDF when powered up.

13.4.2 How can I configure the autostart macro?

You can mark a macro in a SDF as autostart macro. This means that this macro is started automatically when the SDF is loaded. Usually this macro will start the LIN-Bus communication and perform necessary initialisations.

To mark a macro as autostart macro the following steps are necessary.

- Open your SDF using the SessionConf.
- If this SDF does not already have a macro that you want to use as autostart macro please create one.
- Right click on the macro and select "toggle autostart macro". The macro will now have the "[autostart]" marker.



Advice

Please note that only one macro can be marked as autostart macro.



Tip

You probably want to start the LIN-Bus within the macro since it is not automatically started by loading the SDF.

13.4.3 How can I store a SDF persistently in a device?

It is possible to store a SDF persistently on the Baby-LIN. Therefore a special option in the SDF must be configured:

- Start the SessionConf and load your SDF.
- In the left menu click on "SDF Properties".
- Check the "Store SDF in device persistently" option.

- Save your SDF and reload it to a device using the SimpleMenu.



Advice

The option "Store SDF in device persistently" can always be found in the "SDF Properties". It is independent from the SDF version (SDF v2, SDF v3) or type of section currently selected. This option is global for the whole SDF even though it can be found in every section.

13.4.4 How can I configure the Baby-LIN to automatically load a SDF and start a macro?

The Baby-LIN can be configured to automatically start a autostart macro. The Autostart can be triggered if the device is powered on or if the LIN-Bus is powered on. These target-specific options (also known as target configuration) are persistently stored on the Baby-LIN.

There are 2 important options that affect the autostart feature:

Target-specific option	Possible values	Description
Autostart	Off	No schedule or macro is started when the autostart is triggered.
	Schedule	The first schedule but not the autostart macro is started when the autostart is triggered.
	Schedule + Macro	The first schedule and the autostart macro are started when the autostart is triggered.
	Macro only	No schedule but the autostart macro is started when the autostart is triggered.
Autostart on LIN Power on	Off	An autostart is triggered only when the device is powered on.
	On	An autostart is triggered each time the LIN-Bus is powered on.



Advice

If you want to start another but the first schedule simply select the macro only option and start another than the first schedule from that macro.



Tip

Triggering the autostart each time the LIN-Bus is powered on is especially helpful for applications in which the node connected to the LIN-Bus is changed periodically, e.g. EOL applications. In such a case, the simulation may be started automatically every time a new node is connected and the LIN-Bus voltage is turned on.



Warning

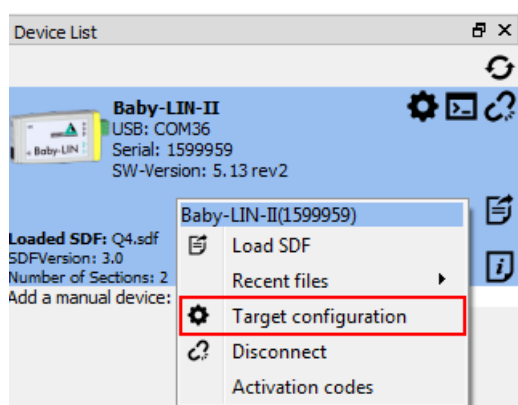
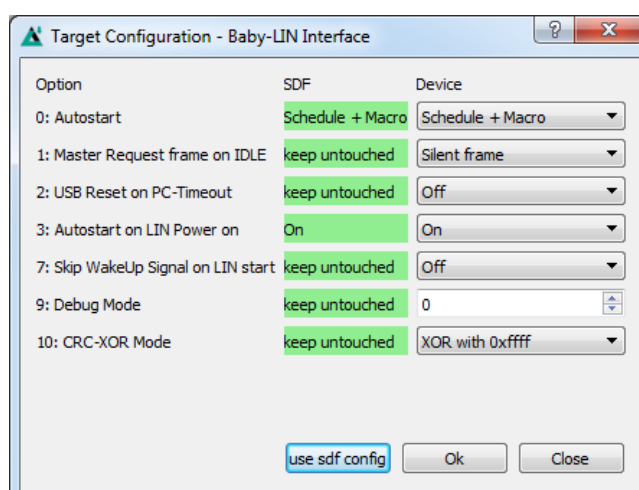
The option "Autostart on LIN Power on" is not required for the Baby-LIN if the LIN voltage is also used as the logic supply.



