DS4330 LIN Interface Board

RTLib Reference

Release 2021-A – May 2021

dSPACE

How to Contact dSPACE

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

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

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

Contents

About This Document

Contents

This reference gives detailed descriptions of the C functions needed to program a DS4330 LIN Interface Board. The C functions can be used to program RTI-specific Simulink S-functions, or to implement your real-time models manually using C programs.

Symbols

dSPACE user documentation uses the following symbols:

Symbol	Description
DANGER	Indicates a hazardous situation that, if not avoided, will result in death or serious injury.
WARNING	Indicates a hazardous situation that, if not avoided, could result in death or serious injury.
CAUTION	Indicates a hazardous situation that, if not avoided, could result in minor or moderate injury.
NOTICE	Indicates a hazard that, if not avoided, could result in property damage.
Note	Indicates important information that you should take into account to avoid malfunctions.
Tip	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.

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.

LIN Basics

Introduction

Different functions and definitions are used to program the DS4330 LIN Interface Board.

Where to go from here

Information in this section

Information in other sections

Using the DSLIN API

Introduction

Several conventions apply when the DSLIN API is used to implement handcoded applications.

Where to go from here

Information in this section

Information in other sections

Supported LIN Specification

LIN specification 1.2 and 1.3

LIN specification 1.2 and 1.3 are supported. In this document, you will find many references to LIN specification 1.3, LIN specification 1.3 is the standard specification that defines a general LIN bus.

LIN specification 2.0

LIN specification 2.0 is partially supported. The RTLib supports the LIN slave functionality for the DS4330.

Naming of the DSLIN API

Function names

The naming of functions is closely related to the structure for which they are used. For example, functions concerning the channel handling contain "channel" in the name.

Object naming

Objects such as channels or nodes have user-definable names which are assigned with a create function.

Example

```
dslin_frame_p lin_frame_rx = NULL;
// Create the handle for the LIN rx frame.
dslin_frame_create( lin_node, "LIN FrameRX", &lin_frame_rx );
```

Instancing and Initializing an Object

Introduction

Objects can be instanced only dynamically. Each object should be initialized with NULL to avoid random values.

Example

The example shows how to instance a node and create a handle to it.

```
// Pointer to a LIN node.
dslin_node_p lin_node = NULL;
// Create the handle for the LIN node.
dslin_node_create( lin_channel, "LIN NodeMaster", &lin_node );
```

Error Handling

Introduction

Errors occurring during function execution are returned via the DSLIN_ERROR enumeration. If no error occurs, DSLIN_OK is returned. For detailed information, refer to LIN Error Handling on page 26.

Errors can be reported to the dSPACE log file with the corresponding `dslin_xxxx_error_print` functions. Each object (board, channel, node and frame) provides its own `dslin_xxxx_error_print` function. The dSPACE log file can be opened in the dSPACE experiment software.

Example

The following example shows how to use a `dslin_xxxx_error_print` function.

```
enum DSLIN_ERROR error;
error = dslin_node_init( lin_node, DSLIN_NODE_MASTER, 64, 64);
dslin_node_error_print( lin_node, error );
```

Object Methods

Introduction

The settings for each object are made with the corresponding `dslin_xxxx_set` functions. The settings have to be applied by the corresponding `dslin_xxxx_apply_settings` function. You can call the `dslin_xxxx_apply_settings` function once, after all the settings functions are executed. This transfers the data at once to the LIN board.

The current value of many settings made for an object can be read by the corresponding `dslin_xxxx_get` function.

Example

The following example shows how to update response data.

```
// Update the response data.
error = dslin_frame_tx_data_set( lin_frame_tx, 8, data );
// Transfer the response data to the LIN board.
error = dslin_frame_apply_settings( lin_frame_tx );
```

Standard Defines

Introduction

Different data fields and enumerations are predefined for the LIN API.

Where to go from here

Information in this section

Predefined Symbols

Introduction

The following tables list the defines, which are predefined in the `dslin.h` include file.

Channel defines

Symbol	Value	Meaning
`DSLIN_CHANNEL_BAUDRATE_DEFAULT`	9600	Default baud rate in bit times (after `dslin_channel_create` is executed, in bit/s)
`DSLIN_CHANNEL_BAUDRATE_MIN`	500	Minimum allowed baud rate for a LIN channel (in bit/s)
`DSLIN_CHANNEL_BAUDRATE_MAX`	22000	Maximum allowed baud rate for a LIN channel (in bit/s)
`DSLIN_CHANNEL_BAUDRATE_STEPSIZE`	10	Minimum step size for the baud rate (in bit/s)
`DSLIN_CHANNEL_BREAKLENGTH_DEFAULT`	14	Default break length in bit times

Symbol	Value	Meaning
DSLIN_CHANNEL_BREAKLENGTH_MIN	1	Minimum break length in bit times. 13 bit times is recommended by LIN specification 1.2.
DSLIN_CHANNEL_BREAKLENGTH_MAX	128	Maximum break length in bit times
DSLIN_CHANNEL_BREAKDELIMITER_DEFAULT	10	Default break delimiter in bit times
DSLIN_CHANNEL_BREAKDELIMITER_MIN	1	Minimum break delimiter in bit times
DSLIN_CHANNEL_BREAKDELIMITER_MAX	128	Maximum break delimiter in bit times
DSLIN_CHANNEL_BUS_IDLE_TIMEOUT_DEFAULT	4	Default bus idle timeout in seconds
DSLIN_CHANNEL_COMMAND_FRAME_IDENTIFIER	0x3C	Specifies the identifier for a master request frame that is used to send commands and data from a master to a slave node.
DSLIN_CHANNEL_COUNT_MAX	256	Maximum number of channels per processor board
DSLIN_CHANNEL_IDENTIFIER_MASK	0x3F	Mask that limits the number of identifiers
DSLIN_CHANNEL_SYNCHFIELD_DEFAULT	0x55	Default synchronization field of each frame header
DSLIN_CHANNEL_TIMEOUT_AFTER_WAKEUP_DEFAULT	128	Default value for timeout after the wake-up command was sent.

Node defines

Symbol	Value	Meaning
DSLIN_NODE_COUNT_MAX	16	Maximum number of nodes installed on a LIN channel
DSLIN_MAX_IDENTIFIER	64	Maximum number of identifiers available for a LIN bus

Frame defines

Symbol	Value	Meaning
DSLIN_FRAME_DEFAULT_RESPONSE_DELAY	64	Default response delay (in seconds)
DSLIN_MAX_FRAME_RESPONSE_DELAY	10.0	Maximum delay for a transmit response (in seconds)
DSLIN_MAX_FRAME_LENGTH	255	Maximum length of a frame (in byte)

Schedule defines

Symbol	Value	Meaning
DSLIN_SCHEDULE_MAX_ENTRIES	64	Maximum number of schedule steps (frame headers) in a schedule
DSLIN_SCHEDULE_COUNT_MAX	32	Maximum number of schedules that can be implemented on a LIN master node
DSLIN_SCHEDULE_MIN_FRAME_TIME	0.0025	Minimum frame time in seconds.
DSLIN_SCHEDULE_MAX_FRAME_TIME	2.0	Maximum frame time in seconds.

Common Defines

Symbol	Value	Meaning
DSLIN_BOARD_INTERRUPT_COUNT_MAX	2048	Maximum number of interrupts supported for a LIN board
DSLIN_OBJ_MAX_NAME_LENGTH	64	Maximum number of characters an object's name can have (including the terminating "0")

Data Types and Enumerations

Data types

The following data types are predefined:

- The type definition declares a reference to a generic LIN board. The pointer is returned by the `dslin4330_board_init` function:

```
typedef dslin_board_t* dslin_board_p
```

- The type definition declares a reference to a LIN node. The pointer is returned by the `dslin_node_init` function:

```
typedef dslin_node_t* dslin_node_p
```

- The type definition declares a reference to a LIN channel. The pointer is returned by the `dslin_channel_create` function:

```
typedef struct dslin_channel_t* dslin_channel_p
```

- The type definition declares a reference to a LIN frame. The pointer is returned by the `dslin_frame_create` function:

```
typedef struct dslin_frame_t* dslin_frame_p
```

- The type definition declares a reference to a LIN schedule. The pointer is returned by the `dslin_schedule_create` function:

```
typedef struct dslin_schedule_t* dslin_schedule_p
```

Enumerations

Several enumerations are predefined. The following predefined symbols must be used:

enum DSLIN_TRANSCEIVER_TYPE Enumeration to define the supported transceivers (currently, only the ISO9141):

Predefined Symbol	Meaning
DSLIN_TRANSCEIVER_ISO9141	Standard LIN transceiver

enum DSLIN_NODE_TERMINATION_TYPE Enumeration to define the termination type of a LIN channel:

Predefined Symbol	Meaning
DSLIN_TERMINATION_MASTER_1K	Termination for a master node
DSLIN_TERMINATION_SLAVE_30K	Termination for a slave node

If the master node is an external, not simulated node, the termination of that channel has to be set to DSLIN_TERMINATION_MASTER_1K. The simulated LIN slave nodes are then terminated with 30 kΩ. If the master is also simulated by the application the termination has to be set to DSLIN_TERMINATION_SLAVE_30K.

The difference between master and slave is that in master configuration the LIN bus has an external 1 kΩ pull-up resistor to the battery voltage and in slave configuration it does not. The pull-up resistor can be enabled/disabled for each channel.

Note

After power-up, all 16 LIN transceivers are configured as slaves.

enum DSLIN_CHANNEL_TX_MODE Enumeration to define a bitfield for the TX mode of the response. This enumeration is used in conjunction with the `dslin_channel_tx_response_write` function.

Predefined Symbol	Meaning
`DSLIN_CHANNEL_TX_ENHANCED_CHECKSUM`	If this bit is set, the extended checksum mode is used for the frame.
`DSLIN_CHANNEL_TX_RESPONSE_DELAY`	If this bit is set, the `response_delay` from `dslin_channel_tx_data_t` is used. See Data Structures: dslin_channel_tx_data_t on page 22.
`DSLIN_CHANNEL_TX_USER_CHECKSUM`	If this bit is set, the internal checksum calculation is deactivated and the external `checksum` from `dslin_channel_tx_data_t` is used. See Data Structures: dslin_channel_tx_data_t on page 22.
`DSLIN_CHANNEL_TX_ONCE`	If this bit is set, the frame is sent only once. If you want to send the frame again, refresh the data with `dslin_channel_tx_response_write`.

enum DSLIN_CHANNEL_RX_STATUS Enumeration to define the receive status of the receive monitor.

