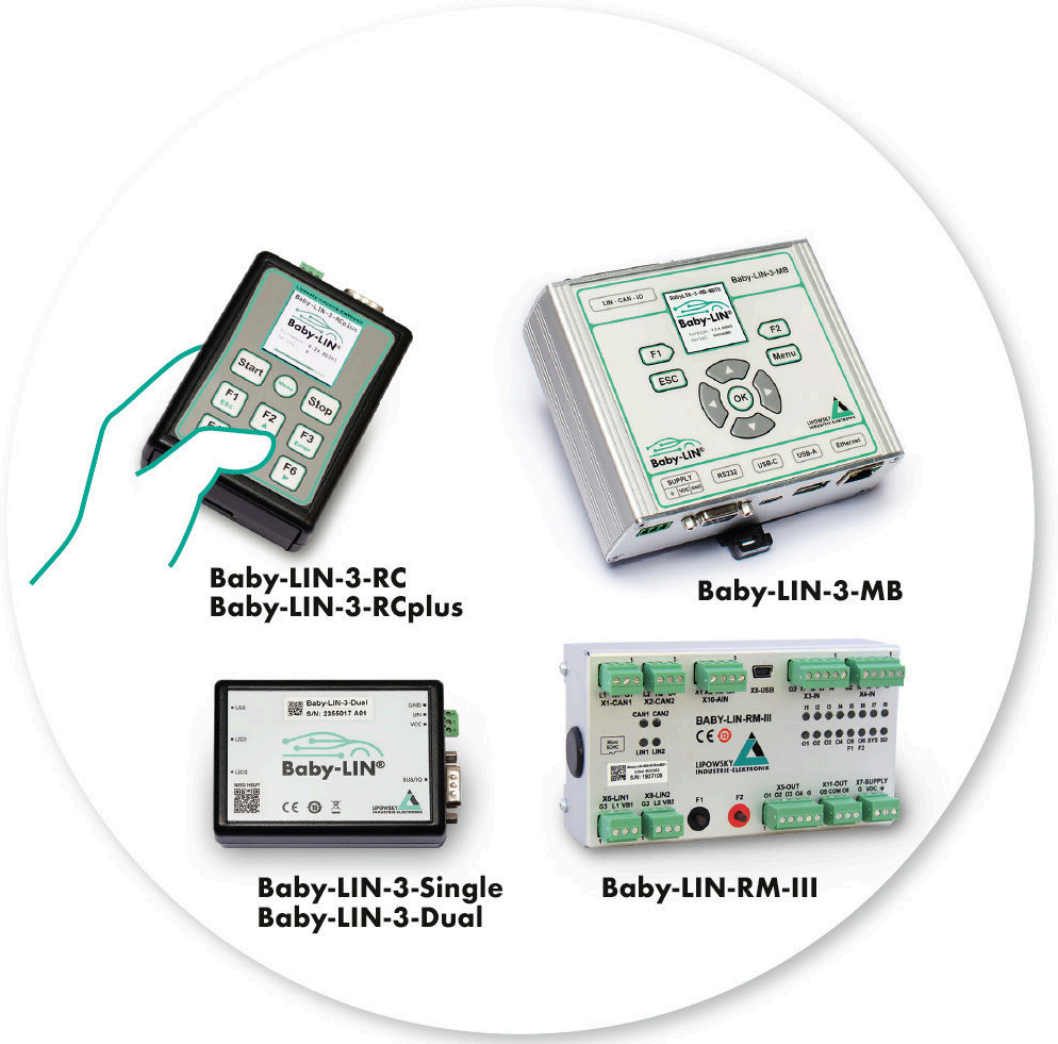
LIPOWSKY
INDUSTRIE-ELEKTRONIK



Logging

Application_Note V1.1

Lipowsky Industrie-Elektronik GmbH

Römerstraße 57 | 64291 Darmstadt | Germany

Phone: +49 (0) 6151 / 93591 - 0

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

1 Overview **2**

2 Log format **3**

2.1 3rd party ASCII . . . 3

2.2 Raw format . . . 4

3 SimpleMenu Logging **5**

4 SDF Logging **8**

4.1 Commands . . . 8

4.2 Configuration parameters . . . 10

4.3 Workflow . . . 12

4.3.1 Initialization . . . 12

4.4 Writing log data . . . 12

4.5 Finish logging . . . 13

5 RAM logger **14**

5.1 Saved data and format . . . 14

5.2 Storage size . . . 14

5.3 Download log data . . . 14

5.3.1 Safe to internal memory . . . 14

5.3.2 Safe to external memory . . . 16

5.3.3 Automatic download . . . 16

6 microSD card Logging **17**

6.1 Introduction . . . 17

6.2 SDF settings . . . 17

6.3 Using 3rd party ASCII format . . . 18

6.4 Using raw format . . . 18

7 Support information **19**

©2025 Lipowsky Industrie-Elektronik GmbH
Römerstraße 57 | 64291 Darmstadt | Germany
Phone: +49 (0) 6151 / 93591 - 0
Website: www.lipowsky.com

1 Overview

All Baby-LIN-Devices have the ability to log any kind of data. It is possible to trigger the logging in different ways and to select the log source. Typical log sources are frames, processed data, like e.g. serial numbers read using protocol services and application status information or the result of a process.

