

DS4330 LIN Interface Board

# Features

Release 2021-A – May 2021

## How to Contact dSPACE

Mail:	dSPACE GmbH Rathenaustraße 26 33102 Paderborn Germany
Tel.:	+49 5251 1638-0
Fax:	+49 5251 16198-0
E-mail:	<a href="mailto:info@dspace.de">info@dspace.de</a>
Web:	<a href="http://www.dspace.com">http://www.dspace.com</a>

## How to Contact dSPACE Support

If you encounter a problem when using dSPACE products, contact your local dSPACE representative:

- Local dSPACE companies and distributors: <http://www.dspace.com/go/locations>
- For countries not listed, contact dSPACE GmbH in Paderborn, Germany.  
Tel.: +49 5251 1638-941 or e-mail: [support@dspace.de](mailto:support@dspace.de)

You can also use the support request form: <http://www.dspace.com/go/supportrequest>. If you are logged on to mydSPACE, you are automatically identified and do not need to add your contact details manually.

If possible, always provide the relevant dSPACE License ID or the serial number of the CmContainer in your support request.

## Software Updates and Patches

dSPACE strongly recommends that you download and install the most recent patches for your current dSPACE installation. Visit <http://www.dspace.com/go/patches> for software updates and patches.

## Important Notice

This publication contains proprietary information that is protected by copyright. All rights are reserved. The publication may be printed for personal or internal use provided all the proprietary markings are retained on all printed copies. In all other cases, the publication must not be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form, in whole or in part, without the prior written consent of dSPACE GmbH.

© 2002 - 2021 by:  
dSPACE GmbH  
Rathenaustraße 26  
33102 Paderborn  
Germany

This publication and the contents hereof are subject to change without notice.

AUTERA, ConfigurationDesk, ControlDesk, MicroAutoBox, MicroLabBox, SCALEXIO, SIMPHERA, SYNECT, SystemDesk, TargetLink and VEOS are registered trademarks of dSPACE GmbH in the United States or other countries, or both. Other brand names or product names are trademarks or registered trademarks of their respective companies or organizations.

# Contents

About This Document	5
Local Interconnect Network (LIN)	7
LIN Basics.....	7
Master/Slave Concept.....	8
Example of a LIN Bus.....	10
Introduction to the Features of the DS4330	11
DS4330 Architecture.....	12
Feature Overview.....	12
DS4330 Interfaces.....	13
Fields of Application	15
Remaining Bus Simulation.....	15
Testing Communication Timing Constraints.....	16
Testing Against Specification Limits.....	16
Detecting the Baud Rate of a LIN Bus.....	17
Testing Diagnostic and Failure Conditions.....	18
Testing Energy-Saving Modes.....	19
LIN Bus Handling	21
Setting Up a LIN Bus.....	22
Basics on LIN Bus Handling.....	22
Specifying the LIN Bus Parameters.....	23
Database Files.....	23
Defining LIN Nodes.....	24
Handling Frames.....	25
Transmitting and Receiving Frames.....	25
Manipulating Frames.....	26
Controlling a LIN Bus.....	27
Switching LIN Nodes.....	27
Setting a LIN Bus to Sleep Mode.....	28
Waking Up a LIN Bus.....	28

Changing LIN Bus Parameters.....	28
Reading Status Information.....	29
Handling Schedules.....	30
Setting Up a LIN Schedule.....	30
Setting the Priority of LIN Schedules.....	31
Using Interrupts.....	32
Defining Interrupts.....	32
 Using the RTI LIN MultiMessage Blockset.....	 33
Basics on the RTI LIN MultiMessage Blockset.....	33
Transmitting and Receiving LIN Frames.....	35
How to Define a Checksum Algorithm.....	38
Manipulating Signals to be Transmitted.....	39
 Limitations.....	 41
Limitations.....	41
Limitations of RTI LIN MultiMessage Blockset.....	41
 Index.....	 47









# About This Document

## Contents

This document provides feature-oriented access to the information you need to implement the functions of the DS4330.

## Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
	Indicates a hazard that, if not avoided, could result in property damage.
	Indicates important information that you should take into account to avoid malfunctions.
	Indicates tips that can make your work easier.
	Indicates a link that refers to a definition in the glossary, which you can find at the end of the document unless stated otherwise.
	Precedes the document title in a link that refers to another document.

## Naming conventions

dSPACE user documentation uses the following naming conventions:

**%name%** Names enclosed in percent signs refer to environment variables for file and path names.

**< >** Angle brackets contain wildcard characters or placeholders for variable file and path names, etc.

---

## Special folders

Some software products use the following special folders:

**Common Program Data folder** A standard folder for application-specific configuration data that is used by all users.

%PROGRAMDATA%\dSPACE\<InstallationGUID>\<ProductName>

or

%PROGRAMDATA%\dSPACE\<ProductName>\<VersionNumber>

**Documents folder** A standard folder for user-specific documents.

%USERPROFILE%\Documents\dSPACE\<ProductName>\<VersionNumber>

**Local Program Data folder** A standard folder for application-specific configuration data that is used by the current, non-roaming user.

%USERPROFILE%\AppData\Local\dSPACE\<InstallationGUID>\<ProductName>

---

## Accessing dSPACE Help and PDF Files


After you install and decrypt dSPACE software, the documentation for the installed products is available in dSPACE Help and as PDF files.

**dSPACE Help (local)** You can open your local installation of dSPACE Help:

- On its home page via Windows Start Menu
- On specific content using context-sensitive help via **F1**

**dSPACE Help (Web)** You can access the Web version of dSPACE Help at [www.dspace.com/go/help](http://www.dspace.com/go/help).

To access the Web version, you must have a *mydSPACE* account.

**PDF files** You can access PDF files via the  icon in dSPACE Help. The PDF opens on the first page.

# Local Interconnect Network (LIN)

## Where to go from here

## Information in this section

<a href="#">LIN Basics.....</a>	<a href="#">7</a>
LIN (local interconnect network) is a low-cost serial communication system for distributed electronic systems in cars.	
<a href="#">Master/Slave Concept.....</a>	<a href="#">8</a>
A LIN bus must have one master (called the LIN master) and may have several slaves (called LIN slaves).	
<a href="#">Example of a LIN Bus.....</a>	<a href="#">10</a>
Example of a typical network in a car.	

## LIN Basics

### Basics

LIN (local interconnect network) is a low-cost serial communication system for distributed electronic systems in cars. LIN is an open standard. It was developed to get a standard protocol to reduce the great variety of different low-end multiplex solutions, and cuts the cost of development, production, service, and logistics in car electronics. It is designed to connect electronic systems with low communication requirements, where the bandwidth and versatility of CAN are not required.

### Supported LIN specifications and SAE standards

- LIN specifications 1.2 and 1.3
- LIN specifications 2.0, 2.1 and 2.2

- SAE J2602 standard (protocol version: J2602\_1\_1.0, language version: J2602\_3\_1.0)

SAE J2602 is a vehicle LIN bus standard based on LIN 2.0 and defined by the Society of Automotive Engineers (SAE).

## Related topics

### Examples

[Example of a LIN Bus..... 10](#)

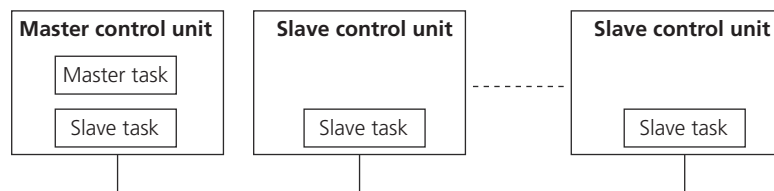
# Master/Slave Concept

## Introduction

A LIN bus consists of different nodes. It must have one master (called the LIN master) and may have several slaves (called LIN slaves). Each LIN master features one LIN master task and one LIN slave task. A LIN slave only features the LIN slave task.

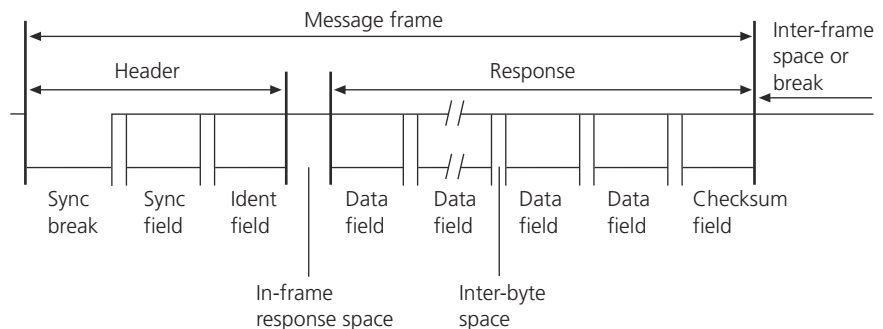
## LIN master

The LIN master task is responsible for schedules (lists of contiguous frames including their characteristics) and sends the respective LIN headers. The LIN slave tasks are responsible for transmitting or receiving the LIN response, see the following illustration.



## LIN messages

LIN messages are set up in a frame format. A frame is composed of a header, which is sent by the LIN master, and a response, which is sent by the LIN slave task, see the following illustration.