Version incompatibility

Newer revisions of the Baby-LIN support the new SDF v3 file format. One of the features of SDF v3 is the setting of preset values for target-specific options.

The target-specific options can be set using the SimpleMenu. Right click on the connected device and click on "Target configuration". In the following dialog you can change the values in the device. For comparison reasons the preset values of the SDF are shown.



13.5 Troubleshooting

13.5.1 How can I find out the device type and serial number?

Each device from the Lipowsky Industrie-Elektronik GmbH has certain properties like the following:

- Device type
- Hardware revision
- Article number
- Serial number

The easiest way to figure out these data is to check the label on the device. But there are other options as well. Check chapter "Identification of the Baby-LIN device" for more information.

13.5.2 How can I check, if the USB connection and/or firmware are working properly?

The USB connection and firmware can only be checked together. A step by step guide is available. Check chapter "[USB connection and firmware check](#)" for more information.

13.5.3 How can I check, if the LIN communication is working properly?

The LIN-Bus communication can be checked without the need for another node. A step by step guide is available. Check chapter "[LIN-Bus communication](#)" for more information.

14 Troubleshooting

14.1 Introduction

This troubleshooting guide is designed to help you, if you have trouble operating the Baby-LIN.

It consists of several topics that help you to identify the problem. Some solutions can be easily applied by yourself while others require the intervention of trained personnel. In the latter case an error report is required. Therefore you should print out and fill in the data in chapter "Error report". Some problems can be solved by your local distributor while others require a repair of the Lipowsky Industrie-Elektronik GmbH.

The next chapter gives you an overview over the covered problems. If you are an experienced Baby-LIN user or already narrowed down the problem, you can directly jump to the relevant topic. Otherwise you may want to follow the topics in order.

14.2 Overview

The following table shows you a list with problems and links to the guide with the solutions.

Problem	Description	Chapter
What software is required and which applications should I install?	This chapter tells you which software is required.	"Required software"
What is the exact product name of my device? What is the hardware revision of my Baby-LIN? What is the serial number of my Baby-LIN?	If you are not sure what device you are using or you need an exact identification, this chapter helps you to identify the exact device type.	"Identification of the Baby-LIN device"
Is the USB port of the Baby-LIN defect? I can not find the Baby-LIN on my PC. I can not establish a connection between the Baby-LIN and a PC.	This chapter will help you, if you can not establish a connection to a PC via USB. It will explain you how to check for hardware and software errors.	"USB connection and firmware check"
How can I update a broken firmware? What can I do if a firmware update fails?	This chapter will help you, if the firmware of your Baby-LIN is broken and you need to repair it.	"Firmware update"
A certain SDF feature does not work. What can I do?	This chapter will help you, if a certain SDF feature does not work properly.	"SDF configuration"
I may have damaged the LIN-Bus interface because of a wiring mistake. How can I test, if the LIN-Bus interface is working properly?	This chapter will guide you through a process that will test the LIN-Bus interface without the need for any other LIN-Bus nodes.	"LIN-Bus communication"
I have another problem.	If you could not find a problem here you may want to check the FAQ. If that also does not help you please contact our support.	"FAQ - Frequently asked questions" "Support information"

14.3 Topics

14.3.1 Required software

The following table shows you the 3 most important components that you should install when using the Baby-LIN:

Component	Description
Baby-LIN driver	This driver is required to connect the Baby-LIN to a PC.
SessionConf	This application is required to create and edit SDFs.
SimpleMenu	This application is required to establish a connection to the Baby-LIN, read out information and upload SDFs.

Here you can find more information:

Topic	Chapter
Where to download the software?	"Downloads"
How to install the software?	"Installation"
An overview over the components of the LinWorks suite.	"LINWorks suite"

14.3.2 Identification of the Baby-LIN device

14.3.2.1 Introduction

If you are not very familiar with the Baby-LIN device family you may have problems identifying the type of your device. The type, identity and properties of a device can be derived from certain information. The following table shows you what this information tell you:

Information	Description
Device type	<p>This is the general device type. Depending on this type you should choose the datasheet and manual you are using.</p> <div>  <div> Advice Please check if you are using the correct manual right now. </div> </div>
Hardware revision	This is the revision of the hardware. During its lifecycle the Baby-LIN can be improved by altering details of its hardware to become more robust and resistant against failures. Each alteration gets a new hardware revision. This number may be required for e.g. hardware alterations.
Article number	This number identifies the device type. It is used only internally by the Lipowsky Industrie-Elektronik GmbH. With this number we can derive the device type and hardware revision from this number.
Serial number	This serial number is the most important information when communicating with the Lipowsky Industrie-Elektronik GmbH. With this number we can derive the device type, hardware revision and article number.