The following table gives an overview what is possible in general:

Description	Devices	Target	Description
SimpleMenu Logging	All	A file on a PC	A log config file can be imported into the SimpleMenu and the data is saved to a file on the PC, good for development.
SDF Logging	All	• A file on a PC • Devices microSD card	Configuration and trigger is set inside the SDF via MacroCommand, most versatile and controllable.
DLL logging	All except MBs	A file on a PC	Same as SDF Logging but a customer application utilizes BLC_Commands triggers and configures the logging.
RAM logger	MBs only	• The internal RAM • Devices microSD card • A USB storage device	Continuously logs data on the device which than can be saved, if needed.

©2025 Lipowsky Industrie-Elektronik GmbH
Römerstraße 57 | 64291 Darmstadt | Germany
Phone: +49 (0) 6151 / 93591 - 0
Website: www.lipowsky.com

2 Log format

Depending on the further processing of the log data there are different types defining the formatting of the final file.

Format	Number	Logged data	Description
Binary format (Deprecated)	0	Frames	Propritary format to minimize the used space. Shouldn't be used for new projects.
3rd party ASCII	1	All	ASCII format that is used by third party products from other vendors.
Raw format	6	Custom	All logging data is written in form of Debug reports allowing a full customization.

2.1 3rd party ASCII

This format is compatible with most formats from other vendors, e.g. Vector. All selected data is automatically logged and no other action is necessary. This also means that the files can get quiet big so carefully select which data you want to log and in which case you want to safe them, e.g. only erroneous tests.

The typical structure looks like this:

```
date Fri May 12 13:38:13 2017
base hex timestamps absolute
internal events logged
// version HARP-4 V.1.43 Build1
```

2.788508 Li	11 Tx 1 00	checksum = ff	CSM = classic
2.788508 Li	11 Tx 1 00	checksum = ff	CSM = classic
3.288498 Li	12 Rx 8 43 a1 16 d0 a7 53 29 00	checksum = 10	CSM = classic
3.538493 Li	13 Rx 0	NodeResponseMissing	
3.788488 Li	11 Tx 1 01	checksum = fe	CSM = classic
4.288531 Li	12 Rx 8 4d a6 19 d3 a9 54 2a 00	checksum = f6	CSM = classic
4.538525 Li	13 Rx 0	NodeResponseMissing	
4.788519 Li	11 Tx 1 02	checksum = fd	CSM = classic
5.288509 Li	12 Rx 8 56 ab 1d d5 ab 55 2a 00	checksum = df	CSM = classic
5.538504 Li	13 Rx 0	NodeResponseMissing	
5.788499 Li	11 Tx 1 03	checksum = fc	CSM = classic
6.288489 Li	12 Rx 8 60 b0 20 d8 ad 56 2b 00	checksum = c6	CSM = classic
6.538536 Li	13 Rx 0	NodeResponseMissing	
6.788531 Li	11 Tx 1 04	checksum = fb	CSM = classic
7.288521 Li	12 Rx 8 6a b5 23 da af 57 2b 00	checksum = af	
7.538515 Li	13 Rx 0	NodeResponseMissing	
7.788510 Li	11 Tx 1 05	checksum = fa	CSM = classic
8.288500 Li	12 Rx 8 74 ba 27 dd b1 58 2c 00	checksum = 95	CSM = classic
8.538495 Li	13 Rx 0	NodeResponseMissing	
8.788490 Li	11 Tx 1 06	checksum = f9	CSM = classic
9.288532 Li	12 Rx 8 7e bf 2a df b3 59 2c 00	checksum = 7e	CSM = classic
9.538527 Li	13 Rx 0	NodeResponseMissing	
9.788522 Li	11 Tx 1 07	checksum = f8	CSM = classic
10.288511 Li	12 Rx 8 88 c4 2d e2 b5 5a 2d 00	checksum = 65	CSM = classic
10.538506 Li	13 Rx 0	NodeResponseMissing	
10.788501 Li	11 Tx 1 08	checksum = f7	CSM = classic

The frame line components are explained using the first frame:

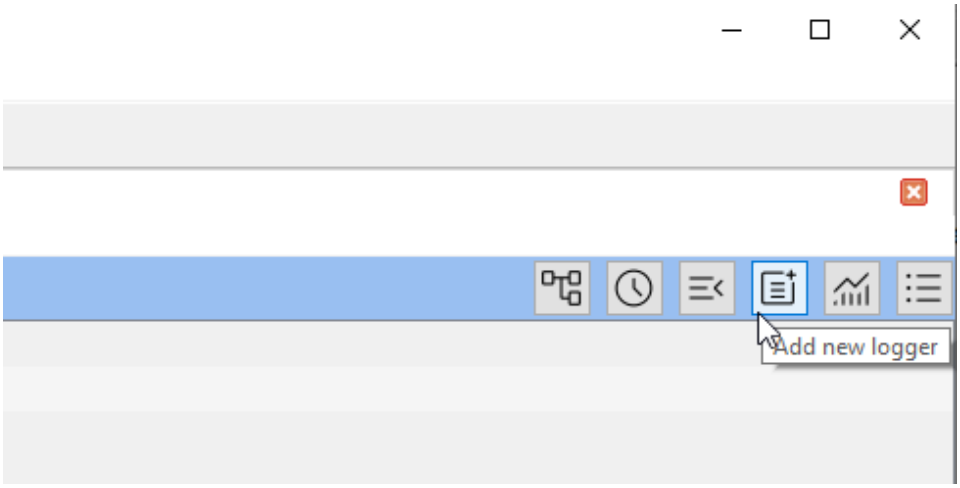
Part	Description
`2.788508`	This is a timestamp. It represents the seconds since the Logging was powered on.
Li	A bus identifier
11	The ID of the frame
Tx	The direction of the frame
1	The length of the frame
00	All data bytes of the frame
checksum = ff	The checksum of the frame.
CSM = classic	The checksum type used by the frame

2.2 Raw format

The Raw format is logging data by using the print command. So everything can be logged and saved in a custom format, but all logging has to be done by hand in the SDF by calling the `Print` macro command.