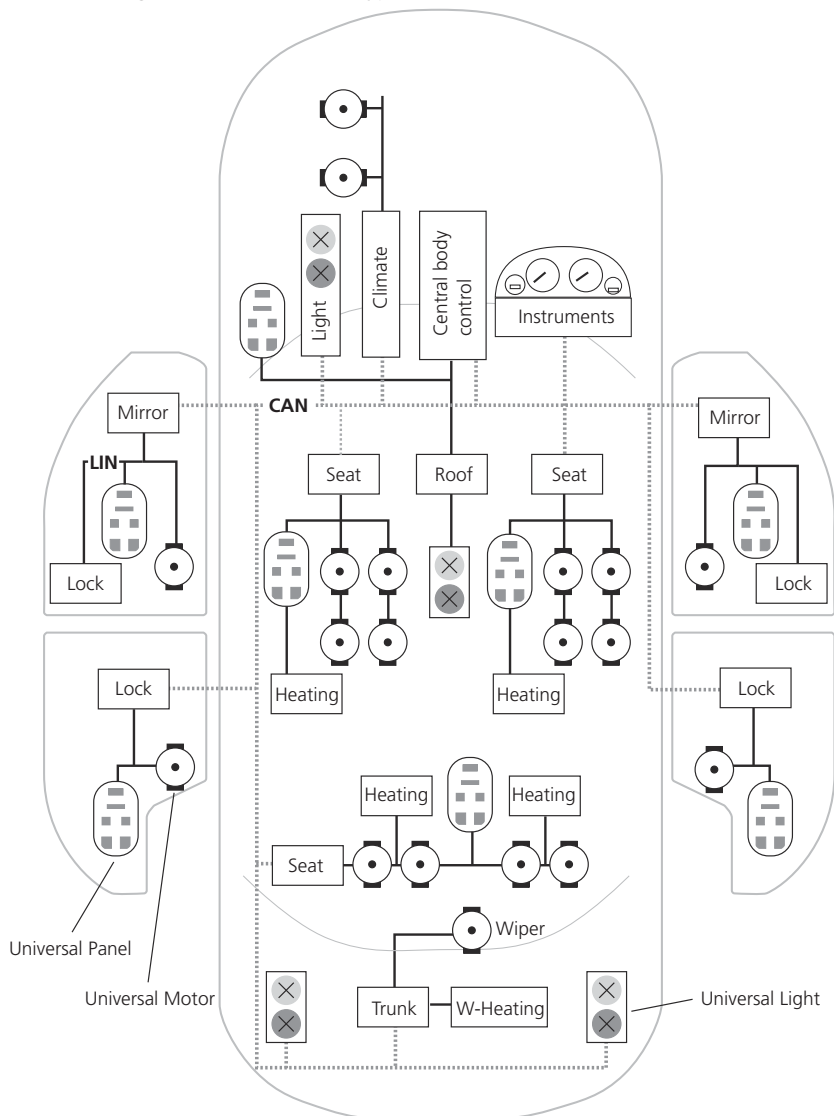
**Detailed information** For more information on the LIN bus protocol and the frame format, refer to the *LIN Protocol Specification*.

<b>Related topics</b>	<b>Basics</b>
	<a href="#">LIN Basics.....</a> 7

## Example of a LIN Bus

### Example

The following illustration shows a typical network in a car.



The CAN bus (dotted line) connects ECUs throughout the whole car. The LIN buses (solid lines) connect ECUs in local areas.

### Related topics

#### Basics

LIN Basics..... 7

# Introduction to the Features of the DS4330

---

## Introduction

The DS4330 LIN Interface Board connects a PHS-bus-based system to the LIN bus.

---

## Where to go from here

### Information in this section

[DS4330 Architecture](#)..... 12

It gives an overview of the functional units in a block diagram.

[Feature Overview](#)..... 12

It gives an overview of the features of the DS4330 LIN Interface Board.

[DS4330 Interfaces](#)..... 13

It describes the interfaces of the DS4330.

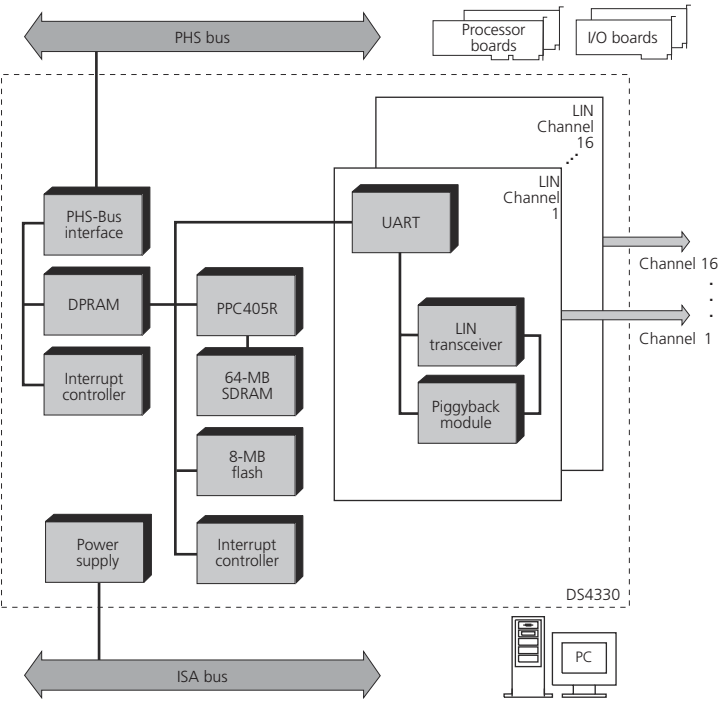
### Information in other sections

[DS4330 Data Sheet \(PHS Bus System Hardware Reference !\[\]\(b792654f2cef9719eabeb6c5be00811e\_img.jpg\)](#))

## DS4330 Architecture

### Functional units and architecture

The following illustration gives an overview of the functional units and architecture of the DS4330:



### Related topics

### References

<a href="#">DS4330 Interfaces</a>	13
<a href="#">Feature Overview</a>	12

## Feature Overview

### Introduction



The DS4330 LIN Interface Board provides the following features.

### LIN bus interface

Providing 16 independent LIN channels for the simulation of LIN nodes (see [DS4330 Interfaces](#) on page 13).

### Connection to external devices

The DS4330 is connected to the external devices via a 50-pin, male I/O connector.

<b>Hardware upgrade</b>	The built-in transceivers can be substituted by new LIN transceivers by using the piggyback module slots on the board. For more information, refer to <a href="#">Customization Modules (PHS Bus System Hardware Reference </a> ).
<b>Supported LIN specifications and SAE standards</b>	<ul style="list-style-type: none"> <li>▪ LIN specifications 1.2 and 1.3</li> <li>▪ LIN specifications 2.0, 2.1 and 2.2</li> <li>▪ SAE J2602 standard (protocol version: J2602_1_1.0, language version: J2602_3_1.0)</li> </ul> <p>SAE J2602 is a vehicle LIN bus standard based on LIN 2.0 and defined by the Society of Automotive Engineers (SAE).</p>
<b>LIN firmware</b>	Before using the RTI LIN MultiMessage Blockset, make sure the LIN firmware version coming with the same dSPACE installation as the blockset is installed on your DS4330. For information on how to update the firmware, refer to the <a href="#">Firmware Manager Manual </a> .
<b>Limitations</b>	There are some limitations when working with the DS4330, see <a href="#">Limitations</a> on page 41.
<b>Related topics</b>	<p>References</p> <div> <a href="#">DS4330 Architecture.....</a> 12 </div>

## DS4330 Interfaces

<b>Introduction</b>	<p>The DS4330 provides a LIN interface for the connection of dSPACE Simulator to LIN buses. It has the following characteristics:</p> <ul style="list-style-type: none"> <li>▪ 16 independent channels, each with the following features: <ul style="list-style-type: none"> <li>▪ Simulates one LIN bus</li> <li>▪ Baud rate up to 20 kBd (selectable via software within the range 1 ... 20 kBd)</li> <li>▪ LIN master or LIN slave termination (selectable via software)</li> <li>▪ LIN transceiver wake-up interrupt</li> </ul> </li> <li>▪ Piggyback modules to integrate alternative LIN transceiver types</li> </ul>
<b>RTI/RTLib support</b>	You can set up and control the simulated LIN bus with the RTI LIN MultiMessage Blockset and RTLib functions. Both, LIN slaves and masters can be simulated.

According to the LIN specification, up to 16 LIN slaves can be simulated. For details, see [LIN Bus Handling](#) on page 21.

**Integration into a PHS-bus-based system**

To be used, the DS4330 must be integrated into a PHS-bus-based system based on a dSPACE processor board. While the DS4330 provides the LIN interfaces, the dSPACE processor board takes over the calculation of the real-time model. That is, applications using DS4330 I/O features are implemented on the processor board.

Communication between processor board and I/O board is performed via the peripheral high speed bus: That is the PHS++ bus for a connection to a dSPACE processor board. In the documentation the term "PHS bus" is used.

**Partitioning the PHS bus with the DS802** With the DS802 PHS Link Board you can spatially partition the PHS bus by arranging the I/O boards in several expansion boxes.

The DS802 can be used in combination with many types of available dSPACE I/O boards. However, some I/O boards and some functionalities of specific I/O boards are not supported.

The I/O board support depends on the dSPACE software release which you use. For a list of supported I/O boards, refer to [DS802 Data Sheet \(PHS Bus System Hardware Reference\)](#).

**Related topics**

**References**

<a href="#">DS4330 Architecture</a> .....	12
<a href="#">Feature Overview</a> .....	12

# Fields of Application

## Introduction

Using dSPACE Simulator and a DS4330 LIN Interface Board, you can test the LIN network features of an electronic control unit (ECU). Several test scenarios are possible.