Predefined Symbol	Meaning
`DSLIN_CHANNEL_RX_CHECKSUM_ERR`	A checksum error was detected in the response.
`DSLIN_CHANNEL_RX_DATA_LOST_ERR`	Data lost error. Note: This is not a bus error. One or more LIN frame responses were overwritten, because the receive monitor was read too slowly.
`DSLIN_CHANNEL_RX_FRAMING_ERR`	A framing error was detected in the response.
`DSLIN_CHANNEL_RX_SNR_ERR`	A slave not responding error was detected.

enum DSLIN_FRAME_MODE Enumeration for defining the frame mode. This enumeration is used in conjunction with the `dslin_frame_mode_set` function.

Predefined Symbol	Meaning
`DSLIN_FRAME_MODE_CLASSIC_CHECKSUM`	The default checksum mode for LIN frames.
`DSLIN_FRAME_MODE_ENHANCED_CHECKSUM`	The checksum mode for LIN 2.0 frames.

Predefined Symbol	Meaning
DSLIN_FRAME_MODE_EVENT_TRIGGERED	Note: Related to LIN2.0 protocol. Event-triggered frame: ▪ Bit errors are not reported to the node. ▪ The data of an event-triggered frame is only sent once if no error occurs during the transmission of the frame response. ▪ If a bit error occurs, the frame is sent again the next time the LIN master request it. Note: When this mode is set for an identifier, all the TX frames with this identifier installed on the same LIN channel use this mode.
DSLIN_FRAME_MODE_UNCONDITIONAL	The default for a TX or RX frame.
DSLIN_FRAME_MODE_TX_ONCE	By default a TX frame is transferred every time the LIN master requests the frame response. This behavior can be disabled for a TX frame. After the mode is activated, the frame will only be sent if new data is available for the frame or the previous transfer of the frame has failed (bit error in reading back the own transmission). Note: When this mode is set for an identifier all the TX frames with this identifier installed on the same LIN channel use this mode.
DSLIN_FRAME_MODE_TX_ALWAYS	The frame response data is always sent when a matching LIN master node request is received. This is the default for a TX frame.

enum DSLIN_NODE_TYPE Enumeration for defining the node type:

Predefined Symbol	Meaning
DSLIN_NODE_SLAVE	Defines a slave node.
DSLIN_NODE_MASTER	Defines a master node. Only the master is allowed to transmit a frame header.

enum NODE_INTERRUPT_TYPE Enumeration to define the various node-related events that trigger an interrupt:

Predefined Symbol	Meaning
DSLIN_NODE_INT_IDPAR_ERROR	Triggers an interrupt by an ID parity error indicating that a slave node receives a wrong parity in a frame header.
DSLIN_NODE_INT_SYNCH_FIELD_ERROR	Triggers an interrupt by an inconsistent synchronization field error indicating that a slave task received a synchronization field different from 0x55 in a frame header.
DSLIN_NODE_INT_NO_BUS_ACTIVITY	Triggers an interrupt by a no-bus-activity condition. No synchronization break was received for more than 25000 bit times since the reception of the last synchronization break.
DSLIN_NODE_INT_EXTRABYTES_DETECTED	Triggers an interrupt when extrabytes occur. Bytes occurring on the LIN bus which cannot be assigned to a certain LIN header or LIN response are called extrabytes. The bytes occur if the receive length of an RX frame is shorter than the length of the TX frame with the same identifier on the same LIN bus, for example.

Predefined Symbol	Meaning
DSLIN_NODE_INT_SLEEP_CMD_RECEIVED	Triggers an interrupt after receiving the sleep command from the LIN master. The sleep command is a master command frame with 0x00 as the first data field. For more information, refer to LIN specification 1.2.
DSLIN_NODE_INT_WAKE_CMD_RECEIVED	Triggers an interrupt if a LIN node is in sleep mode and receives the wake-up command from any LIN node.
DSLIN_NODE_INT_WAKE_CMD_EXECUTED	Triggers an interrupt after the reception of a correct wake-up sequence (walk-up byte and header).
DSLIN_NODE_INT_TX_ERROR_THRESHOLD_EXCEEDED	Triggers an interrupt if an overflow of the transmit error counter occurs. The transmit error counter is only available with a master node. The TX error threshold can be set with the `dslin_node_init` and the `dslin_node_tx_error_threshold_set` functions.
DSLIN_NODE_INT_RX_ERROR_THRESHOLD_EXCEEDED	Triggers an interrupt if an overflow of the receive error counter occurs. The receive error counter is available for master and slave nodes. The RX error threshold can be set with the `dslin_node_init` and the `dslin_node_tx_error_threshold_set` functions.
DSLIN_NODE_INT_TIMEOUT_AFTER_WAKEUP	Triggers an interrupt if a timeout after a wake-up command was received. The timeout occurs if no header was received within 128 bit times after a wake-up command.

enum FRAME_INTERRUPT_TYPE Enumeration to define the various frame-related events that trigger an interrupt:

Predefined Symbol	Meaning
DSLIN_FRAME_INT_HEADER_RECEIVED	Indicates that a header was received correctly.
DSLIN_FRAME_INT_HEADER_SEND_BIT_ERROR	Indicates that the header was transmitted and a bit error was detected. Only for a master node.
DSLIN_FRAME_INT_RESPONSE_RECEIVED	Indicates that a response was received correctly.
DSLIN_FRAME_INT_RESPONSE_SEND	Indicates that a response was sent correctly.
DSLIN_FRAME_INT_RESPONSE_SEND_BIT_ERROR	Indicates that the response was transmitted and a bit error was detected.
DSLIN_FRAME_INT_RESPONSE_CHECKSUM_ERROR	Indicates that there is a checksum error in a received RX frame.
DSLIN_FRAME_INT_SNR_ERROR	Indicates that a slave-not-responding error has occurred. A response was not fully completed within the maximum frame length.
DSLIN_FRAME_INT_HEADER_SEND	Indicates that a header was sent correctly. Only for a master node.

enum SCHEDULE_INTERRUPT_TYPE Enumeration to define the various frame-related events that trigger an interrupt:

Predefined Symbol	Meaning
DSLIN_SCHEDULE_INT_STARTED	Indicates that a schedule was started.
DSLIN_SCHEDULE_INT_COMPLETED	Indicates that a schedule was executed successfully.

Predefined Symbol	Meaning
DSLIN_SCHEDULE_INT_ABORTED	Indicates that a schedule was interrupted before completion. Occurs if the schedule is breakable and is aborted by another schedule. Refer to dslin_schedule_breakable on page 172.
DSLIN_SCHEDULE_INT_RESTARTED	Indicates that an interrupted schedule was restarted. Occurs if the schedule is breakable (see dslin_schedule_breakable on page 172), resume is enabled (see dslin_schedule_resume_enable on page 168) or a restart offset different "1" is set (see dslin_schedule_restart_at on page 171).

Related topics

References

Data Structures: dslin_channel_rx_data_t

Purpose

To get information on the received frame response.

Instances of this data struct can be filled by the `dslin_channel_rx_monitor_client_read` receive monitor function. The received frame response can be evaluated by accessing the data members of the structure.

Syntax

```
typedef struct
{
  UInt8 identifier;
  UInt8 dlc;
  UInt8 rx_status;
  UInt8 checksum;
  dsfloat timestamp;
  UInt8 data[8];
}dslin_channel_rx_data_t
```

Include file

`dslin.h`

Members

identifier Frame identifier in the range 0x00 … 0x3F

dlc Reponse data length

rx_status Bitfield to return the receive status or errors. The bits are defined in the DSLIN_CHANNEL_RX_STATUS enumeration (see Data Types and Enumerations on page 17).

checksum The received checksum of the response.

timestamp Time stamp from the receive event.

data[8] The response data.

Related topics

References

Data Structures: dslin_channel_tx_data_t

Purpose

To define a TX frame response.

Instances of this data struct are used to define a TX response by the `dslin_channel_tx_response_write` receive monitor function.

Syntax

```
typedef struct
{
  UInt8 identifier;
  UInt8 dlc;
  UInt8 tx_mode;
  UInt8 checksum;
  dsfloat response_delay;
  UInt8 data;
}dslin_channel_tx_data_t
```

Include file

`dslin.h`

Members

identifier Frame identifier in the range 0x00 … 0x3F

dlc Response data length in the range 1 … 8

tx_mode Bitfield defining the TX mode of the response. This parameter is provided by the DSLIN_CHANNEL_TX_MODE enumeration (see Data Types and Enumerations on page 17).

If the parameter is set to "0", the response uses the following defaults:

- Classic checksum mode (Use the `DSLIN_CHANNEL_TX_ENHANCED_CHECKSUM` flag to change the value.)
- Response delay 0.0 seconds (Use the `DSLIN_CHANNEL_TX_RESPONSE_DELAY` flag to change the value.)
- Internal checksum calculation (Use the `DSLIN_CHANNEL_TX_USER_CHECKSUM` flag to change the value.)
- Always TX when the LIN master requests the response. (Use the `DSLIN_CHANNEL_TX_ONCE` flag to change the value.)

checksum The user provided checksum is used when `dslin_channel_tx_data_t::tx_mode = DSLIN_CHANNEL_TX_USER_CHECKSUM.`

response_delay The response delay in seconds when `dslin_channel_tx_data_t::tx_mode = DSLIN_CHANNEL_TX_RESPONSE_DELAY.`

data Response data

Related topics

References

dslin_channel_tx_response_write....89
LIN Error Handling....26
Standard Defines....15

Data Structures: dslin_frame_status_t

Purpose

To get information about the current status of a frame.

The status of a received or transmitted frame can be evaluated with the `dslin_frame_info_status_get` function. It returns a pointer to the `dslin_frame_status_t` structure.

Syntax

```
typedef struct
{
  unsigned error;
  unsigned error_bit;
  unsigned error_snr;
  unsigned error_checksum;
  unsigned error_framing;
  unsigned unused;
}dslin_frame_status_t;
```

Include file

`dslin.h`

Members

error Returns "1" if any error occurred during the last processing of the frame, otherwise '0'. The indicated error is unspecific. Use the following members of the structure to get detailed information on the error.

error_bit Returns "1" if a bit error occurred during the transmitting of the response data. The error is detected by a tx frame if the monitored bit value is different from the sent bit value.

error_snr Returns "1" if an error occurred. A slave-not-responding error is detected while waiting for a response from an rx frame.

error_checksum Returns "1" if an error occurred. The checksum error indicates an error in the checksum field of a received response (rx frame).

error_framing Returns "1" if a framing error in one byte of the response (rx frame) occurred. A framing error is an indication that a bus collision happend.

unused Reserved for future use.

Example

```
// This example shows the evaluation of the error
// for a tx frame. This is mainly only the bit error.
dslin_frame_status_t status;
// Get the status and error information.
error = dslin_frame_info_status_get( tx_frame, &status );
// If data was available, evaluate the data.
if( DSLIN_OK == error )
{
// At least one error has occurred.
    if( status.error )
    {
// Test for a bit error.
     if( status.error_bit )
     {
// Error Handling.
     }
    }
}
```

Related topics

References

dslin_frame_info_status_get.....219
LIN Error Handling.....26

Data Structures: dslin_schedule_status_t

Purpose

To get information about the current status of a schedule.