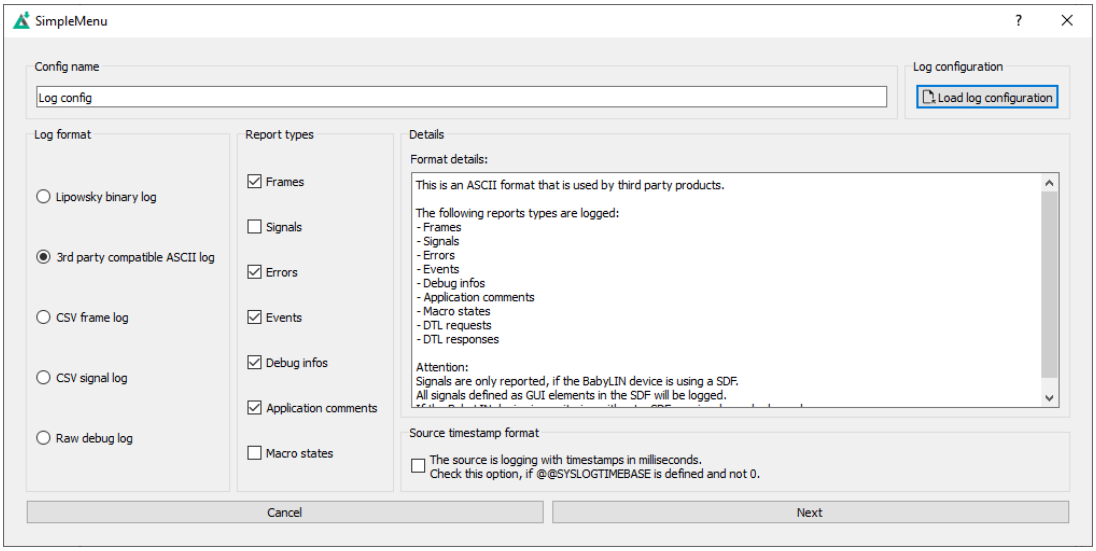
3 SimpleMenu Logging

The SimpleMenu offers the possibility to record the complete bus communication of the Baby-LIN-Device and to save it locally in a file on the computer. To do this, open the integrated log menu in the SimpleMenu by clicking on the `Logger` context button in the upper right corner.

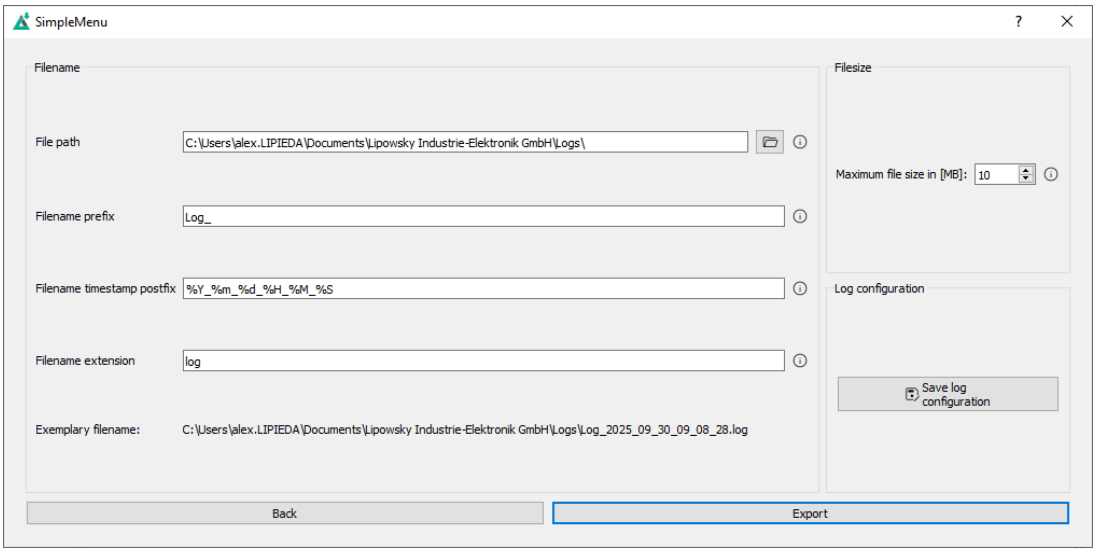


After clicking the button `+ Add logger` a new logger is added and the configuration window opens. Existing configurations can be loaded here or new ones can be created and saved for later.