## Where to go from here

## Information in this section

<a href="#">Remaining Bus Simulation.....</a>	<a href="#">15</a>
Testing the transmission of message frames from the LIN master to simulated and real LIN slaves.	
<a href="#">Testing Communication Timing Constraints.....</a>	<a href="#">16</a>
Testing the time interval when the message frames are transmitted.	
<a href="#">Testing Against Specification Limits.....</a>	<a href="#">16</a>
Testing whether the ECU fulfills the requirements at the specification limits.	
<a href="#">Detecting the Baud Rate of a LIN Bus.....</a>	<a href="#">17</a>
Measuring the baud rate of a LIN bus.	
<a href="#">Testing Diagnostic and Failure Conditions.....</a>	<a href="#">18</a>
Testing the ECU in the case of failures.	
<a href="#">Testing Energy-Saving Modes.....</a>	<a href="#">19</a>
Testing the ECU in energy-saving modes.	

## Remaining Bus Simulation

## Introduction

An ECU with LIN master functionality can be tested using a dSPACE system with a DS4330 LIN Interface Board. The dSPACE system simulates the LIN nodes, which are connected to the LIN bus. The simulated bus members behave exactly

the same way as real bus members. This functionality is called “remaining bus simulation”.

---

**Real and simulated bus members**

For specific test purposes it is useful to replace a simulated bus member by the real bus member. Either a bus member is simulated by a dSPACE system or the real controller is connected to the bus. In order to minimize the number of real-time applications required for the simulation, it is possible to switch off a simulated bus member without compiling the real-time application again. Otherwise a dSPACE simulator used to test the complete car body electronics would require hundreds of different real-time applications.

---

**Related topics****Basics**

Testing Against Specification Limits.....	16
Testing Communication Timing Constraints.....	16
Testing Diagnostic and Failure Conditions.....	18
Testing Energy-Saving Modes.....	19

## Testing Communication Timing Constraints

---

**Testing communication timing constraints**

For control algorithms it is important that an ECU sends its message frame within a given time interval. During simulation the simulator can observe the transmitted message frames and thus check the timing constraints.

---

**Related topics****Basics**

Remaining Bus Simulation.....	15
Testing Against Specification Limits.....	16
Testing Diagnostic and Failure Conditions.....	18
Testing Energy-Saving Modes.....	19

## Testing Against Specification Limits

---

**Introduction**

An ECU has to work under a wide range of conditions. Suppose you specified your ECU to work correctly in a LIN network at 9.6 kBd  $\pm 10\%$ . In this case, the dSPACE simulator has to run the LIN bus at 8.64 kBd to 10.56 kBd.



**In-frame response time** A second important specification of the LIN network is the in-frame response time. This specifies the time between the end of the LIN header transmitted by the LIN master and the beginning of the LIN response transmitted by the LIN slave. Each LIN slave receiving the corresponding message frame and the LIN master must check whether the complete frame is completed within a given time interval. The RTI LIN MultiMessage Blockset can vary the in-frame response time in order to check whether the ECU behaves correctly under all conditions.

**Baud rate detection** The actual available baud rate for a LIN bus can be detected by the RTI LIN MultiMessage Blockset and RTLib functions. This is useful if a model is designed for various master nodes using different baud rates. If you specify the baud rate in an LDF file, this baud rate is fixed. A detailed description of the baud rate detection used follows.

## Related topics

### Basics

Remaining Bus Simulation.....	15
Testing Diagnostic and Failure Conditions.....	18
Testing Energy-Saving Modes.....	19

# Detecting the Baud Rate of a LIN Bus

**Introduction** Baud rate detection with the DS4330 LIN Interface Board works with two hardware units. This limits the number of channels on which the baud rate can be measured to two. Although you can start baud rate measurement on all channels, only two are processed. These are the first two channels on which a synchronization break signal is detected. They are selected automatically and measurement is started. Measurement for all following channels is discarded. You have to restart the measurement (function) if one of the channels is available again. Consider that the measured channels are not marked. They could be measured again, so it could take some time until all channels are measured.

**Detected signal** The detected signal must contain a synchronization field consisting of the pattern '0x55' (hex). The hardware measures the 5 falling edges of the bit pattern (1010 1010).

**Adjusted baud rate** For the channel on which the baud rate is measured the highest baud rate that is expected to occur must be adjusted.

**Baud rate too small**

If the baud rates are too small in relation to the adjusted baud rate they cannot be measured. Baud rate measurement is started when a synchronization break signal is detected. The break signal has a length of at least 10 low bit times. It is followed by a synchronization field. Depending on the baud rate, each low bit of a message is evaluated as a break signal, so measurement is started again and again.

**Used algorithm**

The measurement uses the following algorithm:

$$\text{baudrate\_config}/8 \leq \text{baudrate\_config} \leq \text{baudrate\_config} + 15\%$$

The algorithm describes the actual value range the hardware is able to measure. If the specified baud rate is too small the baud rate cannot be measured correctly. For example, if you configure 9600 baud as the baud rate, but you want to work alternatively with 19,200 baud, this would not be possible. The algorithm first uses the configured value to detect the break signal. A break signal sent with 19,200 baud is too small to be detected with 9600 baud configured hardware. Thus, you must enter the highest available baud rate in the LDF file to start the baud rate measurement with the highest baud rate.

**Related topics****Basics**

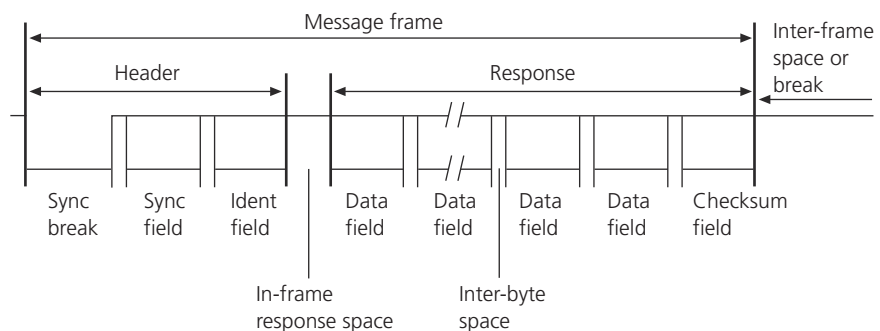
[Testing Against Specification Limits..... 16](#)

## Testing Diagnostic and Failure Conditions

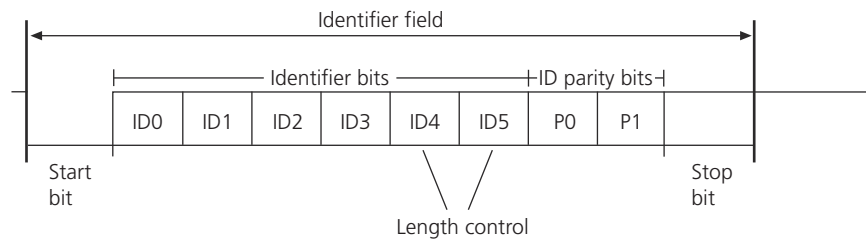
**Testing diagnostic and failure conditions**

Diagnostic and failure procedures are becoming more and more important within modern ECUs. To check the failure handling of an ECU, dSPACE Simulator can generate errors on the LIN bus.

The following illustration shows the format of a LIN message frame to show the errors which can be simulated.



The second illustration shows the format of the identifier field contained in the LIN message frame.



The following errors can be generated for test purposes.

**Inconsistent-synch-byte error** The synch byte is used by the ECUs to synchronize themselves to the LIN bus speed. The synch byte is transmitted as pattern 0x55 via the LIN bus. For error simulation a different pattern can be transmitted.

**Identifier-parity error** The identifier field consists of 6 bits plus 2 identifier parity bits. LIN slaves must not respond to incorrect identifiers or identifiers which are not of interest.

**Slave-not-responding error** After a LIN header is sent, the LIN slave must complete the frame within a given time interval. If this time interval is too long, the other LIN nodes must ignore the message. Bus collisions may occur if the LIN master starts the next LIN header before the LIN slave has finished its response.

**Checksum error** The response from the LIN slave is protected by a checksum field. If the checksum value is invalid, the LIN nodes ignore the message.

## Related topics

### Basics

Remaining Bus Simulation.....	15
Testing Against Specification Limits.....	16
Testing Communication Timing Constraints.....	16
Testing Energy-Saving Modes.....	19

## Testing Energy-Saving Modes

### Introduction

Body electronic control units are often connected to an unswitched power supply, because they must be operable even if the engine is not running: for example, a door lock unit. Therefore, these ECUs consume power even if the car is parked for days. Modern top-of-the-range cars may have more than 100 ECUs. To avoid discharging the battery, the ECUs must switch to low-power sleep mode when they are not in operation.

**Low-power sleep mode**

It is an important test scenario to check whether an ECU is correctly changing to low-power sleep mode and can be woken up again by several events. The dSPACE simulator can simulate and analyze these sleep and wake-up events. ECUs connected via buses can often be woken up and put into sleep mode by bus events.

In the case of LIN, the dSPACE simulator can react to the following events:

Event	Reaction
Sleep command received via the LIN bus	Put the LIN transceiver into sleep mode and inform the application that a sleep command has been received.
Wake-up command received via the LIN bus	Wake up the LIN transceiver and inform the application that a wake-up command has been received via the LIN bus.
Application wants to wake up the LIN bus	Wake up the LIN transceiver and send the wake-up command via the LIN bus.
Application wants to set the LIN bus asleep	Send the goto sleep command over the LIN bus and put the LIN transceiver into sleep mode.