The status of the running schedule can be evaluated with the `dslin_schedule_status_get` function. It returns a pointer to the `dslin_schedule_status_t` structure.

Syntax

```
typedef struct
{
  unsigned active;
  unsigned pending;
  unsigned complete;
  unsigned aborted;
  unsigned unused;
}dslin_schedule_status_t
```

Include file

`dslin.h`

Members

active Returns "1" if the schedule is the currently active schedule. Only one schedule can be active per LIN channel.

pending Returns "1" if the schedule is pending. The schedule is not active and waits to be executed.

complete Returns "1" if the schedule was completely executed.

aborted Returns "1" if the schedule was interrupted by another schedule before it was completed.

unused Reserved for future use.

Related topics

References

Data Structures: dslin_frame_status_t.....23
dslin_schedule_status_get.....175

LIN Error Handling

Introduction

You can monitor several errors that can occur in sending or receiving data.

Where to go from here

Information in this section

Basics on LIN Error Handling

LIN error handling

You can monitor several errors that can occur in sending or receiving data. This is useful for error simulation on the LIN bus. The errors are administered by error counters. Several types of error counters are available:

- Error counter (rx and tx error counters) recommended by LIN specification 1.2. The counter is increased by 8 if an error occurs when sending/receiving a message and decreased by 1 if the message was sent/received without error. The counter can regenerate itself. Sporadically occurring errors do not trigger an event to the application.
- Error counter that is increased (never decreased) when an RX or TX error occurs. The ID indicates which node is defective.

Related topics

References

Data Structures: dslin_frame_status_t.....23

Error Detection of a Slave Node

Introduction

Different types of transmit and receive errors can be detected on a slave node.

Transmit errors

The slave node keeps track of any corrupted transmission by incrementing the slave transmit error counter. This counter is increased by 8 each time the transmitted synchronization or identifier field is locally corrupted. It is decreased by 1 (not below 0) each time both fields are read back properly. Use `dslin_node_tx_error_count_get` to read the total count of transmit errors.

Bit error A sending slave node detects a bit error in the data or checksum field while reading back its own transmission. You can read the bit error with the `dslin_node_info_bit_err_get` function.

Receive errors

A receiving slave node detects the following errors:

Checksum error A slave node detects a checksum error while reading a response from the bus. Use the `dslin_node_info_checksum_err_get` function to read the total count of checksum errors.

Slave not responding error The slave-not-responding error is detected while reading from the bus. The error occurs when a slave expects a message from another slave but no valid message appears on the bus within the given time. The time is specified in DSLIN_CHANNEL_BUS_IDLE_TIMEOUT_DEFAULT. Use the `dslin_node_info_snr_err_get` function to read the slave-not-responding error.

Identifier parity error The ID parity error is detected when a slave node receives a wrong parity in a header. Use the `dslin_node_info_idpar_err_get` function to read the ID parity error. This error does not increase the RX error counter.

Inconsistent-Synch-Byte error The error occurs when a slave task receives a synchronization byte in a header different from 0x55. Use the `dslin_node_info_synch_err_get` function to read the inconsistent-synch-byte error. This error does not increase the RX error counter.

Related topics

References

Error Detection of a Master Node

Introduction

Transmit and receive errors can be detected on a master node.

Transmit errors

The master node keeps track of any corrupted transmission by incrementing the master transmit error counter. This counter is increased by 8 each time the transmitted synchronization or identifier field is locally corrupted. It is decreased by 1 (not below 0) each time both fields are read back properly.

Bit error A sending master node detects a bit error in the data or checksum field while reading back its own transmission. You can read the bit error with the `dslin_node_info_bit_err_get` function. It allows you to read the total count of all bit errors for all transmitted LIN headers. This counter is increased by 1 each time a bit error occurs.

Receive errors

A receiving master node detects the following errors:

Checksum error A master node detects a checksum error while reading a response from the bus. Use the `dslin_node_info_checksum_err_get` function to read the total count of checksum errors.

Slave not responding error The slave-not-responding error is detected while reading from the bus. The error occurs when a master expects a message from another master (depending on the ID) but no valid message appears on the bus within the given time. The time is specified in DSLIN_CHANNEL_BUS_IDLE_TIMEOUT_DEFAULT. When a master does not expect a message (depending on the identifier) the error does not occur. Use the `dslin_node_info_snr_err_get` function to read the slave-not-responding error.

Identifier parity error The ID parity error is detected when a master node receives a wrong parity in a header. Use the `dslin_node_info_idpar_err_get` function to read the ID parity error. This error does not increase the RX error counter.

Inconsistent-Synch-Byte error The error occurs when a master task receives a synchronization byte in a header different from 0x55. Use the `dslin_node_info_synch_err_get` function to read the inconsistent-synch-byte error. This error does not increase the RX error counter.

Related topics

References

Overview of the Node Error Counters

Introduction

Several functions can be used to observe the various errors of a LIN node. Below is a short overview of their classifications, their effects on the common TX and RX counters and which functions must be used to read out their values.

Channel error counters

The following functions can be used to read out the different channel errors:

Function	Meaning
`dslin_node_info_idpar_err_get`	Reads the total count of ID parity errors (RX error).
`dslin_node_info_synch_err_get`	Reads the total count of synch field errors (RX error).
`dslin_node_info_no_bus_activity_err_get`	Reads the total count of no-bus-activity errors (RX error).
`dslin_node_info_extrabytes_err_get`	Reads the total count of extrabyte errors (RX error).
`dslin_node_info_header_bit_err_get`	Reads the total count of bit errors for all sent LIN headers (TX error).

TX frame error counter

The following function can be used to read out the TX frame error counter:

Function	Meaning
`dslin_node_info_bit_err_get`	Reads the total count of bit errors.

RX frame error counters

The following functions can be used to read out the different TX frame errors:

Function	Meaning
`dslin_node_info_checksum_err_get`	Reads the total count of checksum errors.
`dslin_node_info_snr_err_get`	Reads the total count of slave not responding errors.

LIN specification counters

LIN specification 1.2 recommends error counters that are increased by 8 if an error occurs and decreased by 1 if no error occurs:

Function	Meaning
`dslin_node_info_rx_err_get`	Reads the receive error counter for LIN responses.
`dslin_node_info_tx_err_get`	Reads the transmit error counter for LIN headers.

Note

The recommended (LIN specification 1.2) error counters do not distinguish between the different frame IDs. Use the following dSPACE error counters.

dSPACE error counter

The dSPACE error counter relates the errors to the IDs of the frames in which the errors occurred. It is increased by 1.

Function	Meaning
`dslin_node_rx_error_count_get`	Reads the total count of receive errors for a specific LIN response.
`dslin_node_tx_error_count_get`	Reads the total count of transmit errors for a specific LIN header.

Error Code

DSLIN_ERR

The error codes returned by the LIN access functions are listed in the DSLIN_ERR enumeration. The error messages are defined as follows:

Symbol	Meaning
`DSLIN_OK`	No error occurred, the command was accepted by the LIN board.
`DSLIN_OBJECT_REUSED`	An object with the same name as an existing LIN object was returned.
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. The error indicates that the LIN board is overloaded with command processing.
`DSLIN_NO_DATA_AVAILABLE`	The error code can be returned by the first read or get operation. It is a message and indicates no error.
`DSLIN_DATA_LOST`	Data was lost before it was read.
`DSLIN_ERR_MALLOC`	Memory allocation error on the master processor. There is not enough memory available on the master processor.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; the error can occur if a pointer is used which is not initialized or for which the initialization has failed.

Symbol	Meaning
DSLIN_ERR_NAME_TO_LONG	The name of the object is too long. Only 63 characters are allowed.
DSLIN_ERR_WRONG_TYPE	The input handle is the wrong type.
DSLIN_ERR_INTERNAL	An internal error occurred.
DSLIN_ERR_BOARD_NOT_INITIALIZED	The LIN board is not initialized. Use `dslin4330_board_init` to initialize the board.
DSLIN_ERR_BOARD_TIMEOUT	The board does not respond in time. Check the firmware and settings of the LIN board.
DSLIN_ERR_CHANNEL_NUMBER_ILLEGAL	The number of specified channels was not valid.
DSLIN_ERR_CHANNEL_COUNT	The number of LIN channels is limited to DSLIN_CHANNEL_COUNT_MAX per processor platform.
DSLIN_ERR_CHANNEL_NOT_FOUND	The LIN channel searched for was not found.
DSLIN_ERR_CHANNEL_BAUDRATE_ILLEGAL	An illegal baud rate is defined. Valid values are within the range 500 … 22000 bits/s.
DSLIN_ERR_CHANNEL_BREAKLENGTH_ILLEGAL	The break length must be at least 13 bit times long.
DSLIN_ERR_CHANNEL_BREAKDELIMITER_ILLEGAL	The break delimiter must be at least 10 bit times long.
DSLIN_ERR_CHANNEL_TRANSCEIVER_ILLEGAL	An illegal transceiver has been specified. Only ISO9141 transceivers or piggy-back are available.
DSLIN_ERR_CHANNEL_TERMINATION_ILLEGAL	The channels can be terminated with 1-kΩ pull-up resistors (master) or with 30-kΩ pull-up resistors (slave).
DSLIN_ERR_CHANNEL_BAUDRATE_DETECTION_ERROR	An error occurred during baud rate detection. The error occurs if the data contains no valid synchronization field.
DSLIN_ERR_CHANNEL_IS_IN_SLEEP_MODE	The channel is already in sleep mode.
DSLIN_ERR_CHANNEL_IS_DISABLED	The channel is disabled. Use `dslin_channel_enable` to enable the channel.
DSLIN_ERR_CHANNEL_RXMONITOR_NOT_INIT	The receive monitor is not initialized. Use `dslin_channel_rx_monitor_init` to initialize the monitor.
DSLIN_ERR_CHANNEL_RESOURCE_CONFLICT	One serial interface (UART) can be exclusively used for LIN or for DSSER. No mixed use is possible.
DSLIN_ERR_NODE_MASTER_ALREADY_PRESENT	There is already a master defined for the LIN bus. Only one master is allowed.
DSLIN_ERR_NODE_COUNT	There are too many nodes defined. Maximum 16 nodes are available.
DSLIN_ERR_NODE_NOT_FOUND	The LIN node searched for was not found.
DSLIN_ERR_NODE_TYPE_ILLEGAL	You tried to execute an action not allowed for this node. For example, you tried to execute a sleep command on a LIN slave that is only allowed for a LIN master.
DSLIN_ERR_NODE_INTERRUPT_ILLEGAL	The defined interrupt type is not valid for the node.
DSLIN_ERR_NODE_CONF_NOT_INIT	The node configuration is not completely initialized. The preconditions for a working node configuration are:

Symbol	Meaning
	▪ An initial node address is set using `dslin_node_initial_nad_set`. ▪ A current node address is set using `dslin_node_current_nad_set`. ▪ The node is initialized using `dslin_node_configuration_init`. If one of the preconditions is not fulfilled, the `dslin_node_configuration_service` function issues this error.
`DSLIN_ERR_NODE_CONF_SID_NOT_SUPPORTED`	The service identifier (SID) is not supported. The node configuration service detected an unsupported service identifier (SID).
`DSLIN_ERR_FRAME_COUNT`	There are too many frames defined. The valid range is 1 … 64.
`DSLIN_ERR_FRAME_NOT_FOUND`	The LIN frame searched for was not found.
`DSLIN_ERR_FRAME_NOT_INITIALIZED`	Before calling a frame, you have to initialize it with `dslin_frame_tx_init` or `dslin_frame_rx_init`.
`DSLIN_ERR_FRAME_INTERRUPT_ILLEGAL`	The defined interrupt is not valid.
`DSLIN_ERR_FRAME_MODE_ILLEGAL`	Wrong mode used in `dslin_frame_mode_set` on page 211
`DSLIN_ERR_RESPONSE_DELAY_ILLEGAL`	The specified response delay must be within the range 0 … 10 s.
`DSLIN_ERR_RESPONSE_LENGTH_ILLEGAL`	The length of the response is specified in the header's identifier field. Valid values are within the range 0 … 255.
`DSLIN_ERR_INTERRUPT_NOT_INITIALIZED`	The interrrupt is not initialized. Use `dslin_node_interrupt_init` or `dslin_node_frame_interrupt_init` to initialize an interrupt.
`DSLIN_ERR_INTERRUPT_COUNT`	There are too many interrupts defined. You can define up to 2048 interrupts.
`DSLIN_ERR_INTERRUPT_NOT_FOUND`	The LIN interrupt with the specified name was not found.
`DSLIN_ERR_SCHEDULE_INTERRUPT_ILLEGAL`	The defined interrupt type is not valid for the schedule.
`DSLIN_ERR_SCHEDULE_NOT_FOUND`	The LIN schedule searched for was not found.
`DSLIN_ERR_SCHEDULE_COUNT`	Too many LIN schedules specified for the LIN master node. The valid range is within 1 … 32.
`DSLIN_ERR_SCHEDULE_ENTRY_COUNT`	Too many entries specified for the LIN schedule (LIN master node is required). The valid range is 1 … 64.
`DSLIN_ERR_SCHEDULE_FRAME_TIME_ILLEGAL`	The frame time is illegal. The valid range is 0.0025 … 2.0 seconds.
`DSLIN_ERR_SCHEDULE_POSITION_ILLEGAL`	The schedule position is illegal. The valid range is 1 … 64.

DSLIN_ERROR_LEVEL

The different error levels are defined in the `DSLIN_ERROR_LEVEL` enumeration:

Error Level	Returned Values	Description
`DSLIN_NONE`	0	No error occurred.
`DSLIN_INFO`	1	An information message is returned.
`DSLIN_WARN`	2	A warning message is returned.
`DSLIN_FATAL`	3	A fatal error message is returned.

Related topics

References

dslin_error_print

Syntax

```
enum DSLIN_ERROR_LEVEL dslin_error_print (
   dslin_obj_p obj,
   enum DSLIN_ERROR error);
```

Purpose

To print error information about a DSLIN object, for example, for a frame.

Description

The macro prints error messages of the specified DSLIN object to the dSPACE log file. The dSPACE log file can be opened in the dSPACE experiment software.

Parameters

obj Specifies the DSLIN object, for example, a frame.

error Specifies the error that is printed.

Return value

The macro returns the error level specified by the `DSLIN_ERROR_LEVEL` enumeration.

Example

The following example shows how to use the `dslin_error_print` macro.

```
enum DSLIN_ERROR_LEVEL level = DSLIN_NONE;
  level = dslin_error_printf( frame, error );
  if( level )
  {
    // Exception handling
  }
```

DS4330 Access Functions

Introduction

The functions of this module are used to initialize the DS4330 LIN Interface Board.

Where to go from here

Information in this section

dslin4330_board_init

Syntax

```
enum DSLIN_ERROR dslin4330_board_init(
      phs_addr_t base,
      dslin_board_p* board);
```

Include file

`dslin.h`

Purpose

To initialize the LIN board to be used. The function creates a handle to the LIN board.

> **Note**
>
> Use the `dslin4330_board_init` function only during the initialization of the system.

Description

The `dslin4330_board_init` function creates a handle to the LIN board. The returned handle can be used to define channels and update the data processed by the LIN software on the processor board. Refer to `dslin_channel_create` on page 55 and `dslin_board_update` on page 42.

Parameters

base Address of the board in a PHS-bus-based system. Use the predefined macro DS4330_1_BASE, DS4330_2_BASE and so on.

board Returned pointer to the LIN board.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function was executed successfully.
`DSLIN_ERR_MALLOC`	A memory allocation error on the master processor occurred. There is not enough memory available on the master processor board.
`DSLIN_ERR_BOARD_TIMEOUT`	The board does not respond within one second. Check the firmware and settings of the LIN board.
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Example

The example shows how to initialize a LIN board.

```
// The init function outputs an error code
// which is stored in this variable.
enum DSLIN_ERROR error = DSLIN_OK;
// Pointer to the LIN board.
dslin_board_p lin_board = NULL;
// Initialize the LIN board.
// Any error is reported in error.
error = dslin4330_board_init( DS4330_1_BASE, &lin_board );
// Print the error information.
dslin_board_error_print( lin_board, error );
```

```
if( error != DSLIN_OK )
{
// Error handling.
}
```

Related topics

Basics

Local Interconnect Network (LIN) (DS4330 Features 🕮)

References

LIN Error Handling.....26
Standard Defines.....15

ds4330_reset_on_ioerr_enable

Syntax

```
void ds4330_reset_on_ioerr_enable(
   phs_addr_t base);
```

Include file

`dslin.h`

Purpose

To reset the board if the I/O error line signals an error.

Description

In addition to the standard control lines for reading and writing, the PHS bus provides an error line. After the `ds4330_reset_on_ioerr_enable` function is executed the DS4330 reacts to the line's state. If the line is active, the DS4330 is reset. Communication to the DS4330 is no longer possible until you initialize the board again.

Parameters

base PHS bus base address of the board

Return value

None

Related topics

References

DS4330 Access Functions.....35
ds4330_reset_on_ioerr_disable.....38
dslin4330_board_init.....35

ds4330_reset_on_ioerr_disable

Syntax

```
void ds4330_reset_on_ioerr_disable(
   phs_addr_t base);
```

Include file

`dslin.h`

Purpose

To disable resetting the board if the I/O error line signals an error.

Description

In addition to the standard control lines for reading and writing, the PHS bus provides an error line. After the `ds4330_reset_on_ioerr_disable` function is executed the DS4330 does not react to the line's state. If the line is active, DS4330 is not reset and communication to the DS4330 is still possible.

Parameters

base PHS bus base address of the board

Return value

None

Related topics

References

ds4330_dpmem_interrupt_clear

Syntax

```
UInt32 ds4330_dpmem_interrupt_clear(
   phs_addr_t base);
```

Include file	`dslin.h`
Purpose	To clear the DPMEM interrupt.
Description	The `ds4330_dpmem_interrupt_clear` function clears the content of the interrupt memory location on the DPMEM.
Parameters	**base** PHS bus base address of the board
Return value	The return value is not specified and therefore not used.
Example	For examples on how to use the function, see Example of Using LIN Frame Interrupts on page 227 and Example of Requesting LIN Interrupts on page 229.
Related topics	References DS4330 Access Functions....35 LIN Error Handling....26 Standard Defines....15

LIN Access Functions

Introduction

The LIN access functions are arranged in several modules which allow handling of the LIN board, the channels, nodes and frames of a LIN bus. The following modules are provided.

Where to go from here

Information in this section

Information in other sections

LIN Board Handling

Introduction

The functions and definitions of this module can be used for error documentation and LIN board data handling.

Note

You can use the functions of this module only when the board is initialized.

Where to go from here

Information in this section

Information in other sections

dslin_board_update

Syntax

```
enum DSLIN_ERROR dslin_board_update(
dslin_board_p board);
```

Include file

`dslin.h`

Purpose

To update the LIN data on the main processor.

Description

The `dslin_board_update` function copies the data from the LIN board to the internal structures on the processor board, for example, to the DS1007. The function makes the data from the LIN board available to the main processor board. Therefore, it is necessary to call the update function to read the current LIN data.

Parameters

board Pointer to the LIN board. The board must be already initialized by the corresponding initialization function, for example, `dslin4330_board_init`.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	Data was updated successfully.
`DSLIN_NO_DATA_AVAILABLE`	The LIN board has not sent any data to the processor board since the last update.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_board_error_print 43
dslin4330_board_init 35
LIN Error Handling 26
Standard Defines 15

dslin_board_error_print

Syntax

```
define dslin_board_error_print(
      dslinboard_p board,
      enum error);
```

Include file

`dslin.h`

Purpose

To report LIN board errors to the dSPACE log file. If `error == DSLIN_OK`, nothing is written to the output.

Note

- Reporting the error information to the log file is a time-consuming process. Consider this when using the function within your task.
- The dSPACE log file can be opened in the dSPACE experiment software.

Parameters

board Pointer to the LIN board, returned by the board-specific `dslin4330_board_init` function.

error Error code to be written to the log file as plain text.

Return value

None

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)

References

dslin_board_update 42
LIN Error Handling 26
Standard Defines 15

dslin_board_send_wait_mode_enable

Syntax

```
enum DSLIN_ERROR dslin_board_send_wait_mode_enable(
   dslin_board_p board,
   dsfloat timeout);
```

Include file

`dslin.h`

Purpose

To enable the timeout for a command that is requested by a LIN function.

> **Note**
>
> All commands supporting the error code `DSLIN_COMMUNICATION_OVERLOAD` are affected by the `dslin_board_send_wait_mode_enable` function.

Parameters

board Pointer to the LIN board, returned by the board-specific `dslin4330_board_init` function.

timeout Lets you enter the timeout value. The currently executed function waits the specified timeout value. If the function is not capable to send the data or command within the timeout to the LIN board, it terminates itself with the error code `DSLIN_COMMUNICATION_OVERLOAD`.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	Data was updated successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)

References

dslin_board_send_wait_mode_disable 45
dslin4330_board_init 35
LIN Error Handling 26
Standard Defines 15

dslin_board_send_wait_mode_disable

Syntax

```
enum DSLIN_ERROR dslin_board_send_wait_mode_disable(
   dslin_board_p board);
```

Include file

`dslin.h`

Purpose

To disable the timeout for a command that is requested by a LIN function.