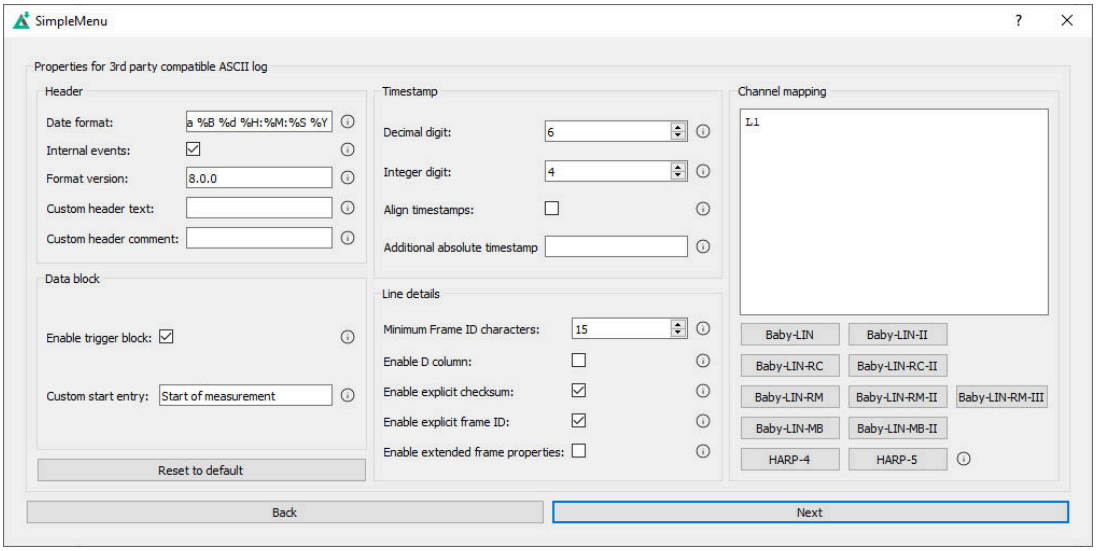
Warning
Lipowsky binary log, CSV frame log and CSV signal log are deprecated and shouldn't be used anymore.



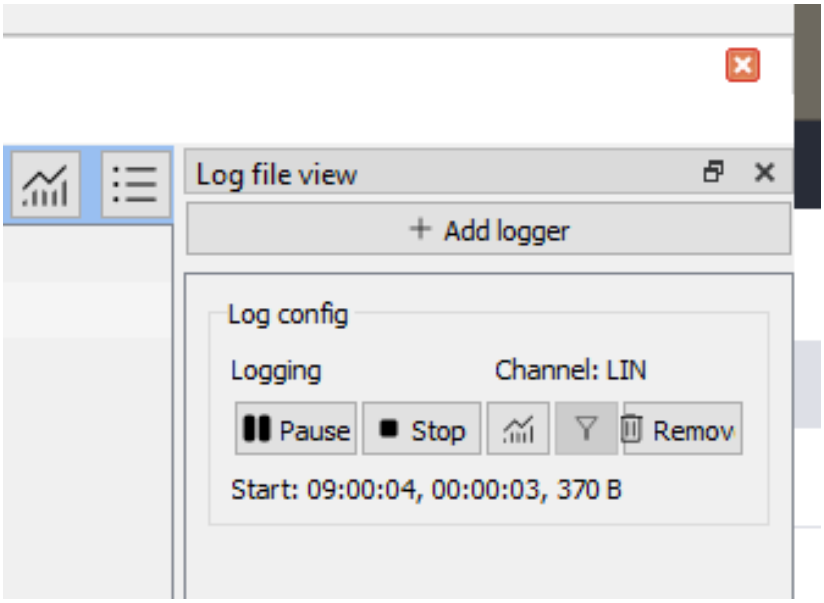
After selecting the log format you can choose how the file should be named and where it should be saved.



For the 3rd party ASCII you can also configure some of the header information. They are partly explained here Configuration parameters.



After the configuration you have the basic log view in which you can start, pause, stop or remove the logger or change it's configuration.


Log file view
Add logger
Log config
Logging
Channel: LIN
Pause
Stop
Remov
Start: 09:00:04, 00:00:03, 370 B

4 SDF Logging

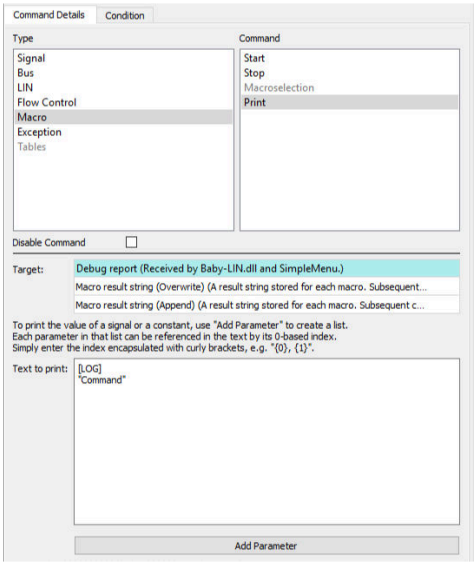
The SDF offers the possibility to set the logging configurations inside the file and execute LOG commands via the macro command "Print on Debug Report". This means that the bus communication can be recorded regardless of how the SDF is executed on the device, allowing easy and subsequent changes.

4.1 Commands

To control the log mechanism, the `Print` macro command. is utilized. It allows to configure, start and stop the logging as to write log data.

Each print command have to be started with the keyword `[LOG]` and target is `Debug report`.





This set of commands is available for control:

Command	Description	Example
`create <handle>`	Creates a new logging object with the number given with `handle` and default configuration. If no number is given the section number is used.	create 1 (Create a new logging object with the number 1)
`assign <mode> <section> <handle>`	Add (mode = 1) or remove (mode = 0) a section to a logging object. `section` is the section number to be logged and `handle` is number of the logging object. If a parameter is empty the current section number is used and assigned.	• assign 1 1 (Add section 1 to the logging object with this section number) • assign 0 1 0 (Remove section 1 of the logging object with number 0) • assign 1 3 1 (Add section 3 to the logging object with number 1)
`config <key_word> <value> <handle>`	Set a configuration value of the logging object with the number given with `handle`. If no number is given the section number is used. See also Configuration parameters.	config log_format 1 1 (Set the logging format of the logging object 1 to 3rd party ASCII)
`start <handle>`	Start the storing of data in memory of logging object with the number given with `handle`. If no number is given the section number is used.	start 1 (Start data storage of logging object with the number 1)
`stop <handle>`	Stops the storing of data in memory of logging object with the number given with `handle`. If no number is given the section number is used. Stopping does not safe data to a file, for this a separate write is necessary!	stop 1 (Stop data storage of logging object with the number 1)
`write <filename> <handle>`	Writes the logged data of the logging object with the number given with `handle` by creating a file with `filename`. If no number is given the section number is used. If no filename is given one is created according to the settings.	• write (Write the data of the logging object with the section number to the file with the name from the configuration) • write Test.txt 0 (Write the data of the logging object 0 to a file names Test.txt)
`close <handle>`	Closes the logging object with the number given with `handle`. If no number is given the section number is used. If `write` was previously called all pending logging data is saved to the file, else all logging data is dismissed.	close 1 (Closes the logging object with the number 1)

Attention
Each argument must be written in a separate line.

Information
`Write` is not appending and should only be called before closing the logging object.

4.2 Configuration parameters

Now the following parameters can be added and set:

Argument	Default value	Used by Format	Description
config_name	Log config	All	Name of the configuration. Can be used to identify the log configuration.
log_format	1		Defines in which format the file will be written, see Log format
file_path	C:<username>\Documents\Lipowsky Industrie-Elektronik GmbH\Logs\	All	Path, where the log files will be created.
file_prefix	Log_	All	Fixed prefix for all log files. The maximum log size will be evaluated depending on this filename prefix.
file_postfix	%Y_%m_%d_%H_%M_%S	All	Adds a timestamp postfix to the filename. The timestamp postfix is only added, if this argument is not empty.
file_extension	log	All	Extension of the log file. A '.' will be added automatically between the filename and extension.
max_filesize	10MB	All	Maximum file size for a log file. If this file size is exceeded: • The current log file will be closed • A new log file will be created • The logging will continue in the new file
enable_frames	True	3rd party ASCII	Enable or disable the logging of frames.
enable_signals	False	3rd party ASCII	Enable or disable the logging of signals.
enable_errors	True	3rd party ASCII	Enable or disable the logging of errors messages.
enable_events	True	3rd party ASCII	Enable or disable the logging of events messages.
enable_debug	True	3rd party ASCII	Enable or disable the logging of debug messages.
enable_comments	True	3rd party ASCII	Enable or disable the logging of application comments.
enable_macrostates	False	3rd party ASCII	Enable or disable the logging of macro states.
data_line_format	%a %B %d %H:%M:%S %Y	3rd party ASCII	Adds a date line on top of the header. The line starts with "date ". The lines are only added, if this argument is not empty.

internal_events_ line_enabled	True	3rd party ASCII	Adds a line with the fixed text "internal events logged".
format_version	"8.0.0"	3rd party ASCII	Adds a line with a version number, e.g. "11.0.0". The line in the log file will then be: "version 11.0.0" The lines are only added, if this argument is not empty.
additional_header_text		3rd party ASCII	Adds a custom header text without the comment prefix. A new line can be startet with \n. The lines are only added, if this argument is not empty.
additional_header_comment		3rd party ASCII	This header comment will be added with the comment prefix. A new line can be startet with \n. The lines are only added, if this argument is not empty.
trigger_block_enabled	True	3rd party ASCII	Marks the complete log as the trigger block. Before the regular logging, this text is added with a timestamp: "Begin Triggerblock". After the regular logging, this text is added: "End TriggerBlock".
start_entry	"Start of measurement"	3rd party ASCII	Adds this text, at the beginning of the regular logging. The lines are only added, if this argument is not empty.
timestamp_decimal_places	6	3rd party ASCII	Number of decimal places used after the decimal point. If the number of decimal places of the timestamp is lower, zeros will be added.
timestamp_integer_places	4	3rd party ASCII	Number of integer places used before the decimal point. If the number of integer places of the timestamp is lower, space characters will be added.
align_timestamp_enabled	false	3rd party ASCII	Check to align the timestamp of comments and reports with the frame reports. 3 space characters will be added before the timestamp of a frame report.
preceding_timestamp_format	<>	3rd party ASCII	Adds an absolute timestamp before each report. The absolute timestamp is only added, if this format string is not empty.
min_frame_id_chars	15	3rd party ASCII	Number of characters the frame ID will at least use. If the the number of characters of the frame ID is lower, spaces (0x20) will be added.

d_column_enabled	False	3rd party ASCII	If enabled a column of d characters is added before the data length of frame reports.
checksum_enabled	True	3rd party ASCII	If enabled the checksum entry will be printed as decimal at the end.
frame_id_enabled	True	3rd party ASCII	If enabled the frame ID will be printed as decimal at the end.
extended_frame_ properties_enabled	False	3rd party ASCII	If enabled the extended frame properties like protocol will be added. This option break the compatibility with similar formats!
channel_mapping	L1	3rd party ASCII	Assigns a name to each channel in the log file. If more than one channel is named they have to separated by \n.

4.3 Workflow

This chapter explains a typical SDF logging workflow, for example a part should be tested.

4.3.1 Initialization

At first, a logger object has to be created. This logger object can be configured in many ways, especially the data that is logged, the file format and the mode when data is written.



Attention
Each argument must be written in a separate line.