**Related topics****Basics**

Remaining Bus Simulation.....	15
Testing Against Specification Limits.....	16
Testing Communication Timing Constraints.....	16
Testing Diagnostic and Failure Conditions.....	18

# LIN Bus Handling

**Introduction**                      The RTLib provides the C functions for implementing a LIN bus communication.

Where to go from here	Information in this section
	<a href="#">Setting Up a LIN Bus.....</a> 22 To transmit and receive frames you have to set up the LIN bus.
	<a href="#">Handling Frames.....</a> 25 During simulation you can handle frames that are interchanged between the LIN nodes.
	<a href="#">Controlling a LIN Bus.....</a> 27 The LIN bus can be controlled by the real-time application. You can switch between simulated and real nodes, or simulate energy-saving modes.
	<a href="#">Handling Schedules.....</a> 30 Schedules can be used to send frames according to predefined lists. Interrupts can be triggered for different schedule events.
	<a href="#">Using Interrupts.....</a> 32 Interrupts can be generated by bus events from a node, frame or schedule. The interrupts can be used to trigger interrupt-driven subsystems.

# Setting Up a LIN Bus

## Introduction

To transmit and receive frames you have to set up the LIN bus.

## Where to go from here

## Information in this section

<a href="#">Basics on LIN Bus Handling</a> .....	22
Provides basic information on how to simulate LIN master and LIN slaves.	
<a href="#">Specifying the LIN Bus Parameters</a> .....	23
As the first step you must set up the LIN bus in your model or application.	
<a href="#">Database Files</a> .....	23
Bus communication is specified in databases, which specify the information transmitted by an ECU, how it is coded and how often it is sent via the bus.	
<a href="#">Defining LIN Nodes</a> .....	24
After the LIN bus is set up, you can define all simulated LIN nodes.	

## Basics on LIN Bus Handling

### Number of LIN buses

The DS4330 LIN Interface Board provides 16 independent LIN channels. Therefore, you can simulate 16 LIN buses with LIN master and LIN slaves. LIN data from within a real-time model or application can be transmitted and received. LIN frames and LIN bus events as well as energy-saving scenarios such as wake-up and sleep mode are supported.

### Connection to the LIN bus



For detailed information on how to connect a DS4330 LIN Interface Board to a LIN bus, refer to [Mapping of I/O Signals \(PHS Bus System Hardware Reference !\[\]\(35dc653d59570f8f891c312eeece91a2\_img.jpg\)](#)).

## Related topics

## References

[Mapping of I/O Signals \(PHS Bus System Hardware Reference !\[\]\(aab88c0d099e5d18d6533a97b13ec28d\_img.jpg\)](#))

## Specifying the LIN Bus Parameters

<b>Introduction</b>	As the first step you must set up the LIN bus in your model or application.
<b>Configuration steps</b>	<p>If you want to set up the LIN bus, you have to perform some configuration steps.</p> <ol style="list-style-type: none"> <li>1. Set the parameters to select the LIN Interface Board, the LIN channel, the transceiver type the LIN bus is connected to and the termination resistance. For information on the pinouts, refer to <a href="#">Interface Connector (P1) (PHS Bus System Hardware Reference </a>).</li> <li>2. Use a database file in the LDF, DBC, FIBEX or AUTOSAR XML file format. One database file can be used for several LIN buses. For more information on these file formats, refer to <a href="#">Database Files</a> on page 23.</li> <li>3. Define a name for the LIN bus. This name must be unique in the model or application.</li> <li>4. If the baud rate is not specified in the database file, define the baud rate.</li> </ol>
<b>RTLib user</b>	Use the <code>dslin4330_board_init</code> , <code>dslin_channel_create</code> and <code>dslin_channel_init</code> functions. Define the necessary parameters in them. For detailed information on the functions, refer to <a href="#">LIN Channel Handling (DS4330 RTLib Reference </a> ).

## Database Files

<b>Introduction</b>	Specifying communication via buses in a car is a complex activity, because a bus consists of a lot of different ECUs, which may be developed by several departments. Therefore, bus communication is specified in databases, which specify the information transmitted by an ECU, how it is coded and how often it is sent via the bus.
<b>Supported database file types</b>	<p><b>LDF file</b> The LDF file format describes a complete LIN network and contains all the information necessary to configure it. The file format was specially developed for LIN networks, so it can describe all the features of a LIN network. For more information on the LDF file format, refer to the <i>LIN Configuration Language Specification</i>.</p> <p><b>DBC file</b> The DBC file format is designed by Vector Informatik GmbH for the CANalyzer database files. Although it was developed for CAN buses, it can also be used for LIN buses with some limitations:</p> <ul style="list-style-type: none"> <li>▪ As the CAN protocol has no standard definitions for master nodes and schedules, such definitions are not supported by the DBC files.</li> </ul>

- According to the LIN specification only byte layouts in the Intel format are supported. Byte layouts in Motorola format are not supported.

**FIBEX file** The Field Bus Exchange (FIBEX) format is an XML exchange file format. It is used for data exchange between different tools that work with message-oriented bus communication. A FIBEX file usually describes more than one bus system. You therefore have to select one of the available bus systems if you work with a FIBEX file as the database.

**AUTOSAR system description file** AUTOSAR system description files are XML files that describe a system according to AUTOSAR. A system is a combination of a hardware topology, a software architecture, a network communication, and information on the mappings between these elements. AUTOSAR system description files are instances of the AUTOSAR System Template. The AUTOSAR System Template contains a description of the network communication and hardware topology according to the FIBEX standard defined by ASAM e.V.

AUTOSAR system description files are files of AUTOSAR XML file type that can be used to export or exchange information on system descriptions.

For information on which AUTOSAR Releases are supported, refer to [Features of the RTI LIN MultiMessage Blockset \(RTI LIN MultiMessage Blockset Reference !\[\]\(96cc62f861fdd6e50510c0224a756dff\_img.jpg\)](#)).

## Defining LIN Nodes

### Introduction

After the LIN bus is set up, you can define all simulated LIN nodes.

### LIN node types

Two types of LIN nodes are available:

- The master node controls the LIN bus and protocol and controls which message has to be sent over the LIN bus at a specified time.
- The slave nodes receive frame headers from the master and send the corresponding frame over the LIN bus.

### Setting up a LIN node

The following steps have to be taken to setup a LIN node. You must define at least one LIN node for a LIN bus.

1. Specify the LIN bus to which the LIN node is connected.
2. Select the LIN node from the database file or define a user-defined node (only slave type possible). The name must be unique in the LIN bus.
3. Repeat the above steps for all simulated LIN nodes of the LIN bus.
4. Define the TX and RX error counters available for detecting errors that occur during the sending or receiving of frames.

### RTLib user

Use the `dslin_node_create` and `dslin_node_init` functions. Refer to [LIN Node Handling \(DS4330 RTLib Reference !\[\]\(4146d17f71dced09c6ad789cacceaa6d\_img.jpg\)](#)).



# Handling Frames

## Database file

The signals to be transferred via a LIN bus are composed as frames. The frame composition is described by the database file, which is selected during bus setup.

## Where to go from here

### Information in this section

#### [Transmitting and Receiving Frames](#)..... 25

After you have set up the LIN node, frames can be transmitted or received in the model or application.

#### [Manipulating Frames](#)..... 26

In some application fields it is necessary to access the frames before they are transmitted or decoded. For this purpose you can manipulate the frame data.

### Information in other sections

#### [Specifying the LIN Bus Parameters](#)..... 23

As the first step you must set up the LIN bus in your model or application.

# Transmitting and Receiving Frames

## Introduction

The signals are transmitted in frames. Therefore, the signals must be encoded before they are transmitted via the LIN bus. The rules for encoding are defined in the database file. In the same way, a received frame must be decoded first to get the signals.

## RTLib user

The RTLib for the DS4330 provides functions for transmitting and receiving frames. Functions to encode messages to frames or to decode frames to messages are not provided by RTLib. You have to program them yourself. Refer to [LIN Frame Handling \(DS4330 RTLib Reference !\[\]\(56549452e01ca28bdf2500ced9653143\_img.jpg\)](#)).

## Manipulating Frames

---

### Introduction

In some application fields it is necessary to access the frames before they are transmitted or decoded. For this purpose you can manipulate the frame data, for example, to simulate a transmission error.

---

### RTLib user

The RTLib for the DS4330 provides functions for transmitting and receiving frames. Functions to encode messages to frames or to decode frames to messages are not provided by RTLib. You have to program them yourself. Additionally, you also have to program the function for manipulating the frames. For information on the transfer functions, refer to [LIN Frame Handling \(DS4330 RTLib Reference !\[\]\(339a16584d5da0f0a3ca4e9ec17bf6a1\_img.jpg\)](#)).

# Controlling a LIN Bus

## Introduction

The LIN bus can be controlled by the real-time application. You can switch between simulated and real nodes, or simulate energy-saving modes.

## Where to go from here

## Information in this section

### [Switching LIN Nodes..... 27](#)

The LIN nodes connected to a LIN bus can be real or simulated in the model or application. You can enable and disable the simulation of a LIN node while the application is running.

### [Setting a LIN Bus to Sleep Mode..... 28](#)

Only LIN buses in sleep mode (energy saving mode) can be woken up.

### [Waking Up a LIN Bus..... 28](#)

An important feature of a LIN bus is the support of energy-saving modes. This means that it can be put to sleep and woken up after receiving a specific signal.