The following chapters explain how you can find out these information.



Advice

If you require the device type and/or the hardware revision, but only have the article number or the serial number, please contact us. Check chapter "[Support information](#)" for more informations.

14.3.2.2 Using the label

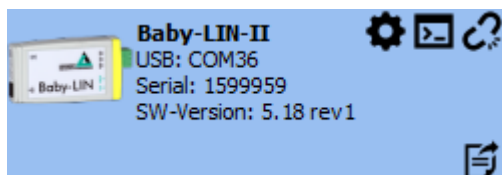
Each device is delivered with a label containing all important information. This is the easiest way to identify your device:

 <p>Baby-LIN-II-A01 Artikel: 8000840 S/N: 1599959</p>	<p>The first line contains two information:</p> <ol style="list-style-type: none"> 1 The device type: Baby-LIN-II 2 The hardware revision: A01 <p>The second line contains the article number: 8000840</p> <p>The third line contains the serial number of the device: 1599959</p>
--	--

14.3.2.3 Using the SimpleMenu

If the label is no longer readable or present on your Baby-LIN you can read out the serial number using the SimpleMenu. Just follow the following steps:

- Connect your Baby-LIN using a USB cable to your PC.
- Start the SimpleMenu.
- The "Device List" should now show at least one device.
- Establish a connection by clicking the "Connect" button.
- The first line displays the device type and the third line the serial number of the Baby-LIN.



14.3.2.4 Using delivery notes

All the information are also present on the delivery note you received from Lipowsky Industrie-Elektronik GmbH. If you ordered your Baby-LIN from a distributor the delivery notes may not contain these informations.

We will deliver to you the following positions:

Pos.	Art-No.	Article Name	Amount	Batch Number
1	8000899	BABY-LIN-MB-II-RevB03	1	Total amount delivered
		Multi-bus gateway with RS-232 and LAN host interface. Hardware includes 2 * LIN, 1 * CAN high speed, digital input and output, switchable LIN supply, 2 piggy-pack sockets for MIF extension modules. Integrated short time ups for save system shutdown. ARM A5 main CPU running embedded Linux. Activation of LIN1 interface with SDF-V3 Support is included. Separately sold options: Activation codes to support for LIN2 and CAN-HS interface. Up to 2 MIF-LIN-02 modules each with 2 LIN interfaces so the system can support a total of 6 LIN bus lines.	1 Stück	batch 19987001
		country of origin: Germany customs tariff number : 90309085 weight: 460 gr		
		SerialNumbers [19987001]: 1606147		

Information	Location	Example
Device type	Third column, first part of "Article Name"	Baby-LIN-MB-II
Hardware revision	Third column, second part of "Article Name"	B03
Article number	Second column, "Art-No."	8000899
Serial number	Bottom line, "SerialNumbers [...]:"	1606147

14.3.2.5 Ask your distributor

If none of the previous methods worked, you can contact your local distributor or Lipowsky Industrie-Elektronik GmbH. See chapter ["Support information"](#) for more information.

14.3.3 USB connection and firmware check

The functioning of the USB connection and the firmware need to be checked in one sequence since both have to work without error to communicate with the processor of the Baby-LIN.

Please follow the instructions below to check if a USB connection can be established and if the firmware reacts like it is supposed.

If you experience a problem or behaviour that is not described below, please contact the Lipowsky Industrie-Elektronik GmbH. Check chapter ["Support information"](#) for more information.

- 1 Remove all connectors from the Baby-LIN.

Known problems	Recommended action
The can not be switched off, since it does not react to user input.	<ul style="list-style-type: none"> The firmware seems to have a failure. Check chapter "Firmware update" for more information.

- 2 Connect the Baby-LIN using the provided USB cable.