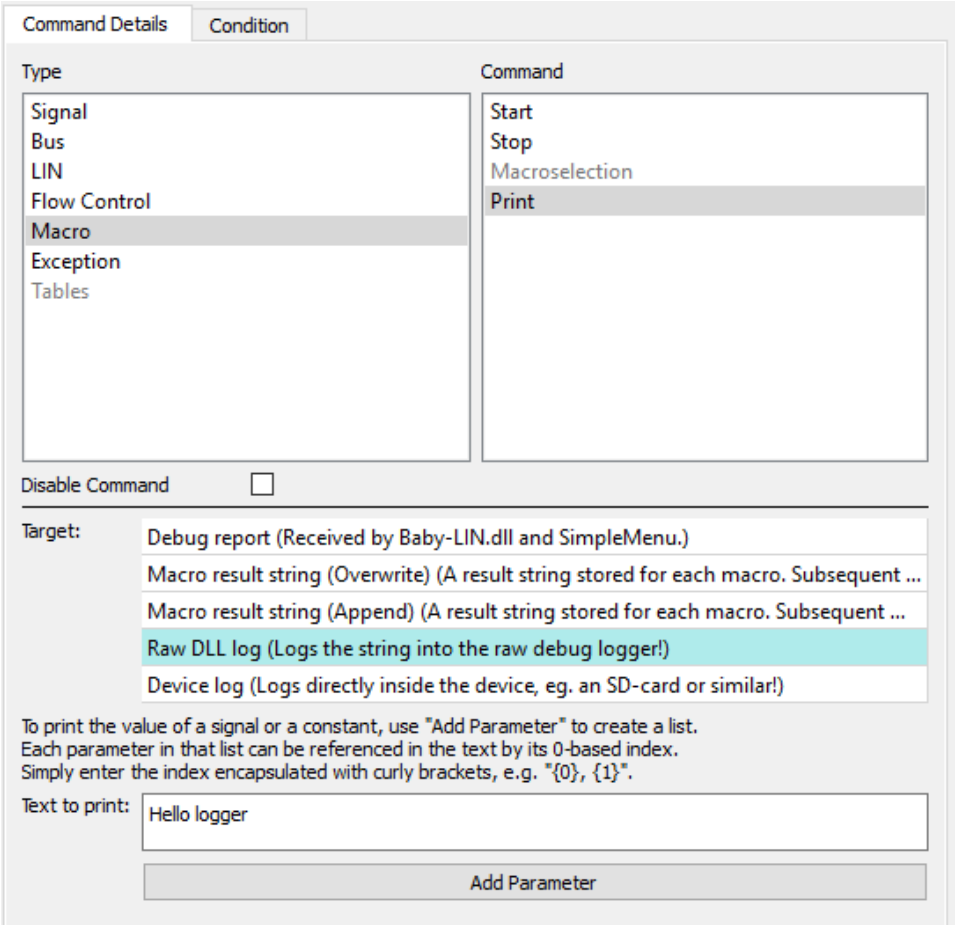
To create a logger add a macro print command and set Debug report as target. The following lines are needed:

Argument	Argument	Description
[LOG]	None	Keyword for the device to know this report will control the logging.
create	Optional handle number	A new logger object will be created.
assign	Optional handle and channel number	A channel is added to the logger. If no parameter is given the channel of the current section and the handle with the current section's number is used.
config	Add needed Configuration parameters and optional handle number	Configure the logger for your needs. If no configuration value is set for a parameter it's default value is used.
start	Optional handle number	Start the logging of data in memory.

4.4 Writing log data

If the 3rd party ASCII format is selected all selected data is written automatically, so no extra SDF commands are required.

For the Raw format data is written by hand. For this add a `Print` macro command and set `Raw DLL log` as target this time. Now all text added is saved by the logger. You can use all formatting option of the print command and also access the macro result string via `{R}`.



4.5 Finish logging

When all desired information was gathered the data has to be written to a file with `write`. After write was issued data can still be logged.

As last step, e.g. the end of a test run, the logging object has to be closed by `close`. If no `write` was issued before the close command is called all data will be discarded. This can be used for example to only log parts which didn't pass the test.

5 RAM logger

The RAM logger is the basic logging mechanism of the Logging . It is active by default and can be disabled with `firmware.logging.ram.disabled`.

Once booted the device will start to collect data and safe it to a section in the internal RAM.



Attention
The data is collected in RAM until the download is triggered by the user. If the device is switches off, all data in the RAM is lost.

5.1 Saved data and format

The log data is stored in the 3rd party ASCII format and the following information is logged:

- Bus frames
- Errors
- Events
- Debug information
- Extended frame properties

The files are saved as `ram_log_channel_<channel number>_%d-%m-%Y-%H-%M-%S.txt`

5.2 Storage size

The used RAM section to store the data is 40MB large. Once this section is full the oldest data is overwritten again.