### [Changing LIN Bus Parameters..... 28](#)

To test the LIN bus under different operating conditions, some parameters must be changed during simulation.

### [Reading Status Information..... 29](#)

During simulation you can read the status information of LIN nodes.

## Switching LIN Nodes

## Introduction

If you want to replace a simulated LIN node by a real controller you have to disable the LIN node in the model or application. RTLib contains functions to disable or enable simulated LIN nodes during simulation. Therefore, it is not necessary to recompile the model or application even if the LIN bus structure changes.

### Note

Do not enable a simulated LIN node if the corresponding real LIN node is also connected to the LIN bus. Otherwise two LIN slaves would answer to the same header and cause errors.

**RTLib user** Use `dslin_node_enable` to enable a LIN slave and `dslin_node_disable` to disable a LIN slave. Refer to [LIN Node Handling \(DS4330 RTLib Reference !\[\]\(21199eb166cc97331a0c54c649195dcc\_img.jpg\)](#)).

## Setting a LIN Bus to Sleep Mode

**Introduction** For testing the energy-saving mode, the whole LIN bus can be set to sleep mode. The corresponding signal has to be sent by the LIN master node. For waking up the LIN bus, see [Waking Up a LIN Bus](#) on page 28.

**RTLib user** Use `dslin_node_command_sleep` to set a LIN bus to sleep mode. Refer to [dslin\\_node\\_command\\_sleep \(DS4330 RTLib Reference !\[\]\(e78f798d4ea5c530c9db49e7d26e6b95\_img.jpg\)](#)).

## Waking Up a LIN Bus

**Introduction** An important feature of a LIN bus is the support of energy-saving modes. This means that it can be put to sleep and woken up after receiving a specific signal. This frame can be generated by an RTLib function.

**RTLib user** Use `dslin_node_command_wakeup` to wake up a LIN bus. Refer to the [dslin\\_node\\_command\\_wakeup \(DS4330 RTLib Reference !\[\]\(ec9132f1d27c8919987d92907322654d\_img.jpg\)](#)).

### Related topics

#### Basics

<a href="#">Setting a LIN Bus to Sleep Mode.....</a>	<a href="#">28</a>
<a href="#">Testing Energy-Saving Modes.....</a>	<a href="#">19</a>

## Changing LIN Bus Parameters

**Introduction** To test if the ECU fulfills the requirements of the specification, you have to change some bus and frame parameters during the simulation (see [Testing Energy-Saving Modes](#) on page 19).

---

**RTLib user**

Use `dslin_channel_baudrate_set`, `dslin_channel_breaklength_set`, `dslin_channel_breakdelimiter_set`, `dslin_channel_synchfield_set` to change the parameters and `dslin_channel_apply_settings` to activate the new settings.

---

**Related topics****References**

[dslin\\_channel\\_apply\\_settings \(DS4330 RTLib Reference !\[\]\(74d4806277d7e73349d8e8c0897931e9\_img.jpg\)\)](#)  
[dslin\\_channel\\_baudrate\\_set \(DS4330 RTLib Reference !\[\]\(5f42d2cd7ad901bc24e5d35a38c777fd\_img.jpg\)\)](#)  
[dslin\\_channel\\_breakdelimiter\\_set \(DS4330 RTLib Reference !\[\]\(628bc0b1ef2b63d1fc4442fb794e3e78\_img.jpg\)\)](#)  
[dslin\\_channel\\_breaklength\\_set \(DS4330 RTLib Reference !\[\]\(210e01d0c2c300cf4405442bfd570b4e\_img.jpg\)\)](#)  
[dslin\\_channel\\_synchfield\\_set \(DS4330 RTLib Reference !\[\]\(78a7bc4d4f5b30b32890ad523045e9bf\_img.jpg\)\)](#)

## Reading Status Information

---

**Introduction**

Five different message error types are specified in the LIN specification (see *LIN Protocol Specification 1.2*). While the LIN nodes are running, the errors are counted. You can read the counters for each node type (master and slave) with RTLib functions.

---

**RTLib user**

Several functions are provided to get status information for both node types. Refer to [LIN Frame Handling \(DS4330 RTLib Reference !\[\]\(6bb0e4f14c4133b37d2887cb37e67ddd\_img.jpg\)\)](#).

# Handling Schedules

## Introduction

Schedules can be used to send frames according to predefined lists. Interrupts can be triggered for different schedule events.

## Where to go from here

## Information in this section

### [Setting Up a LIN Schedule.....30](#)

Currently executed LIN schedules can be interrupted and restarted or resumed. The behavior can be defined.

### [Setting the Priority of LIN Schedules.....31](#)

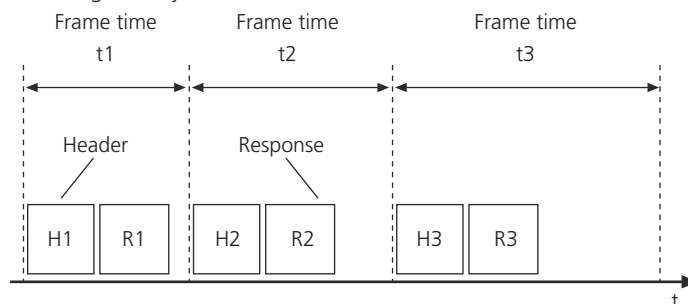
If you have defined several schedules in an LDF file, you can implement a decision logic to rank the schedules according to their priority. With RTLib you can start and stop a schedule with the corresponding functions. No ranking is required.

## Setting Up a LIN Schedule

### Basics on LIN schedules

LIN schedules have to be predefined in LDF files. The master node sends the listed headers and the corresponding nodes answer by sending the corresponding frame. Running schedules can be interrupted to start another schedule. After completing this schedule, the interrupted schedule can be restarted from the beginning or resumed where it was interrupted.

If you want to send several frames over the LIN bus in a specified time, you must use LIN schedules. RTLib provides functions to set up, start and stop LIN schedules. The following illustration shows a schedule with several frames. The frame time specifies the time to send the header and to get the response including a delay time.



The schedule timing is calculated by the DS4330. The times of the DS4330 and the processor board are synchronized to ensure that time stamps and the simulation time are consistent.

---

**RTLib user**

Several functions are provided to specify and handle LIN schedules. Refer to [LIN Schedule Handling \(DS4330 RTLib Reference !\[\]\(d84e7ea36f695d92cb39ec32c307ac93\_img.jpg\)](#)).

## Setting the Priority of LIN Schedules

---

**Introduction**

If you use several schedules you can define the priority of LIN schedules. Schedules with higher priority can interrupt schedules with lower priority.

---

**RTLib user**

With RTLib you can start and stop a schedule with the corresponding functions. Schedules cannot be ranked.

# Using Interrupts

## Defining Interrupts

### Introduction

To trigger a subsystem, nodes or frames can generate interrupts when different events occur.

### Events

The following events are defined for a node.

**Node events** The following events are defined for nodes:

- Bus wake-up
- Bus sleep request
- No bus activity
- RX error counter threshold exceeded
- TX error counter threshold exceeded
- Identifier-parity error
- Inconsistent-synch-field error

**Events for a received frame** The following events are defined for a received frame:

- Message received
- Checksum error
- Slave-not-responding error

**Events for a transmitted frame** The following events are defined for a transmitted frame:

- Header received
- Frame received
- Message transmitted
- Bit error

**Schedule events** The following events are defined for schedules:

- Schedule started
- Schedule terminated
- Schedule aborted
- Schedule restarted

For a description of the events, refer to the documentation of the interrupt features.

### RTLib user

Several functions are provided to implement node, frame and schedule interrupts. Refer to [LIN Interrupt Handling \(DS4330 RTLib Reference !\[\]\(9db214d549b9aeebe72aa11d3a5c4b1a\_img.jpg\)](#)).



# Using the RTI LIN MultiMessage Blockset

## Introduction

The RTI LIN MultiMessage Blockset is a Simulink blockset for efficient and dynamic handling of complex LIN setups in hardware-in-the-loop (HIL) applications.

## Where to go from here

## Information in this section

<a href="#">Basics on the RTI LIN MultiMessage Blockset.....</a>	<a href="#">33</a>
Provides an overview of the features of the RTI LIN MultiMessage Blockset.	
<a href="#">Transmitting and Receiving LIN Frames.....</a>	<a href="#">35</a>
Large LIN frame bundles can be managed from a single Simulink block provided by the RTI LIN MultiMessage Blockset.	
<a href="#">How to Define a Checksum Algorithm.....</a>	<a href="#">38</a>
You can specify your own checksum algorithm for frames.	
<a href="#">Manipulating Signals to be Transmitted.....</a>	<a href="#">39</a>
You can analyze signals of RX frames or change the values of signals of TX frames in the experiment software.	

## Basics on the RTI LIN MultiMessage Blockset

## Introduction

The RTI LIN MultiMessage Blockset is a Simulink blockset for efficient and dynamic handling of complex LIN setups in hardware-in-the-loop (HIL) applications. All the incoming RX frames and outgoing TX frames of an entire LIN controller can be controlled by a single Simulink block. LIN communication is configured via database files (DBC file format, LDF file format, FIBEX file format, MAT file format, or AUTOSAR XML file format).