> **Note**
>
> All commands supporting the error code `DSLIN_COMMUNICATION_OVERLOAD` are affected by the `dslin_board_send_wait_mode_disable` function.

Parameters

board Pointer to the LIN board, returned by the board-specific `dslin4330_board_init` function.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	Data was updated successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)

References

dslin_board_send_wait_mode_enable..... 44
dslin4330_board_init..... 35
LIN Error Handling..... 26
Standard Defines..... 15

LIN Channel Handling

Introduction

The following functions are used to initialize and handle LIN channels.

Note

Each channel of the LIN board corresponds to a serial channel. You can combine various channels to specify a LIN bus that contains nodes with different baud rates, for example.

Where to go from here

Information in this section

Information in other sections

Example of Initializing a LIN Channel

Preconditions

You need the following information to initialize a LIN channel:

- The name of the LIN channel (physical interface)
- Baud rate of the bus
- Values of the LIN bus timing parameters. If you are unsure use the default values.
- The transceiver (ISO9141)
- The termination

Example

The following example shows how to initialize a channel:

```
#include <dslin.h>
void lin_channel_init( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  dslin_channel_p lin_channel = NULL;
  // Create the handle for the LIN interface.
  error = dslin_channel_create( lin_board, "LIN-Interface0", 1, &lin_channel );
  dslin_channel_error_print( lin_channel, error );
```

```
  // Initialize the LIN interface.
  error = dslin_channel_init( lin_channel, //Referenced LIN interface
                              9600,        //baudrate
                              13,          //break lenght
                              5,           //break delimiter
                              DSLIN_TRANSCEIVER_ISO9141,
                              DSLIN_TERMINATION_SLAVE_30K );
  dslin_channel_error_print( lin_channel, error );
  // Start the LIN interface.
  error = dslin_channel_enable( lin_channel );
  dslin_channel_error_print( lin_channel, error );
}
```

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

References

Example of Setting up a Response Frame Directly on a LIN Channel

Example

The following example shows how to setup two frame responses waiting for an incoming corresponding LIN header. If the received LIN header matches, the LIN response is sent.

```
#include <brtenv.h>
#include <dslin.h>
/* Data structures used for the example. */
dslin_board_p           LinBoard   = NULL;
dslin_channel_p         LinChannel = NULL;
dslin_channel_tx_data_t LinTxFrame1 = {0};
dslin_channel_tx_data_t LinTxFrame2 = {0};
void main(void)
{
  init();
  dslin4330_board_init(DS4330_1_BASE, &LinBoard);
  /* Setup the LIN channel */
  dslin_channel_create(LinBoard, "CH0", 1, &LinChannel);
  dslin_channel_init(LinChannel, 9600, 20, 1,
                     DSLIN_TRANSCEIVER_ISO9141,
                     DSLIN_TERMINATION_SLAVE_30K );
```

```
  /* Set all data members to zero. */
  memset(&LinTxFrame1, 0, sizeof(dslin_channel_tx_data_t));
  /* Setup one response with enhanced checksum, dlc=2 and identifier=0x1. */
  LinTxFrame1.identifier = 0x1;
  LinTxFrame1.dlc        = 2;
  LinTxFrame1.tx_mode    = DSLIN_CHANNEL_TX_ENHANCED_CHECKSUM;
  LinTxFrame1.data[0]    = 0xAB;
  LinTxFrame1.data[1]    = 0xCD;
  /* Transfer the settings to the hardware. */
  dslin_channel_tx_response_write(LinChannel, &LinTxFrame1);
  /* Setup one response with classic checksum, dlc=4 and identifier=0x2. */
  memset(&LinTxFrame2, 0, sizeof(dslin_channel_tx_data_t));
  LinTxFrame2.identifier = 0x2;
  LinTxFrame2.dlc        = 4;
  LinTxFrame2.data[0]    = 0x12;
  LinTxFrame2.data[1]    = 0x34;
  LinTxFrame2.data[2]    = 0x56;
  LinTxFrame2.data[3]    = 0x78;
  /* Transfer the settings to the hardware. */
  dslin_channel_tx_response_write(LinChannel, &LinTxFrame2);
  /* Activate the LIN channel. */
  dsline_channel_enable(LinChannel);
  /*
   * Now the hardware waits for a matching LIN header.
   * If a header a matching LIN header was received the
   * LIN response is automatically sent by the hardware.
   */
  for(;;)
  {
      RTLIB_BACKGROUND_SERVICE();
      dslin_board_update(LinBoard);
  }
}
```

Related topics

References

Example of Monitoring Data of a LIN Bus

Preconditions

You have to provide a working external LIN communication.

Example

The following example shows how to observe data of a LIN bus:

```
#include <brtenv.h>
#include <dslin.h>
/* Data structures used for the example. */
dslin_board_p           LinBoard   = NULL;
dslin_channel_p         LinChannel = NULL;
dslin_channel_rx_data_t LinMonitorData;
UInt32 LinMonitorClient = 0;
void main(void)
{
  init();
  dslin4330_board_init(DS4330_1_BASE, &LinBoard);
  /* Setup the LIN channel */
  dslin_channel_create(LinBoard,
                       "LinChannel", 1,
                       &LinChannel);
  dslin_channel_init(LinChannel,
                     9600, 20, 1, /* baudrate = 9600baud */
                     DSLIN_TRANSCEIVER_ISO9141,
                     DSLIN_TERMINATION_SLAVE_30K );
  /*
   * Setup the LIN monitor with a max DLC of 8
   * and a FIFO to store 100 LIN response frames.
   */
  dslin_channel_rx_monitor_init(LinChannel, 8, 100);
  /* Connect one client to read from the monitor FIFO. */
  dslin_channel_rx_monitor_client_init(LinChannel, &LinMonitorClient );
  /* Activate the LIN channel. */
  dslin_channel_enable(LinChannel);
  for(;;)
  {
      RTLIB_BACKGROUND_SERVICE();
      dslin_board_update(LinBoard);
      /* Read from the monitor FIFO. */
      if(DSLIN_OK == dslin_channel_rx_monitor_client_read(LinChannel,
                                                          LinMonitorClient,
                                                          &LinMonitorData))
      {
          /* At least one frame response or one error was received. */
          /* Test if this was an error. */
          if (LinMonitorData.rx_status)
          {
              if (LinMonitorData.rx_status == DSLIN_CHANNEL_RX_CHECKSUM_ERR)
              {
                  msg_info_printf(0,0,"LIN checksum error detected!" );
              }
              if (LinMonitorData.rx_status == DSLIN_CHANNEL_RX_SNR_ERR)
              {
                  msg_info_printf(0,0,"LIN slave not responding error!" );
              }
```

```
            if (LinMonitorData.rx_status == DSLIN_CHANNEL_RX_DATA_LOST_ERR)
            {
                msg_info_printf(0,0,"Data lost error!" );
            }
        }
        else
        {
            /* No error detected, output the frame information! */
            msg_info_printf(0,0,
            "RX id:0x%x,dlc:%d,data(hex):%x,%x,%x,%x,%x,%x,%x,%x",
            LinMonitorData.identifier, LinMonitorData.dlc,
            LinMonitorData.data[0], LinMonitorData.data[1],
            LinMonitorData.data[2], LinMonitorData.data[3],
            LinMonitorData.data[4], LinMonitorData.data[5],
            LinMonitorData.data[6], LinMonitorData.data[7]);
        }
    }
  }
}
```

Related topics

References

dslin_channel_lookup

Syntax

```
enum DSLIN_ERROR dslin_channel_lookup (
   const char * channel_name
   dslin_channel_p * channel);
```

Include file

`dslin.h`

Purpose

To look for an existing LIN channel.

Note

Use the `dslin_channel_lookup` function only during the initialization phase of the system.

Parameters

channel_name Name of the LIN channel to search for. The length of the name is limited to 63 characters.

channel Returns a pointer to a LIN channel. The function returns NULL if no channel was found.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The channel searched for was found.
DSLIN_ERR_CHANNEL_NOT_FOUND	No channel with the specified name found on the processor board.

Example

The example shows how to implement code to search for a specified LIN channel.

```
enum DSLIN_ERROR error = DSLIN_OK;
dslin_channel_p lin_channel = NULL;
// Search the handle to the LIN channel with the name "LIN Interface0".
error = dslin_channel_lookup( "LIN Interface0", &lin_channel );
dslin_channel_error_print( lin_channel, error );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Channel.....49

References

dslin_channel_create.....55
dslin_channel_error_print.....59
dslin_channel_init.....57
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_create

Syntax

```
enum DSLIN_ERROR dslin_channel_create(
   dslin_board_p board,
   const char* name,
   UInt8 channel_no,
   dslin_channel_p* channel);
```

Include file

`dslin.h`

Purpose

To create a LIN channel (LIN interface). Each LIN channel represents a serial interface.

Description

The `dslin_channel_create` function allocates memory for a new LIN channel or returns the pointer to an existing LIN channel with the specified name on the processor board. A newly created channel is disabled by default. Refer to `dslin_channel_enable` on page 60 for information on how to enable the created channel. If an existing channel is returned, the enabling state depends on the state of that channel.

Note

Use the `dslin_channel_create` function only during the initialization phase of the system.

Parameters

board Pointer to the LIN board the channel is connected to.

name Name of the LIN channel. The name is limited to 63 characters. If the string is NULL or empty (“ “) the LIN channel cannot be found by the `dslin_channel_lookup` function (see `dslin_channel_lookup` on page 53).

channel_no Physical channel number used on the board. The range depends on the number of LIN channels supported by the I/O board used, for example, DS4330 supports 1 … 16 channels.

channel Returned pointer to the created LIN channel. Returns NULL if the function fails.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	A new channel has been successfully created.
DSLIN_OBJECT_REUSED	A pointer to an existing channel with the same name on the node was returned.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_ERR_MALLOC	Memory allocation error. There is not enough memory available on the master processor board.
DSLIN_ERR_CHANNEL_NUMBER_ILLEGAL	Illegal channel number; valid values are within the range 1 … 16.
DSLIN_ERR_CHANNEL_COUNT	Too many channels specified for the complete processor board. The number of channels available is DSLIN_CHANNEL_COUNT_MAX.

Example

The example shows how to create a new LIN channel. For a detailed example of LIN channel handling, refer to Example of Initializing a LIN Channel on page 49.

```
enum DSLIN_ERROR error = DSLIN_OK;
dslin_channel_p lin_channel = NULL;
// Create the handle for the LIN channel.
error = dslin_channel_create( lin_board, "LIN-Bus0", 1, &lin_channel );
dslin_channel_error_print( lin_channel, error );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Channel 49

References

dslin_channel_error_print 59
dslin_channel_init 57
dslin_channel_lookup 53
LIN Channel Handling 47
LIN Error Handling 26
Standard Defines 15

dslin_channel_init

Syntax

```
enum DSLIN_ERROR dslin_channel_init (
   dslin_channel_p channel,
   UINt16 baudrate,
   UInt8 breaklength,
   UInt8 breakdelimiter,
   enum DSLIN_TRANSCEIVER_TYPE transceiver,
   enum DSLIN_TERMINATION_TYPE termination);
```

Include file

`dslin.h`

Purpose

To initialize a LIN channel (interface). Each LIN channel represents a serial interface.