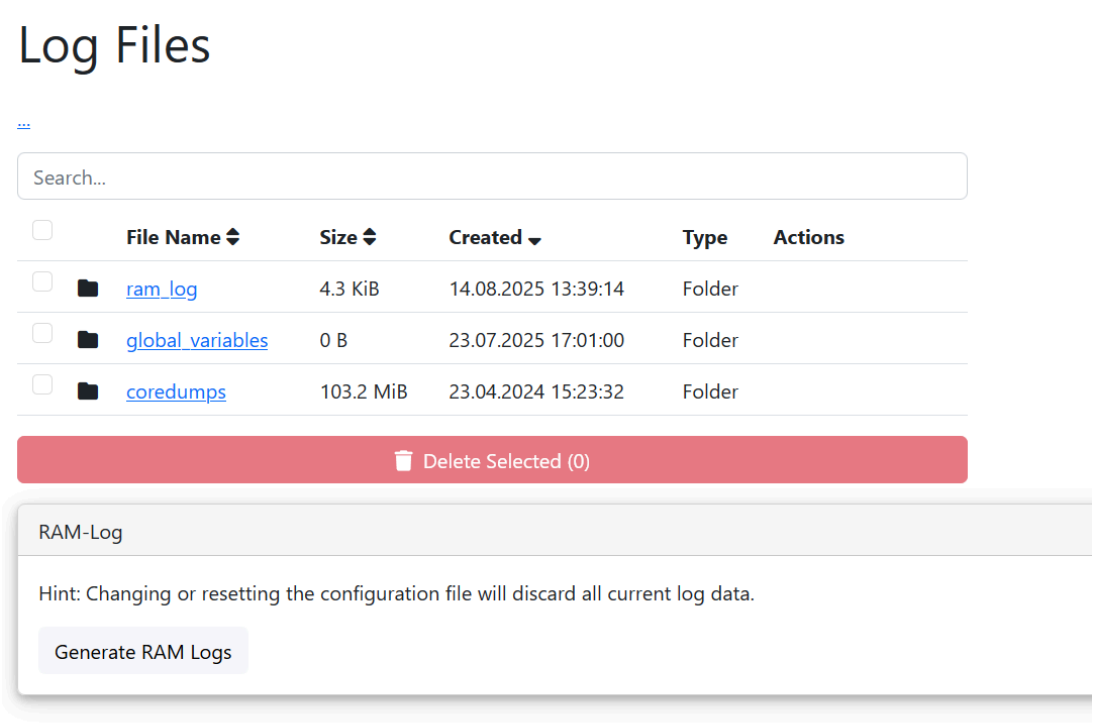
Attention
The time to fill the section depends on the number and type of active bus interfaces.

5.3 Download log data

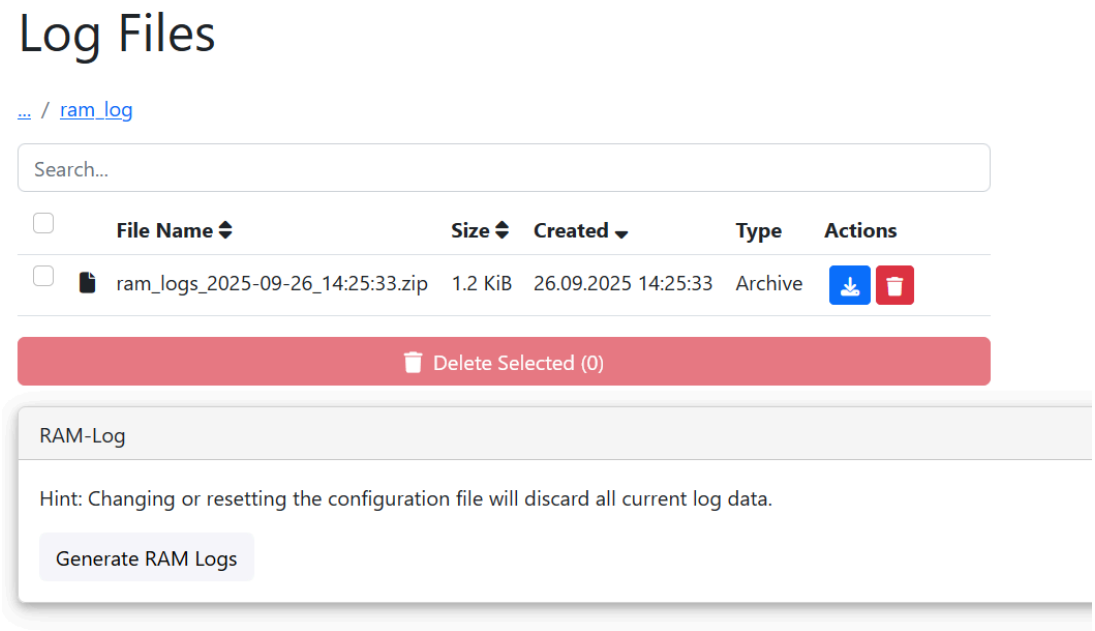
If all desired data is logged it has to be saved from RAM to an permanent flash drive.

5.3.1 Safe to internal memory

To trigger the download you can use the web interface by navigating to `Logging` and it's sub menu `Log files`. Click the `Generate RAM Logs` button, confirm the pop-up window and if generation is finished a new download window will open.



This will also save the log data to the internal microSD card from which they can be downloaded via the web interface anytime. The log files can be found in the sub-folder `ram_log`



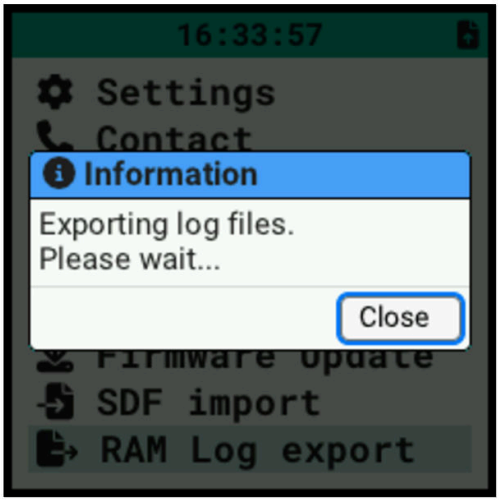
5.3.2 Safe to external memory

Also the RAM logs can be exported onto a connected USB flash drive via the device display in case the device has no network connection. Press the `Menu` button of the device and navigate down to `RAM Log export` in the `USB flash drive category`. Start the export by pressing the `OK` button.