**Supported dSPACE platforms**

The RTI LIN MultiMessage Blockset is supported by the following platforms:

- SCALEXIO systems with a DS2671 Bus Board, DS2672 Bus Module, DS6301 CAN/LIN Board, and/or DS6351 LIN Board
- PHS-bus-based systems (DS1006 or DS1007 modular systems) with a DS4330 LIN Interface Board
- MicroAutoBox II with LIN interface

You have to recreate all the RTI LIN MultiMessage blocks if you switch platform (for example, from SCALEXIO systems with a DS2671 Bus Board, DS2672 Bus Module, DS6301 CAN/LIN Board, and/or DS6351 LIN Board to a DS1007, or from a DS1007 to a MicroAutoBox II). To recreate all the RTILINMM blocks at once, select **Create S-function for All LIN Blocks** from the options menu of the GeneralSetup block (refer to [Options Menu \(RTILINMM GeneralSetup\) \(RTI LIN MultiMessage Blockset Reference\)](#))).

**Note**

The RTI LIN MultiMessage Blockset is not supported by the MicroAutoBox III. If you work with the MicroAutoBox III, you must use the Bus Manager to implement LIN bus simulation.

**Managing large LIN frame bundles**

With the RTI LIN MultiMessage Blockset, you can configure and control a large number of LIN frames from a single Simulink block. This reduces the size of model files and the time required for code generation and the build process.

**Manipulating signals to be transmitted**

The RTI LIN MultiMessage Blockset can manipulate the counter values of signals before they are transmitted.

When simulating with signal manipulation, you can specify whether to use the signal from the model or the TRC file. This is useful for simulating error values.

**Specifying signal saturation**

You can use the signal limits specified in the database file or specify other limits. If the input signal is outside of the specified range, it is set to the minimum or maximum limit. Refer to [Saturation Page \(RTILINMM MainSetup\) \(RTI LIN MultiMessage Blockset Reference\)](#)).

**Updating a model**

The RTI LIN MultiMessage Blockset creates an S-function for the specified database file. You can easily update the LIN configuration of a model by replacing the database file and updating the S-function.

**Modifying model parameters during run time**

Model parameters such as frames or signal values can be modified during run time either via model input or via the Bus Navigator in ControlDesk. For modifying model parameters via the Bus Navigator a variable description file

(TRC) is automatically generated each time you create an S-function for the RTILINMM MainSetup block. The entries of the TRC file let you analyze received signals, change the values of signals to be transmitted, etc. In ControlDesk, you can access the settings specified in the TRC file via the model's system description file (SDF). The SDF file bundles all the TRC files and additional information for the application.

(Relevant only for SCALEXIO systems with a DS2671 Bus Board, DS2672 Bus Module, DS6301 CAN/LIN Board, and/or DS6351 LIN Board) For information on where to find the signals of the LIN bus in the TRC file, refer to [Details on the Variable Groups of the Behavior Model \(ConfigurationDesk Real-Time Implementation Guide !\[\]\(c507f772dba2b921f86777f01218e570\_img.jpg\)](#)).

<b>Variables of custom code</b>	You can include variables of custom code in a <b>USR.TRC</b> file in addition to the parameters of the RTI LIN MultiMessage Blockset.
<b>TRC file entries with initial data</b>	TRC/SDF files generated for Simulink models including blocks from the RTI LIN MultiMessage Blockset contain initial data. The RTI LIN MultiMessage Blockset supplies all variables with initial values when they are included in the TRC file. TRC files with initial data allow you to perform offline calibration with ControlDesk.
<b>Visualization with the Bus Navigator</b>	The RTI LIN MultiMessage Blockset supports visualization with the Bus Navigator. Layouts/instruments are generated on demand and provide access to all the LIN signals and all the switches required to configure LIN communication during run time. You do not have to preconfigure layouts by hand.
<b>Monitoring and logging LIN bus communication</b>	The RTI LIN MultiMessage Blockset supports filtered and unfiltered monitoring and logging of LIN bus communication with the Bus Navigator. You can observe the raw and physical data of LIN frames on a LIN bus, and log the raw data of LIN frames. You can monitor and log LIN bus events.

## Transmitting and Receiving LIN Frames

<b>Introduction</b>	Large LIN frame bundles (up to 63 frames) can be managed from a single Simulink block provided by the RTI LIN MultiMessage Blockset.
<b>Defining LIN communication</b>	<p>To define the LIN communication of a LIN controller, you must provide a database file.</p> <p><b>LDF file as the database</b> You can use the LDF file format as the database for LIN communication. The LDF file format describes a complete LIN network and</p>

contains all the information necessary to configure it. The file format was specially developed for LIN networks, so it can describe all the features of a LIN network. For more information on the LDF file format, refer to the *LIN Configuration Language Specification*.

**DBC file as the database** You can also use the DBC file format as the database for LIN communication. The DBC file format was developed by Vector Informatik GmbH, Stuttgart, Germany. For the RTI LIN MultiMessage Blockset, you can use all the DBC files that pass the consistency check of Vector Informatik's CANdb++ Editor.

The DBC file format was developed for CAN buses, but it can also be used for LIN buses with some limitations:

- As the CAN protocol has no standard definitions for master nodes and schedules, such definitions are not supported by the DBC file formats.
- The LIN specification supports only byte layouts in the Intel format. Byte layouts in Motorola format are not supported.

**FIBEX file as the database** The Field Bus Exchange (FIBEX) format is an XML exchange file format. It is used for data exchange between different tools that work with message-oriented bus communication. A FIBEX file usually describes more than one bus system. You therefore have to select one of the available bus systems if you work with a FIBEX file as the database.

**MAT file as the database** You can also use the MAT file format as the database for LIN communication, or specify other database file formats as the database. You must convert your database files into the MAT file format for this purpose.

**AUTOSAR system description file as the database** You can also use AUTOSAR system description files as the database for LIN communication. AUTOSAR (AUTomotive Open System ARchitecture) is an industry partnership that aims to develop and establish an open standard for automotive electric/electronic (E/E) architectures.

AUTOSAR system description files are files of AUTOSAR XML file type that describe a system according to AUTOSAR. A system is a combination of a hardware topology, a software architecture, a network communication, and information on the mappings between these elements. AUTOSAR system description files are instances of the AUTOSAR System Template. The AUTOSAR System Template contains a description of the network communication and hardware topology according to the FIBEX standard defined by ASAM e.V.

---


## Defining RX frames and TX frames

You can receive and/or transmit each frame defined in the database file that you specify for LIN communication.


**Defining RX frames** You can define RX frames on the RX Frames Page (RTILINMM MainSetup).

**Defining TX frames** You can define TX frames on the TX Frames Page (RTILINMM MainSetup).

**Working with event-triggered frames**

The RTI LIN MultiMessage Blockset allows you to work with event-triggered frames. In contrast to a standard LIN frame, the frame header of an event-triggered frame is assigned to several frame responses of different LIN nodes. This allows you to transmit the frames of different nodes via the same frame slot. If you work with LIN 2.1 or later, you can specify collision resolver schedules to resolve collisions which might occur if two or more event-triggered frames are sent at the same time via the same frame slot. For details, refer to [Eventtriggered Frames Page \(RTILINMM MainSetup\)](#) (RTI LIN MultiMessage Blockset Reference ).

**Working with raw data**


The RTI LIN MultiMessage Blockset allows you to work with the raw data of frames. You have to select the frames for this purpose. The RTILINMM MainSetup block then provides the raw data of these frames to the model byte-wise for further manipulation. You can easily access the raw data of RX frames via a Simulink Bus Selector block. For details, refer to [Raw Data Page \(RTILINMM MainSetup\)](#) (RTI LIN MultiMessage Blockset Reference ).

**Implementing checksum algorithms**

In addition to the checksums of the LIN frames, you can use a signal to transmit a user-defined checksum. You can implement checksum algorithms for the checksum calculation of TX frames and checksum verification of RX frames.

**Checksum header file** You have to specify the checksum algorithms in a checksum header file. This needs to have a C-coded switch-case directive to switch between algorithms.

**Checksum calculation for TX frames** You can assign a checksum algorithm to each TX frame. A checksum is calculated according to the algorithm and assigned to the TX frame. Then the frame is transmitted together with the calculated checksum.

**Checksum check for RX frames** You can assign a checksum algorithm to each RX frame. A checksum is calculated for the frame and compared to the checksum in the received frame. If they differ, this is indicated at the error ports for RX frames if these ports are enabled. For details, refer to [User Checksum Definition Page \(RTILINMM MainSetup\)](#) (RTI LIN MultiMessage Blockset Reference ).

**LIN frame checksum manipulation** You can manipulate the checksums of LIN frames. The checksum is increased by a specified fixed offset value for a defined number of times.

**Receiving frames with a different frame length than specified in database file**

If the length of a received frame is different than that specified in the database file, how the frame is handled depends on its length.

**Received frame shorter than expected** If a frame is shorter than specified in the database file, it is not decoded. The error port is set to **Slave not responding** (RX\_Error = 16). The frame status remains 0. The RX\_Time is updated. The received data is output at the RX Raw Data port. Raw data bytes

which are not received are set to 0. You can read the frame length at the Frame Length port.

**Received frame longer than expected** If a frame is longer than specified in the database file, only the specified frame length is read. The next byte is interpreted as a checksum. A checksum error is therefore displayed in most cases. The raw data, RX\_Time and RX\_DeltaTime are updated. The error port is set to **Checksum Error (RX\_Error = 8)**. If the last byte is the correct checksum, the frame is decoded as usual.

## How to Define a Checksum Algorithm

### Objective


You can specify your own checksum algorithm for frames.