3 Check if the green RUN-LED LD1 is blinking.

Known problems	Recommended action
The Baby-LIN is not supplied with voltage by the USB port of the PC.	<ul style="list-style-type: none"> • Replace the USB cable. • Use another USB port. • Use an active USB Hub. • Try another PC.
The firmware may have a failure.	<ul style="list-style-type: none"> • The firmware seems to have a failure. Check chapter "Firmware update" for more information.
A rare hardware failure may have happened.	<ul style="list-style-type: none"> • Exclude firmware failures. Check chapter "Firmware update" for more information.

4 Check if the Baby-LIN is listed in the device manager. It should be listed under the "Ports (COM & LPT)" node. If you have problems finding the device, disconnecting and reconnecting can help you. Simply look out for changes.



Advice

The Baby-LIN should be listed as: Baby-LIN LIN Bus Interface.

Known problems	Recommended action
The Windows device manager shows no new device, because the USB interface is not supplied with voltage by the USB port of the PC.	<ul style="list-style-type: none"> • Replace the USB cable. • Use another USB port. • Try another PC.
The Windows device manager shows an "unknown device" and/or lists the device in the "Other devices" group, because the Baby-LIN driver is not installed.	<ul style="list-style-type: none"> • (Re-)Install the newest Baby-LIN driver.

5 Start the SimpleMenu and check if the Baby-LIN is listed in the "Device List".

6 Click the connect button.

If the connection could be established, the SimpleMenu shows informations like the serial number and firmware revision. This tells you, that the USB connection and firmware are working and no failure could be detected.

If no connection could be established, the SimpleMenu will not show any information. In this case please continue with the next step.

7 Please check the LED LD1. Is it blinking once every second for about 50 ms? Now please add the LIN-Bus voltage and check the LED LD1 again. Is it now blinking once every second for about 500 ms?

Known problems	Recommended action
The firmware may have a failure.	<ul style="list-style-type: none"> • The firmware seems to have a failure. Check chapter "Firmware update" for more information.

14.3.4 Firmware update

Despite intense testing of firmware versions it may happen, that an error leads to the failure of the Baby-LIN. Such failures may have different causes and impacts. Since we are constantly developing the firmware, an update to the newest firmware version could solve problems. Therefore we recommend updating the firmware to the newest firmware version whenever you are experiencing problems.

Please follow the instructions below to install a new firmware.

- 1 Download the newest firmware from our customer portal. Check chapter ["Downloads"](#) for more information.
- 2 Remove all connectors from the Baby-LIN.
- 3 Connect the Baby-LIN using the provided USB cable.
- 4 Start the `blprog.exe` and follow the instructions.
- 5 Check the result of the `blprog.exe`. If the flashing of the firmware was successful, continue with the following steps. If the flashing fails read below.
- 6 Remove the USB cable.
- 7 Test the Baby-LIN with your target application. If the Baby-LIN still does not work as supposed keep reading below.

If the flashing does not work the first time, we recommend repeating the procedure one more time. If the flashing keeps failing or the flashed device still does not work as supposed, the problem can not be resolved by you. One of the following points may be the reason:

- A firmware update was aborted.
- The wrong firmware was flashed (e.g. a Baby-LIN-RM firmware flashed to a Baby-LIN).
- A rare hardware failure.

Since the problem can not be resolved by you, the Baby-LIN has to be sent in. Check chapter ["Shipping instructions"](#) for more information.

14.3.5 SDF configuration

Sometimes a problem occurs after downloading a SDF into the Baby-LIN. In most of these cases the firmware of the Baby-LIN does not support certain features of the SDF. This usually happens, when the LinWorks version is newer than the firmware of the Baby-LIN. It can be fixed by updating the firmware. Additionally you should have a look at our update philosophy.

Early firmware versions also had a problem with persistently stored SDFs. It could happen, that a Baby-LIN, which had a SDF persistently stored, could get a problem with its SDF configuration, after programming a new firmware. This effect can be fixed by updating the firmware. Newer firmware versions will erase persistently stored SDF automatically before starting a firmware update and therefore do not have this problem.

Our update philosophy	Check chapter "Update philosophy" for more information.
Update the firmware:	Check chapter "Firmware update" for more information.
If a firmware update already failed:	Check chapter "Firmware update" for more information.

14.3.6 LIN-Bus communication

This guide shows you how to test the LIN-Bus interface. If this test succeeds you know, that the LIN-Bus interface hardware is working.

Please follow the instructions below to test the LIN-Bus interface.