Note

- Start the LIN channel with the `dslin_channel_enable` function.
- Use the `dslin_channel_init` function only during the initialization of the system.

Description

The `dslin_channel_init` function performs a setup for one physical interface (channel) to the LIN bus. The LIN bus can be operated in different configurations.

If the master node is an external, non-simulated node, no termination must be set for it.

The simulated LIN slave nodes are terminated with 30 kΩ (DSLIN_TERMINATION_SLAVE_30K). If the master is also simulated by the application the termination has to be set to DSLIN_TERMINATION_MASTER_1K.

Parameters

channel Pointer to a LIN channel.

baudrate Specifies the baud rate according to LIN specification 1.2. The following values are available:

- 2400 bit/s
- 9600 bit/s
- 19200 bit/s

Note

- You can also use baud rates within the extended range 500 … 22000 Bit/s. The increment is to be defined with `DSLIN_CHANNEL_BAUDRATE_STEPSIZE`. See Standard Defines on page 15.
- The `breaklength` and `breakdelimiter` parameters are related to LIN master channel. Only a master can send a frame header.

breaklength Specifies the synchronization break length. Valid values are within the range 1 … 128 bit times. The minimum bit time is 13 according to LIN specification 1.2.

breakdelimiter Specifies the break delimiter. Valid values are within the range 1 … 128 bit times.

transceiver Specifies the transceiver used. The valid symbols are:

Symbol	Meaning
`DSLIN_TRANSCEIVER_ISO9141`	Standard LIN transceiver

termination Specifies the termination type of the LIN channel. See Data Types and Enumerations on page 17 for detailed information on termination. Valid values are:

Symbol	Meaning
`DSLIN_TERMINATION_MASTER_1K`	Termination for a master node.
`DSLIN_TERMINATION_SLAVE_30K`	Termination for a slave node

Return value The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new channel has been created successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_CHANNEL_BAUDRATE_ILLEGAL`	An illegal baud rate is used. Valid values are within 500 … 22000 kBaud.
`DSLIN_ERR_CHANNEL_BREAKLENGTH_ILLEGAL`	Illegal break length is used.
`DSLIN_ERR_CHANNEL_BREAKDELIMITER_ILLEGAL`	Illegal break delimiter is used.
`DSLIN_ERR_CHANNEL_TRANSCEIVER_ILLEGAL`	The selected transceiver is not supported.
`DSLIN_ERR_CHANNEL_TERMINATION_ILLEGAL`	The selected termination is not supported.

Example

The example shows how to initialize a LIN channel.

For a detailed example of LIN channel handling, refer to dslin_channel_lookup on page 53.

```
enum DSLIN_ERROR error = DSLIN_OK;
// Initialize the LIN interface.
error = dslin_channel_init( lin_channel, //Referenced LIN interface
                            9600,        //baud rate
                            13,          //break lenght
                            5,           //break delimiter
                            DSLIN_TRANSCEIVER_ISO9141,
                            DSLIN_TERMINATION_SLAVE_30K );
dslin_channel_error_print( lin_channel, error );
if( error != DSLIN_OK )
{
// Error handling.
}
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_error_print

Syntax

```
define dslin_channel_error_print(
   dslin_channel_p channel,
   int error);
```

Include file

`dslin.h`

Purpose

To report errors to the dSPACE log file.

If `error==DSLIN_OK`, nothing is written to the log file.

Note

- Reporting the error information to the log file is a time-consuming process. Consider this when using the function within your task.
- The dSPACE log file can be opened in the dSPACE experiment software.

Parameters

channel Pointer to a LIN channel.

error Error code to be written to the log file as plain text.

Return value

None

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Channel 49

References

dslin_channel_create 55
dslin_channel_init 57
dslin_channel_lookup 53
LIN Channel Handling 47
LIN Error Handling 26
Standard Defines 15

dslin_channel_enable

Syntax

```
enum DSLIN_ERROR dslin_channel_enable(
   dslin_channel_p channel);
```

Include file

`dslin.h`

Purpose

To enable the LIN channel.

Description

The `dslin_channel_enable` function activates the physical interface (channel) to the LIN bus. The termination and the transceiver are not affected by the function.

Parameters

channel Pointer to a LIN channel.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The channel is enabled.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Example

The following example shows how to enable a LIN channel. For a detailed example of LIN channel handling, refer to Example of Initializing a LIN Channel on page 49.

```
error = dslin_channel_enable(lin_channel);

if(DSLIN_OK != error)
{
// error handling
}
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_disable

Syntax

```
enum DSLIN_ERROR dslin_channel_disable(
   dslin_channel_p channel);
```

Include file

`dslin.h`

Purpose

To disable a LIN channel.

Description

The physical interface to the LIN bus is deactivated. The termination and the transceiver are not affected by the function.

Parameters

channel Pointer to a LIN channel.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The channel is disabled.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_init.....57
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_transceiver_set

Syntax

```
enum DSLIN_ERROR dslin_channel_transceiver_set(
   dslin_channel_p channel,
   enum DSLIN_TRANSCEIVER_TYPE transceiver);
```

Include file

`dslin.h`

Purpose

To select the LIN transceiver.

Note

The settings specified by the `dslin_channel_transceiver_set` function have to be applied with the `dslin_frame_apply_settings` function.

Parameters

channel Pointer to a LIN channel.

transceiver Specifies the transceiver used. The valid symbols are:

Symbol	Meaning
`DSLIN_TRANSCEIVER_ISO9141`	Standard LIN transceiver

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The transceiver for the channel has been successfully specified.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
DSLIN_ERR_CHANNEL_TRANSCEIVER_ILLEGAL	The transceiver used is not supported.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_apply_settings 76
dslin_channel_transceiver_get 64
dslin_channel_transceiver_sleep 65
LIN Channel Handling 47
LIN Error Handling 26
Standard Defines 15

dslin_channel_transceiver_get

Syntax

```
enum DSLIN_ERROR dslin_channel_transceiver_get(
   dslin_channel_p  channel,
   DSLIN_TRANSCEIVER_ENUM*  pTransceiver);
```

Include file

`dslin.h`

Purpose

To get the LIN transceiver type of a LIN channel.

Description

The function returns the transceiver type used by the selected LIN channel.

Parameters

channel Pointer to a LIN channel.

pTransceiver Pointer to the returned transceiver type.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The function has successfully returned the transceiver type.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if channel or pTransceiver == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_transceiver_set 63
LIN Channel Handling 47
LIN Error Handling 26
Standard Defines 15

dslin_channel_transceiver_sleep

Syntax

```
enum DSLIN_ERROR dslin_channel_transceiver_sleep(
   dslin_channel_p channel);
```

Include file

dslin.h

Purpose

To set the LIN transceiver to sleep mode.

Note

Only the transceiver, not the whole LIN bus, is set to sleep mode.

Parameters

channel Pointer to a LIN channel.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The transceiver has been set to sleep mode.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_apply_settings.....76
dslin_channel_transceiver_set.....63
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_termination_set

Syntax

```
enum DSLIN_ERROR dslin_channel_termination_set(
   dslin_channel_p channel,
   enum DSLIN_TERMINATION_TYPE termination);
```

Include file

dslin.h

Purpose

To specify the termination type of the LIN channel.

> **Note**
>
> The settings specified by the `dslin_channel_termination_set` function have to be applied with the `dslin_frame_apply_settings` function.

Description

Each LIN channel can be configured as the master or as a slave. The difference between master and slave is that in master configuration the LIN bus has an external 1-kΩ pull-up resistor to the battery voltage and in slave configuration it does not. The pull-up resistor can be enabled/disabled for each channel.

> **Note**
>
> After power-up, all 16 LIN transceivers are configured as slaves.

Parameters

channel Pointer to a LIN channel.

termination Select one of the predefined symbols to specify the termination. Valid values are:

Symbol	Meaning
`DSLIN_TERMINATION_MASTER_1K`	Termination for a master node
`DSLIN_TERMINATION_SLAVE_30K`	Termination for a slave node

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The termination for the selected channel has been set successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_CHANNEL_TERMINATION_ILLEGAL`	The termination is not supported.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_apply_settings.....76
dslin_channel_termination_get.....68
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_termination_get

Syntax

```
enum DSLIN_ERROR dslin_channel_termination_get(
   dslin_channel_p  channel,
   DSLIN_TERMINATION_ENUM*  pTermination);
```

Include file

`dslin.h`

Purpose

To get the LIN termination type of a LIN channel.

Description

The function returns the termination type specified for the selected LIN channel.

Parameters

channel Pointer to a LIN channel.

pTermination Pointer to the returned termination type.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the termination type.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel or pTermination == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_termination_set.....66
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_baudrate_set

Syntax

```
enum DSLIN_ERROR dslin_channel_baudrate_set(
   dslin_channel_p channel,
   UInt16 baudrate);
```

Include file

`dslin.h`

Purpose

To set the baud rate of the LIN channel.

Note

The settings specified by the `dslin_channel_baudrate_set` function have to be applied with the `dslin_channel_apply_settings` function (see `dslin_channel_apply_settings` on page 76).

Description

It is recommended to use the baud rates according to LIN specification 1.2. The following values are valid:

- 2400 bit/s
- 9600 bit/s
- 19200 bit/s

Nevertheless, you can use the extended range within 500 … 22000 bit/s. The increment to increase the baud rate is specified in `DSLIN_CHANNEL_BAUDRATE_STEPSIZE`. Refer to Data Types and Enumerations on page 17.

Parameters

channel Pointer to a LIN channel.

baudrate Selects the baud rate for the LIN channel. For the valid values, see the list above.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The baud rate for the selected channel has been set successfully.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
DSLIN_ERR_CHANNEL_BAUDRATE_ILLEGAL	An illegal baud rate is used. Valid values are within 500 … 22000 Bit/s.