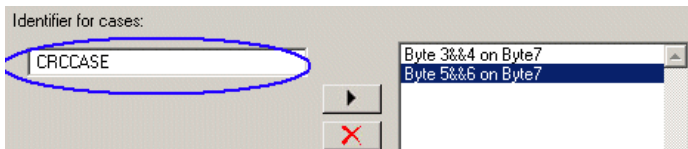
### Basics



You can implement checksum algorithms for frames via a *checksum header file*. Each algorithm must have a C-coded switch-case directive in the header file.

### Method


#### To define a checksum algorithm

- 1 Open the User Checksum Definition page.
- 2 Click  to specify an existing checksum header file, or enter a checksum header file name in the Header file edit field.
- 3 Name the checksum algorithms in the Identifier for cases field.



- 4 In the checksum header file, sort the algorithms.  
The order you specify for the algorithms corresponds to the `crctype` index of the switch-case directives in the header file.
- 5 Click  to create the header file if you entered its file name in the Header file edit field in step 1.
- 6 Click  to edit the file in MATLAB's M-File Editor.
- 7 In the checksum header file, edit your checksum algorithms. Note that the header file must contain at least one switch-case directive.  
You have access to the following input parameters of the header file:

Input Parameter	Description
<code>crcoption</code>	<ul style="list-style-type: none"> <li>▪ '0' if applied to a TX frame</li> </ul>

Input Parameter	Description
UInt8* FrameRAW_DATA	<ul style="list-style-type: none"> <li>▪ '1' if applied to an RX frame</li> </ul> See <a href="#">User Checksum Frames Page (RTILINMM MainSetup) (RTI LIN MultiMessage Blockset Reference </a> ).
crctype	Pointer to 8 bytes of raw data
CsBitPos	Index of the switch-case directives for the checksum algorithms. Corresponds to the order you specify in step 3 (see above).
CsLength	Start position of the checksum signal
MsgLength	Length of the checksum signal
MsgId	Frame length (in range 1 ... 8)
	Frame ID (1 ... 59)

**Result** Your checksum algorithm is used.

#### Related topics

#### References

[RTILINMM MainSetup \(RTI LIN MultiMessage Blockset Reference !\[\]\(83f22ed94ec5517769dd76d702c6bfd8\_img.jpg\)](#))  
[User Checksum Definition Page \(RTILINMM MainSetup\) \(RTI LIN MultiMessage Blockset Reference !\[\]\(58518edde73d42d67a35a8ed26134c7b\_img.jpg\)](#))

## Manipulating Signals to be Transmitted

### Introduction

All the signals of all the RX and TX frames (see [Transmitting and Receiving LIN Frames](#) on page 35) automatically get corresponding entries in the generated TRC file. This allows you to analyze them (signals of RX frames) or change their values (signals of TX frames) in the experiment software.


### Manipulating signals to be transmitted

The RTI LIN MultiMessage Blockset provides several options to manipulate the values of signals before they are transmitted. You can switch between the options you have specified via entries in the generated TRC file.

You can switch between these options in the experiment software.

**TX signals** The RTILINMM MainSetup block allows you to get the value of TX signals either from the Simulink model or from a variable in ControlDesk. The values of these TX signals can be changed from within the model. Their values cannot be changed in ControlDesk if the **Input Manipulation** option is cleared. Refer to [Input Manipulation Page \(RTILINMM MainSetup\) \(RTI LIN MultiMessage Blockset Reference !\[\]\(c444627dab9fee9a1550c053ffaaaae2\_img.jpg\)](#)).

**Counter signals** You can specify a counter signal that provides the number of frame transmissions. You can specify the counter start value, the increment,

and the counter stop value. Each time the counter reaches the stop value, it turns around to the counter start value. Refer to [Counter Page \(RTILINMM MainSetup\)](#) (RTI LIN MultiMessage Blockset Reference )

---

## Related topics

### Basics

[Transmitting and Receiving LIN Frames.....](#) 35



# Limitations

**Introduction** There are a few limitations to note when working with the DS4330 LIN Interface Board.

**Where to go from here** **Information in this section**

<a href="#">Limitations.....</a>	<a href="#">41</a>
Note the limitations when working with the DS4330.	
<a href="#">Limitations of RTI LIN MultiMessage Blockset.....</a>	<a href="#">41</a>
Limitations apply when you use the RTI LIN MultiMessage Blockset.	

## Limitations

**High-level protocols** The RTLib for the DS4330 support only software up to ISO level 2. This means that messages can be received or transmitted. High-level protocols, for example, KWP2000 or other diagnostic protocols, are not supported. You must program high-level protocols yourself.

**DBC files** DBC files do not support the master functionality, because the CAN bus specification does not provide masters and schedules.

## Limitations of RTI LIN MultiMessage Blockset

**Introduction** Limitations apply when you use the RTI LIN MultiMessage Blockset.

## RTI LIN MultiMessage Blockset

The following limitations apply to the RTI LIN MultiMessage Blockset:

- Do not use the RTI LIN MultiMessage Blockset in enabled subsystems, triggered subsystems and configurable subsystems. As an alternative, you can disable nodes or frames of the RTI LIN MultiMessage Blockset by setting the corresponding option.
- Do not run the RTI LIN MultiMessage Blockset in a separate task.
- Do not copy blocks of the RTI LIN MultiMessage Blockset. To add further blocks of the RTI LIN MultiMessage Blockset to a model, always take them directly from the `rtilinmm.lib` library.
- If you switch the platform from a SCALEXIO system to a non-SCALEXIO system or vice versa, you have to recreate all RTILINMM blocks. To recreate all RTILINMM blocks at once, select Create S-function for all LIN Blocks from the Options menu of the GeneralSetup block (refer to [Options Menu \(RTILINMM GeneralSetup\) \(RTI LIN MultiMessage Blockset Reference\)](#)). If you switch the platform from one non-SCALEXIO system to another (for example, from a DS1006 to a MicroAutoBox II) but the board and channel settings are not suitable for the new platform, you have to open the RTILINMM ControllerSetup blocks and recreate the S-functions for the blocks.
- RTILINMM ControllerSetup blocks that are added to the model but that are not assigned to RTILINMM MainSetup blocks (i.e., 'free floating' RTILINMM ControllerSetup blocks) do not initialize LIN channels of the real-time hardware. Therefore, these LIN channels cannot be accessed during run time via ControlDesk or Real-Time Testing (RTT) (e.g., for monitoring purposes).
- The RTI LIN MultiMessage Blockset is not included in the RTI update mechanism and is not updated when you open a model with an older version. To update the RTI LIN MultiMessage Blockset, invoke Create S-Function for all LIN Blocks from the Options menu of the RTILINMM GeneralSetup block (refer to [RTILINMM GeneralSetup \(RTI LIN MultiMessage Blockset Reference\)](#)).

As an alternative, you can create new S-functions for all RTILINMM blocks manually (use the following order):

1. RTILINMM GeneralSetup
2. RTILINMM ControllerSetup
3. RTILINMM MainSetup

- Model path names with multi-byte character encodings are not supported.
- The RTI LIN MultiMessage Blockset generates data structures on the basis of the relevant element names specified in the database file. The length of an element name is limited to 56 characters. If an element name exceeds this limit, the RTI LIN MultiMessage Blockset shortens the name to 56 characters, using a checksum to ensure name uniqueness, and makes an entry in the log file.

The following list shows the element types whose maximum name length must not exceed 56 characters:

- Frames
- Event-triggered frames

- Signals
- UpdateBit signals
- Nodes
- Schedules
- Schedule entries
- Simulink can store design data that your model uses in a data dictionary as a persistent repository. Data dictionaries are not supported by the RTI LIN MultiMessage Blockset.

**DBC file as the database**

The following limitations apply if your database file is in the DBC file format:

- As the CAN protocol has no standard definitions for master nodes and schedules, such definitions are not supported by the DBC files.
- According to the LIN specification only byte layouts in the Intel format are supported. Byte layouts in Motorola format are not supported.

**LDF file as the database**

The RTI LIN MultiMessage Blockset does not support multiple physical encodings in the LDF file. If several physical encoding types are defined for a signal, the RTI LIN MultiMessage Blockset uses the first type it finds for the signal.

**FIBEX file as the database**

The following limitations apply if you import a FIBEX file as the database:

- The RTI LIN MultiMessage Blockset does not support multiple computation methods for signals. If several CompuMethods are defined for a signal in the FIBEX file, the RTI LIN MultiMessage Blockset uses the first linear computation method it finds for the signal.
- You cannot specify the checksum calculation type individually for each LIN communication frame. The RTI LIN MultiMessage Blockset always uses specific checksum calculation types as follows:
  - As of LIN protocol version 2.0, the 'ENHANCED' checksum calculation type is used.
  - For LIN protocol versions < 2.0, the 'CLASSIC' checksum calculation type is used.
  - The 'CLASSIC' checksum calculation type is always used for the master request and slave response frames.

**MAT file as the database**

In the RTI LIN MultiMessage Blockset, the length of signal names is restricted to 32 characters. However, MATLAB allows longer signal names. When MATLAB entries are mapped to the signals in RTILINMM, the signal names are truncated at the end and supplemented by a consecutive number, if necessary. To ensure that unchanged signal names are used in the RTI LIN MultiMessage Blockset, the signal names in the Simulink model must not exceed 32 characters.