Information
This menu point is only visible when a correct formatted USB drive is detected.

Attention
Check **??** for information about the port capabilities.





The folder in which the log files are stored is `BL-MB-01\log\`. The filename will contain the date, time and channel the log file was started on and follows the scheme described in Saved data and format.

The Logging creates one file for each channel and one global file. All filenames contain a timestamp.

5.3.3 Automatic download

Currently an automatic download or external trigger via TCP/IP is not available but is planed to be implemented in 2026.

6 microSD card Logging

Attention
The microSD card has to be formatted as FAT16 or FAT32.

Information
Unlike with the other logging options with the microSD card only one logger can be active at once.

6.1 Introduction

For devices with a microSD card the log data can also be stored as a file on the card. The folder in which the log files are stored is named `<Device Serial Number>\<Device Serial Number>-Logs\`. The filename will contain the device Serial number, the channel name and a continuous number in the format `<Device Serial Number>-<channel_name>_xxxxx.log`

6.2 SDF settings

There are a few system variables with which the logging on the microSD card is setup.

`@@SYSLOGFORMAT`

Like with the other logging options you can choose a file format between the listed formats in Log format. The selected format is specified by @@SYSLOGFORMAT and has the following values:

Format	Format number	Value of @@SYSLOGFORMAT
Binary format (Deprecated)	0	0 (Default)
3rd party ASCII	1	1
Raw format	6	5

Attention
@@SYSLOGFORMAT has to be present as virtual signal, even if the default value is used, otherwise the logging is not started.

`@@SYSLOGCONTROL`

Is used to control, e.g. start or stop, the logging process and has the following options:

Value	Write/Read	Description
0	Write	The logging will be stopped and the file on the card closed.
	Read	Logging is not active or waiting for a valid microSD card.
1	Write	Start the logging.
	Read	A logger is currently running.
2	Write	Pause the logging without closing the log file.
3	Read	The logger was started but is not running yet.

Warning
After writing 1 to @@SYSLOGCONTROL it takes some time (several 100ms, depending on the card size and content) until the logger is writing data. Poll @@SYSLOGCONTROL till you read 1 to insure the logger was started successfully and no data is lost.

@@SYSLOGFILESIZE

Furthermore you can specify how large a single file should be. This can be useful for longer logging periods to generate easier readable files. The size is set by @@SYSLOGFILESIZE and can vary between 1 and 4096 MBytes, where 4096 is the default value.
Once the file reaches this size it is automatically closed and a new file is started.

@@SYSCARDSIZE

Can used to read the card size e.g. to evaluate if the card size will fit the expected logging data.

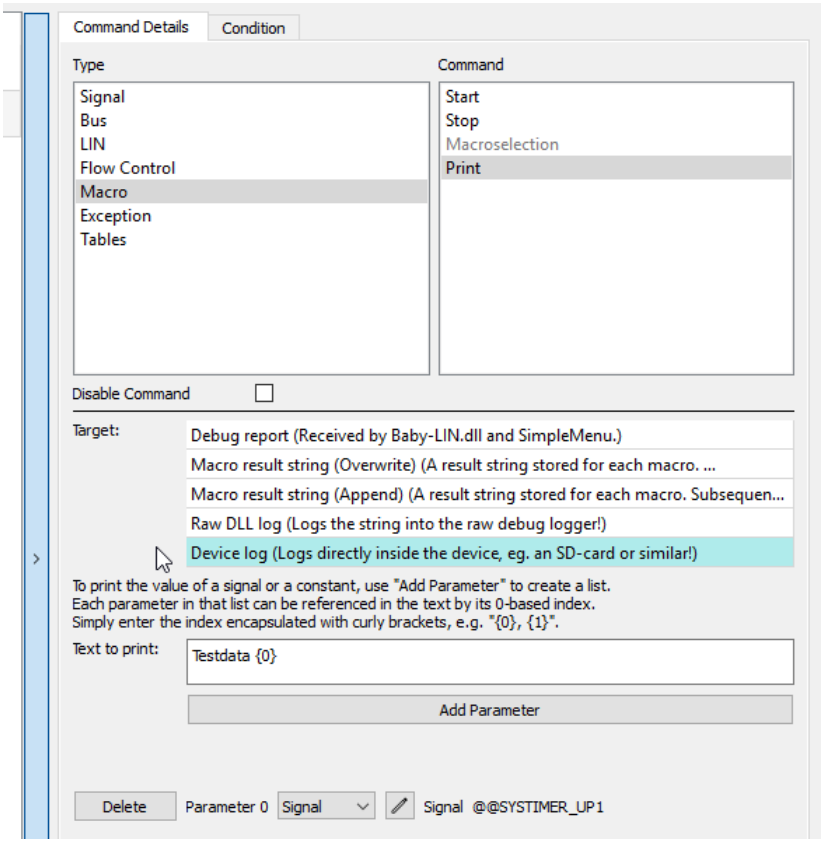
6.3 Using 3rd party ASCII format

If you want the save the data in 3rd party ASCII setup the format and start the logger.

Afterwards all data is saved automatically without user interaction with all the default options, which are exclusive for this format, mentioned here Configuration parameters.

6.4 Using raw format

If you want the save the data in Raw format instead you setup the format and start the logger as previously described. To write data add a `Print` macro command with `Device log` as target. You can use all formatting option of the print command and also access the macro result string via `{R}`.



7 Support information

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

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

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

Author	Lipowsky Industrie-Elektronik GmbH Römerstraße 57 64291 Darmstadt
Phone	+49 (0) 6151 / 93591 - 0
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 9001:2015