Example

The following example shows you how to set the baud rate.

```
error = dslin_channel_baudrate_set(lin_channel, 9600);
if( DSLIN_OK != error )
{
// error handling
}
// Do not forget to apply the settings!
error = dslin_channel_apply_settings( lin_channel );
if( DSLIN_OK != error )
{
// error handling
}
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_baudrate_detection_disable.....80
dslin_channel_baudrate_detection_enable.....79
dslin_channel_baudrate_detection_get.....81
dslin_channel_baudrate_get.....71
LIN Channel Handling.....47

dslin_channel_baudrate_get

Syntax

```
enum DSLIN_ERROR dslin_channel_baudrate_get(
   dslin_channel_p  channel,
   UInt32*  pBaudrate);
```

Include file

`dslin.h`

Purpose

To get the baud rate of a LIN channel.

Description

The function returns the baud rate specified for the selected LIN channel.

Parameters

channel Pointer to a LIN channel.

pBaudrate Pointer to the returned baud rate.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the baud rate.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel or pBaudrate == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_baudrate_set.....69
LIN Channel Handling.....47

dslin_channel_breaklength_set

Syntax

```
enum DSLIN_ERROR dslin_channel_breaklength_set(
   dslin_channel_p channel,
   UInt8 breaklength);
```

Include file

`dslin.h`

Purpose

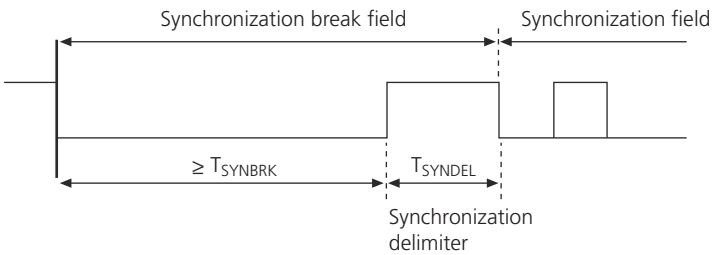
To set the synchronization break length for a LIN master channel.

Note

- The settings specified by the `dslin_channel_breaklength_set` function have to be applied with the `dslin_channel_apply_settings` function.
- The `dslin_channel_breaklength_set` function is related to a LIN master channel. Only a master can send frame headers.

Description

The beginning of a message frame can be identified by the first field of the frame, which is the synchronization break field. It is part of the header that is always sent by a master node. The synchronization break enables the slave tasks to synchronize on the bus clock. According to LIN specification 1.2, the minimum break length (T_{SYNBRK}) is 13 bit times. See the following illustration, which shows the dominant signal during the synchronization break and the following synchronization delimiter field (recessive):



For detailed information, refer to LIN specification 1.2. You can also specify the break delimiter value, see `dslin_channel_breakdelimiter_set` on page 73.

Parameters

channel Pointer to a LIN channel.

breaklength Specifies the synchronization break length within the range 1 … 128 bit times. The minimum value is 13 bit times.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The baud rate for the selected channel has been set successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_CHANNEL_BREAKLENGTH_ILLEGAL`	An illegal break length was used. The minimum break length is 13 bit times.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_apply_settings.....76
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_breakdelimiter_set

Syntax

```
enum DSLIN_ERROR dslin_channel_breakdelimiter_set(
   dslin_channel_p channel,
   UInt8 breakdelimiter);
```

Include file

`dslin.h`

Purpose

To set the synchronization break delimiter for a LIN master channel.

Note

- The settings specified by the `dslin_channel_breakdelimiter_set` function have to be applied with the `dslin_channel_apply_settings` function.
- The `dslin_channel_breakdelimiter_set` function is related to a LIN master channel. Only a master can send a frame header.

Description

The synchronization break delimiter is part of the synchronization break and enables the slave to detect the start bit of the following synchronization field. According to LIN specification 1.2 the minimum value (T_{SYNDEL}) is 1 bit time. For more information, see dslin_channel_breaklength_set on page 72.

Parameters

channel Pointer to a LIN channel.

breakdelimiter Specifies the break delimiter value. It has to be within the range 1 … 128 bit times. The minimum value is 1 bit time.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The baud rate for the selected channel has been set successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_CHANNEL_BREAKDELIMITER_ILLEGAL`	An illegal breaklength is used. The minimum break delimiter length is 1 bit time.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_apply_settings.....76
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_synchfield_set

Syntax

```
enum DSLIN_ERROR dslin_channel_breakdelimiter_set(
   dslin_channel_p channel,
   UInt8 synchfield);
```

Include file

`dslin.h`

Purpose

To specify the synchronization field for a LIN master channel.

Note

- The settings specified by the `dslin_channel_synchfield_set` function have to be applied with the `dslin_frame_apply_settings` function.
- The `dslin_channel_synchfield_set` function is related to a LIN master channel. Only a master can send a frame header.

Description

The synchronization field is part of the frame header, follows the synchronization break field and contains the information on the bus clock. According to LIN specification 1.2 the synchronization field consists of the pattern ‘0x55’ (hex). During synchronization the time between the falling and rising edges of the bit pattern is measured.

The pattern always consists of 8 bits. You only can change the bit pattern by entering another value. See the following table for examples of bit patterns:

Decimal	Binary	Hexadecimal
0	0000 0000	0x00
85	1010 1010	0x55
255	1111 1111	0xFF

For detailed information on the synchronization procedure, refer to *LIN specification 1.2*.

Parameter

channel Pointer to a LIN channel.

synchfield Lets you specify the bit times for the synchronization field within the range 1 … 255 (dec.). LIN specification 1.2 proposes 0x55 (85 dec.).

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The baud rate for the selected channel has been set successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)

References

dslin_channel_apply_settings.....76
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_apply_settings

Syntax

```
enum DSLIN_ERROR dslin_channel_apply_settings(
   dslin_channel_p channel);
```

Include file `dslin.h`

Purpose

To apply the settings done with the channel set functions. The function transfers the data to the slave processor of the LIN board.

> **Note**
>
> After you have specified the settings with the corresponding `dslin_xxxx_set` functions, you can call the `dslin_channel_apply_settings` function once to transfer all settings to the slave processor of the LIN board.

Parameters

channel Pointer to a LIN channel.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The settings have been transferred successfully to the slave processor.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_restore_settings.....78
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_restore_settings

Syntax

```
enum DSLIN_ERROR dslin_channel_restore_settings(
   dslin_channel_p channel);
```

Include

`dslin.h`

Purpose

To restore the initialization values used during the last execution of `dslin_channel_init`.

Description

You can restore the following values used during the initialization of the channel. The parameter values currently used are overwritten:

- Baud rate
- Break length
- Break delimiter

Parameters

channel Pointer to a LIN channel.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The channel settings have been restored successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_baudrate_detection_enable

Syntax

```
enum DSLIN_ERROR dslin_channel_baudrate_detection_enable(
   dslin_channel_p channel);
```

Include

`dslin.h`

Purpose

To enable baud rate detection.

Note

The LIN channel must be enabled with the `dslin_channel_enable` function to receive a LIN header.

Description

The baud rate of the LIN bus can be detected by evaluating the synchronization pattern of a LIN header. You can read the detected baud rate with the `dslin_channel_baudrate_detection_get` function.

For more information on baud rate detection, refer to Testing Against Specification Limits (DS4330 Features 🕮).

Parameter

channel Pointer to a LIN channel.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The settings have been transferred successfully to the slave processor.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Example

The following example shows how to enable the baud rate detection.

```
dslin_channel_p channel = NULL;
dslin_channel_create( lin_board, "LIN1", 1, &channel );
dslin_channel_enable( channel );
dslin_channel_baudrate_detection_enable( channel );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_baudrate_detection_disable.....80
dslin_channel_baudrate_detection_get.....81
dslin_channel_enable.....60
LIN Channel Handling.....47

dslin_channel_baudrate_detection_disable

Syntax

```
enum DSLIN_ERROR dslin_channel_baudrate_detection_disable(
   dslin_channel_p channel);
```

Include

dslin.h

Purpose

To disable baud rate detection.

Parameter

channel Pointer to a LIN channel.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The settings have been transferred successfully to the slave processor.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_baudrate_detection_get 81
LIN Channel Handling 47
Standard Defines 15

dslin_channel_baudrate_detection_get

Syntax

```
enum DSLIN_ERROR dslin_channel_baudrate_detection_get(
   dslin_channel_p channel,
   dsfloat* baudrate,
   dsfloat* timestamp);
```

Include

`dslin.h`

Purpose

To read the detected baud rate.

Description

If no baud rate is detected, the `DSLIN_NO_DATA_AVAILABLE` symbol and the last detected baud rate and time stamp are returned. If no baud rate was detected before, the baud rate specified by the `dslin_channel_init` or `dslin_channel_baudrate_set` function is returned. If no baud rate was specified and no baud rate detection was executed before, the default baud rate is returned (DSLIN_CHANNEL_BAUDRATE_DEFAULT).

Errors can occur if the data measured contains no valid synchronization field. Refer to `dslin_channel_synchfield_set` on page 75.

For more information on baud rate detection, refer to Testing Against Specification Limits (DS4330 Features 🕮).

Parameters

channel Pointer to a LIN channel.

baudrate Address where the detected baud rate is stored.

timestamp Address where the time stamp is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	Data was updated successfully.
`DSLIN_NO_DATA_AVAILABLE`	The LIN board has not sent any data to the processor board since the last update.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_CHANNEL_BAUDRATE_ILLEGAL`	The detected baud rate is illegal. Valid values are within the range 500 … 20000 bits/s.
`DSLIN_CHANNEL_BAUDRATE_DETECTION_ERROR`	The baud rate detection has failed.

Example

The following example shows how to get the detected baud rate.

```
dsfloat baudrate = 0.0;
dsfloat timestamp= 0.0;
if( DSLIN_OK
== dslin_channel_baudrate_detection_get( channel, &baudrate, &timesta
mp ) )
{
    // Do something useful.
}
else
{
    // Handle the error.
}
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_baudrate_detection_disable.....80
dslin_channel_baudrate_detection_enable.....79
dslin_channel_baudrate_set.....69
dslin_channel_init.....57
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_is_wake

Syntax

```
enum DSLIN_ERROR dslin_channel_is_wake(
   dslin_channel_p channel,
   enum DSLIN_BOOL* is_wake);
```

Include file

`dslin.h`

Purpose

To check whether the channel is in wake-up or sleep mode.

Note

The channel is in wake-up mode if the `dslin_channel_enable` function was executed successfully. Before calling the `dslin_channel_is_wake` function, you have to update the state information by calling the `dslin_board_update` function.

Description

The function enables you to check whether or not this channel is in sleep mode.

Parameters

channel Pointer to a LIN channel.

is_wake Address where the state of the channel is stored. Two values are available:

Symbol	Meaning
DSLIN_TRUE	Indicates that the channel is in wake-up mode.
DSLIN_FALSE	Indicates that the channel is in sleep mode after having received a sleep command from the master.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	Check on whether channel is in sleep or wake-up mode was successful.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_ERR_NO_DATA_AVAILABLE	There is no data available with this channel.

Example

The example shows how to use the function.

```
enum DSLIN_ERROR error;
enum DSLIN_BOOL is_wake = DSLIN_FALSE;
error = dslin_channel_is_wake( channel, &is_wake );
if( DSLIN_TRUE == is_wake && DSLIN_ERROR == DSLIN_OK )
{
// The LIN channel is waked.
}
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Waking Up a LIN Bus (DS4330 Features 🕮)

References

dslin_board_update....42
dslin_channel_apply_settings....76
dslin_channel_enable....60
dslin_channel_transceiver_sleep....65
LIN Channel Handling....47
LIN Error Handling....26
Standard Defines....15

dslin_channel_rx_monitor_init

Syntax

```
enum DSLIN_ERROR dslin_channel_rx_monitor_init(
   dslin_channel_p channel,
   UInt8 max_dlc,
   UInt32 number_of_buffered_frames);
```

Include file

`dslin.h`

Purpose

To initialize the receive monitor for the LIN frame response.