**AUTOSAR system description file as the database**

The following limitations apply if you import an AUTOSAR system description file as the database:

Limitation	Limitation Applies to AUTOSAR Release												
	3.1.4	3.2.1	3.2.2	4.0.3	4.1.1	4.1.2	4.2.1	4.2.2	4.3.0	4.3.1	4.4.0	Classic R19-11 <sup>1)</sup>	Classic R20-11 <sup>2)</sup>
Configuring selected slave nodes via RTI LIN MultiMessage Blockset is not possible due to the following limitations, which apply to AUTOSAR System Templates: <ul style="list-style-type: none"><li>In AUTOSAR, it is not possible to define collision resolver schedules for event-triggered frames. So if your database file is an AUTOSAR system description file, you have to manually assign a collision schedule to each event-triggered frame on the Collision Resolver Page (RTILINMM MainSetup).</li><li>The following initial node configuration parameters for slave nodes are not provided by AUTOSAR system description files:<ul style="list-style-type: none"><li>Supplier ID</li><li>Function ID</li><li>Variant ID</li><li>Initial NAD</li></ul>For the first three parameters, no identifier values are displayed on the Network Node Identification Page (RTILINMM MainSetup). However, you can specify an initial node address (NAD). By default, the configured NAD is used. If you do not specify an initial NAD, the selected node is interpreted as an unconfigured slave node.</li><li>AUTOSAR does not support the assignment of lists of configurable frames to network nodes. Thus, you cannot select configurable frames to specify an initial frame ID.</li></ul>	✓	✓	✓	–	–	–	–	–	–	–	–	–	–
When importing an AUTOSAR system description file as the database, the RTI LIN MultiMessage Blockset supports the computation method from source value to physical representation (converted value) only.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
The RTI LIN MultiMessage Blockset does not support the following features that can be defined in an AUTOSAR system description file: <ul style="list-style-type: none"><li>End-to-end communication protection (E2E protection)</li><li>Unit groups</li></ul>	– <sup>3)</sup>	–	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
When working with an AUTOSAR ECU Extract as the database, the RTI LIN MultiMessage Blockset does not support the import of an AUTOSAR ECU Extract that describes a slave node.	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

<sup>1)</sup> AUTOSAR Classic Platform Release R19-11

<sup>2)</sup> AUTOSAR Classic Platform Release R20-11

<sup>3)</sup> E2E protection is not supported by this AUTOSAR Release.

- The RTI LIN MultiMessage Blockset does not support the new features of AUTOSAR Release 4.4.0 and AUTOSAR Classic Platform Release R19-11 and R20-11.

---

#### Limitations with LIN specifications 2.x

- The RTI LIN MultiMessage Blockset does not support sporadic frames.
- The RTI LIN MultiMessage Blockset does not support dynamic frames. (Dynamic frames are provided by the LIN 2.0 specification only.)
- With a LIN node configuration compliant with LIN 2.0 and later, the RTI LIN MultiMessage Blockset supports the slave node configuration only. You can implement master functionality for a node configuration in the Simulink model, for example, by using raw data access for the master request frame. Refer to [Network Node Configuration Page \(RTILINMM MainSetup\) \(RTI LIN MultiMessage Blockset Reference\)](#).
- The RTI LIN MultiMessage Blockset does not support assign NAD frames.
- The RTI LIN MultiMessage Blockset does not support assign frame ID range frames.
- The RTI LIN MultiMessage Blockset does not support conditional change NAD frames.

---

#### Limitations with SAE J2602 support

The following limitations apply to J2602 support by the RTI LIN MultiMessage Blockset:

- The RTI LIN MultiMessage Blockset supports the SAE J2602 standard. It expects LIN protocol version J2602\_1\_1.0 and LIN language version J2602\_3\_1.0. If you use an LDF or AUTOSAR system description file with a different J2602 protocol version or language version (the version must also start with "J2602"), the RTI LIN MultiMessage Blockset continues working but this may lead to unpredictable behavior. A warning message is generated in the log file.
- The RTI LIN MultiMessage Blockset does not support sporadic frames.
- With a LIN node configuration compliant with J2602, the RTI LIN MultiMessage Blockset supports the slave node configuration according to LIN 2.0 only. You can implement master functionality for a node configuration in the Simulink model, for example, by using raw data access for the master request frame. Refer to [Network Node Configuration Page \(RTILINMM MainSetup\) \(RTI LIN MultiMessage Blockset Reference\)](#).
- When you import a J2602-compliant LDF or AUTOSAR system description file, the following optional master parameters are ignored by the RTI LIN MultiMessage Blockset:
  - max\_header\_length
  - response\_tolerance
- When you import a J2602-compliant LDF or AUTOSAR system description file, the following optional node attributes are ignored by the RTI LIN MultiMessage Blockset:
  - response\_tolerance
  - wakeup\_time
  - poweron\_time
- Compliance with J2602 response tolerances cannot be ensured.

- The maximum number of supported schedule table entries is 255.
- In contrast to the SAE J2602 specification, the unused bits of a frame are transmitted as dominant bits (0), not as recessive bits.
- Every clearly identified signal may be contained only once in a frame.
- The RTI LIN MultiMessage Blockset does not abort header transmission if an error occurs.
- The RTI LIN MultiMessage Blockset cannot access certain error states (ID parity error, framing error, bit error). There is no possibility to react to these errors within the model.
- There is no automatic parameterization of the J2602 status byte.
- The RTI LIN MultiMessage Blockset does not automatically support the Targeted Reset.
- The RTI LIN MultiMessage Blockset does not support the J2602 broadcast reset functionality.
- The RTI LIN MultiMessage Blockset does not support wake-up timings as specified in the J2602 standard. There is no automatic generation of wake up request sequences. If necessary, you can implement wake up request sequences in the Simulink model by using wake up requests which are supported by the RTI LIN MultiMessage Blockset.
- According to the J2602 specification, a sleep mode that lasts for at least four seconds is interpreted as a bus error. This error is indicated by the J2602 status byte. The RTI LIN MultiMessage Blockset does not interpret a sleep mode of this length as an error.
- According to the J2602 specification, a baud rate accuracy of  $\pm 0.5\%$  is required. For the MicroAutoBox II, this condition is not met for the standard base baud rate of 10417 bit/s. The closest available baud rate of 10472 bit/s (which corresponds to a variance of +0.53%) is used instead.

### Visualization with the Bus Navigator

The current version of the RTI LIN MultiMessage Blockset supports visualization with the Bus Navigator in ControlDesk 4.2.1 or later. You cannot work with earlier versions of ControlDesk in connection with applications created with the current version of the RTI LIN MultiMessage Blockset.

### Related topics

### References

[Collision Resolver Page \(RTILINMM MainSetup\) \(RTI LIN MultiMessage Blockset Reference !\[\]\(5a132f13505a6571904d622757b7a8f0\_img.jpg\)\)](#)  
[Network Node Identification Page \(RTILINMM MainSetup\) \(RTI LIN MultiMessage Blockset Reference !\[\]\(0f17417dd77a61b2fdbff69a33adf9f2\_img.jpg\)\)](#)

**A**

AUTOSAR system description file 24  
AUTOSAR System Template 24

**B**

basics of LIN bus 7  
basics on RTI LIN MultiMessage Blockset 33  
baud rate detection 17

**C**

checksum algorithm  
  defining 38  
  implementing 37  
checksum error 19  
Common Program Data folder 6

**D**

database files 23  
DBC file 23  
defining  
  interrupts 32  
  LIN frame 25  
  LIN master 24  
  LIN slave 24  
defining LIN communication 35  
defining RX and TX frames 36  
Documents folder 6  
DS4330  
  functional units 12  
  limitations 41  
DS802  
  partitioning PHS bus 14

**E**

energy-saving modes 19  
example of LIN bus 10

**F**

FIBEX file 24  
frames  
  manipulating 26  
  receiving 25  
  transmitting 25  
functional units of the DS4330 12

**I**

identifier-parity error 19  
implementing checksum algorithm 37  
inconsistent-synch-byte error 19  
in-frame response time 17  
interrupts  
  defining 32

**L**

LDF file 23

limitations  
  DS4330 41  
limitations of RTI LIN MultiMessage Blockset 41  
LIN  
  schedules 30  
  sleep mode 28  
LIN bus  
  connection 22  
  controlling 27  
  handling 21  
  setting up 22  
  waking up 28  
LIN bus basics 7  
LIN bus parameters  
  changing 28  
  specifying 23  
LIN buses  
  number 22  
LIN communication  
  defining 35  
LIN frame  
  defining 25  
LIN frames  
  receiving 35  
  transmitting 35  
LIN master 8  
LIN nodes  
  switching 27  
LIN slave  
  defining 24  
LIN specifications  
  supported 7, 13  
Local Program Data folder 6

**M**

manipulating  
  frames 26  
manipulating TX signals 39  
message frame 8

**P**

partitioning PHS bus with DS802 14

**R**

raw data  
  working with 37  
reading  
  status information 29  
receiving frames 25  
remaining bus simulation 15  
RTI LIN MultiMessage Blockset  
  basics 33  
  limitations 41  
RTILINMM  
  checksum header file 38  
RX frames  
  defining 36

**S**

SAE standards  
  supported 7, 13  
schedule  
  LIN 30  
  priority 31  
  setting up 30  
setting up  
  LIN bus 22  
slave-not-responding error 19  
sleep mode  
  LIN bus 28  
status information  
  reading 29  
switching  
  LIN nodes 27

**T**

transmitting frames 25  
TX frames  
  defining 36  
TX signals  
  manipulating 39

**W**

waking up a LIN bus 28  
working with raw data 37