- 1 Remove all connectors from the Baby-LIN.
- 2 Connect the Baby-LIN using the provided USB cable.

- 3 Please check the LED LD1. It should now blink once every second for about 50 ms.
- 4 Now connect the LIN-Bus connector. Make sure the LIN-Bus supply voltage is present and within the valid area of 8-36 VDC. The LIN-Bus line itself should be not connected with any nodes that could interfere with the test.
- 5 Please check the LED LD1. It should now blink once every second for about 500 ms?
- 6 Start the SimpleMenu, connect to your device and load the "LinCommunicationTest.sdf" that is embedded in this manual. Check chapter ["Embedded files"](#) for more information.
- 7 Now check the LED LD2. If it blinks or is permanent on LIN-Bus errors occur. This means the LIN-Bus interface hardware has a failure.

LD2	Description
Off	No LIN-Bus errors occur. The LIN-Bus interface is working as supposed.
On or blinks	Errors are occurring on the LIN-Bus. The LIN-Bus interface hardware seems to have a failure.



Version incompatibility

This test will test the LIN-Bus driver. Even if this test succeeds a failure could be between the LIN-Bus driver and the output pins.

14.4 Shipping instructions

There are several reasons why you would want to sent in your Baby-LIN:

- An unresolvable firmware error.
- A hardware failure.
- Any custom modification.



Advice

In the first two cases please make sure you tried all tips from the troubleshooting guide. Check chapter ["Troubleshooting"](#) for more information.



Version incompatibility

In the first two cases please fill out the error report so we know exactly what your problem is. Check chapter ["Error report"](#) for more information.

Depending on where your company is located you should ship your Baby-LIN to your local distributor or directly to the Lipowsky Industrie-Elektronik GmbH:

Location:	Outside the EU	Inside the EU
Recipient:	Local distributor	Lipowsky Industrie-Elektronik GmbH
Shipping address:	Check chapter "Distributors" for more information.	Check chapter "Support information" for more information.

**Attention**

To prevent additional costs, please contact the recipient for additional shipping details, before shipping your Baby-LIN.

14.5 Error report

General information	(You can simply add a business card instead of filling out these two fields)
Company information <ul style="list-style-type: none"> • Address • Division 	
Contact person <ul style="list-style-type: none"> • Name • Phone • E-Mail 	

Distributor information	
Where have you bought your device?	

Device information	
Device type:	
Serial number:	
Firmware version:	

Software information					
Software	Used by you?	Version	Software	Used by you?	Version
LDFEdit	☐ yes ☐ no		Baby-LIN driver	☐ yes ☐ no	
SessionConf	☐ yes ☐ no		Baby-LIN-DLL	☐ yes ☐ no	
SimpleMenu	☐ yes ☐ no		Baby-LIN-DLL .NET Wrapper	☐ yes ☐ no	
			Baby-LIN-MB-Tool	☐ yes ☐ no	

Problem description				
Please describe your problem <ul style="list-style-type: none"> • Which feature is causing problems? • How is the device setup? • Which interfaces are you using? • What have you tested? (Please use the backside for more details)				
Is the USB connection working?	☐ yes	☐ no	☐ not tested	☐ not used
Have you tried a firmware update?	☐ yes	☐ no		
Is the LIN-Bus interface working?	☐ yes	☐ no	☐ not tested	☐ not used

15 Embedded files

15.1 Access embedded files

Some parts of this manual show or require certain files. To allow access to these files, they are embedded within this PDF document. They can be extracted using typical PDF viewers.

To access a single file you can click on the Link. Alternatively you can open the attachment view and save the files you want.



Advice

The attachment view often is an alternate view to the contents table and often symbolized using a paper clip.

15.2 LinCommunicationTest.sdf

This SDF can be used to test the LIN-Bus interface. One master and two slaves will be simulated by the Baby-LIN. Therefore a communication over the LIN-Bus interface is simulated so that the LIN-Bus driver is tested.

15.2.1 LinCommunicationTest

[LinCommunicationTest \[v1.0\]](#)

15.3 DiagPure_19200.sdf

This SDFFile can be used to communicate using diagnostic frames. It contains a LDFile with a Master Request and a Slave Response frame. Both frames a periodically put on the LIN-Bus by the "DiagTable" schedule.

15.3.1 DiagPure_19200

[DiagPure_19200](#)