Parameters

channel Pointer to a LIN channel

max_dlc The greatest expected data length. In most cases dlc_max = 8 is a useful value.

number_of_buffered_frames The number of LIN responses the monitor can store.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The monitor is initialized successfully.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Channel 49

References

dslin_channel_rx_monitor_client_init 87
LIN Channel Handling 47
LIN Error Handling 26

dslin_channel_rx_monitor_clear

Syntax

```
enum DSLIN_ERROR dslin_channel_rx_monitor_clear(
   dslin_channel_p channel);
```

Include file

`dslin.h`

Purpose

To clear the receive monitor.

Parameters

channel Pointer to a LIN channel

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The monitor is cleared successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_ERR_CHANNEL_RXMONITOR_NOT_INIT`	The receive monitor is not initialized. Use `dslin_channel_rx_monitor_init` to initialize the monitor.

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Channel 49

References

dslin_channel_rx_monitor_init 85
LIN Channel Handling 47
LIN Error Handling 26

dslin_channel_rx_monitor_client_init

Syntax

```
enum DSLIN_ERROR dslin_channel_rx_monitor_client_init(
   dslin_channel_p channel,
   UInt32* client_number);
```

Include file

`dslin.h`

Purpose

To initialize a client to receive monitor data.

Parameters

channel Pointer to a LIN channel

client_number Returned reference number of the client. Use this reference number when reading from the receive monitor using the `dslin_channel_rx_monitor_client_read` function.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The client is initialized successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_ERR_CHANNEL_RXMONITOR_NOT_INIT`	The receive monitor is not initialized. Use `dslin_channel_rx_monitor_init` to initialize the monitor.

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Channel 49

References

dslin_channel_rx_monitor_client_read 88
dslin_channel_rx_monitor_init 85
LIN Channel Handling 47
LIN Error Handling 26

dslin_channel_rx_monitor_client_read

Syntax

```
enum DSLIN_ERROR dslin_channel_rx_monitor_client_read(
   dslin_channel_p channel,
   UInt32 client_number,
   dslin_channel_rx_data_t* data);
```

Include file

`dslin.h`

Purpose

To read a response from the receive monitor.

Description

The response and its status is copied in the data structure of the type `Data Structures: dslin_channel_rx_data_t`.

Parameters

channel Pointer to a LIN channel

client_number Reference number of the client. This number is returned by the `dslin_channel_rx_monitor_client_init` function.

data The response is copied in the `Data Structures: dslin_channel_rx_data_t` structure.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The response is read successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_ERR_CHANNEL_RXMONITOR_NOT_INIT`	The receive monitor is not initialized. Use `dslin_channel_rx_monitor_init` to initialize the monitor.
`DSLIN_NO_DATA_AVAILABLE`	The LIN board has not sent any data to the processor board since the last update.

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

Examples

References

dslin_channel_tx_response_write

Syntax

```
enum DSLIN_ERROR dslin_channel_tx_response_write(
   dslin_channel_p channel,
   dslin_channel_rx_data_t* data);
```

Include file

`dslin.h`

Purpose

To update one send response.

Description

This function can be used to configure TX frame responses.

Parameters

channel Pointer to a LIN channel

data Pointer to the frame response configuration with the `Data Structures: dslin_channel_rx_data_t` data type

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	Check on whether channel is in sleep or wake-up mode was successful.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.

Error Code	Meaning
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_ERR_RESPONSE_LENGTH_ILLEGAL	The value is invalid. The valid range for the response is within 0...255. The length of the response is specified in the header's identifier field.

Related topics

Basics

Setting Up a LIN Bus (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Channel.....49

References

Data Structures: dslin_channel_rx_data_t.....21
LIN Channel Handling.....47
LIN Error Handling.....26

dslin_channel_board_get

Syntax

```
enum DSLIN_ERROR dslin_channel_board_get(
   dslin_channel_p channel,
   dslin_board_p* board);
```

Include file

`dslin.h`

Purpose

To get the pointer to the LIN board used.

Description

The `dslin_channel_board_get` function allows you to get a pointer to the LIN board used. This is useful if you want to execute a board-specific function within a channel function, for example, if you want to perform a board update with the `dslin_board_update` function.

Parameters

channel Pointer to a LIN channel.

board Returned pointer to the LIN board.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The pointer to the board is returned successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERROR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_board_update.....42
dslin4330_board_init.....35
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_list_get

Syntax

```
enum DSLIN_ERROR dslin_channel_list_get(
   dslin_channel_p**  pChannelList,
   UInt32*  pSize);
```

Include file

`dslin.h`

Purpose

To return a list of LIN channels.

Description

The list contains all the LIN channels created via the `dslin_channel_create` function. The returned channels are available for further use according to the current application.

Parameters

pChannelList Pointer to the returned LIN channel list.

pSize Number of LIN channels

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The function has successfully returned the channel list.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if pChannelList or pSize == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Example

The example shows how to get a list of LIN channels.

```
int i;
UInt32 Size;
UInt32 IoType;
UInt8 IsUsed;
UInt32 Baudrate;
char Descriptor[32];
DSLIN_TRANSCEIVER_ENUM Transceiver;
DSLIN_TERMINATION_ENUM Termination;
dslin_channel_address_t ChannelAddress;
enum DSLIN_ERROR error = DSLIN_OK;
dslin_channel_p* pChannelList = 0;

error = dslin_channel_list_get(&pChannelList, &Size);

for(i=0;i<Size;i++)
{
dslin_channel_is_used(pChannelList[i], &IsUsed);
dslin_channel_transeiver_get(pChannelList[i], &Transceiver);
dslin_channel_termination_get(pChannelList[i], &Termination);
dslin_channel_baudrate_get(pChannelList[i], &Baudrate);
dslin_channel_descriptor_get(pChannelList[i], Descriptor, 32);
dslin_channel_io_address_get(pChannelList[i], &ChannelAddress);
dslin_channel_io_type_get(pChannelList[i], &IoType);
}
```

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_create..... 55
LIN Channel Handling..... 47
LIN Error Handling..... 26
Standard Defines..... 15

dslin_channel_descriptor_get

Syntax

```
enum DSLIN_ERROR dslin_channel_descriptor_get(
   dslin_channel_p  channel,
   char*  pDescriptor,
   UInt32  MaxLen);
```

Include file

`dslin.h`

Purpose

To get the name of a LIN channel.

Description

The function returns the name specified for the LIN channel via the `dslin_channel_create` function.

Parameters

channel Pointer to a LIN channel.

pDescriptor Pointer to the returned channel descriptor.

MaxLen Specifies the maximum length for the channel descriptor.
The length of the name specified by the `dslin_channel_create` function is limited to 63 characters.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the name of the LIN channel.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_create.....55
LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

dslin_channel_is_used

Syntax

```
enum DSLIN_ERROR dslin_channel_is_used(
   dslin_channel_p  channel,
   UInt8*  pIsUsed);
```

Include file

`dslin.h`

Purpose

To check whether a LIN channel is used (enabled).

Parameters

channel Pointer to a LIN channel.

pIsUsed Pointer to the state of the channel. Two values are available:

Value	Meaning
1	Indicates that the channel is used (enabled/started).
0	Indicates that the channel is not used (disabled/stopped).

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The check on whether the LIN channel is used (enabled) was successful.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

LIN Channel Handling 47
LIN Error Handling 26
Standard Defines 15

dslin_channel_io_address_get

Syntax

```
enum DSLIN_ERROR dslin_channel_descriptor_get(
   dslin_channel_p  channel,
   dslin_channel_address_t*  pChannelAddress);
```

Include file

`dslin.h`

Purpose

To get the LIN I/O address used.

Description

The function returns the physical channel number specified for the LIN channel via the `dslin_channel_create` function.

Parameters

channel Pointer to a LIN channel.

pChannelAddress Pointer to the returned LIN I/O address.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the LIN I/O address.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel or pDescriptor == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

dslin_channel_create..... 55
LIN Channel Handling..... 47
LIN Error Handling..... 26
Standard Defines..... 15

dslin_channel_io_type_get

Syntax

```
enum DSLIN_ERROR dslin_channel_io_type_get(
   dslin_channel_p  channel,
   UInt32*  pIoType);
```

Include file

`dslin.h`

Purpose

To get the board/module type of a LIN channel.

Description

This function returns the type of the I/O board or module used for the selected channel.

For the DS4330, the type is VCM_MID_DS4330.

Parameters

channel Pointer to a LIN channel.

pIoType Pointer to the returned board or module type.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The function has successfully returned the board/module type.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)
Setting Up a LIN Bus (DS4330 Features 🕮)

References

LIN Channel Handling.....47
LIN Error Handling.....26
Standard Defines.....15

LIN Node Handling

Introduction

The following functions and definitions are used to initialize and handle LIN nodes.

Where to go from here

Information in this section

Information in other sections

Example of Initializing a LIN Slave Node

Preconditions

You need the following information to initialize a LIN slave node:

- The handle or the name of the LIN channel (physical interface) on which the node is to be installed.
- The name of the LIN slave node (mostly from the LIN database file)

Example

The following example shows how to initialize a LIN slave node:

```
#include <dslin.h>
void lin_node_slave_init( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  dslin_node_p lin_node;
```

```
// Look if there is a node with that name installed on appropriate LIN channel.
  error = dslin_node_parity_offset_set("LIN Node0","LIN Interface0",&lin_node );
  dslin_node_error_print( lin_node, error );
// Terminate the application (Not really an error handling!)
  if( DSLIN_OK != error )
  {
    exit(1);
  }
// Create the handle for the LIN node.
  error = dslin_node_create( lin_channel, "LIN Node0", &lin_node );
  dslin_node_error_print( lin_node, error );
// Terminate the application (Not really an error handling!
  if( DSLIN_OK != error )
  {
    exit(1);
  }
// Initialize the LIN node.
  error = dslin_node_init( lin_node, DSLIN_NODE_SLAVE, 64, 64 );
  dslin_node_error_print( lin_node, error );
// Terminate the application (Not really an error handling!)
  if( DSLIN_OK != error )
  {
    exit(1);
  }
  error = dslin_node_enable( lin_node );
  dslin_node_error_print( lin_node, error );
// Terminate the application (Not really an error handling!)
  if( DSLIN_OK != error )
  {
    exit(1);
  }
}
```

Related topics

References

dslin_node_lookup

Syntax

```
enum DSLIN_ERROR dslin_node_lookup(
   const char* node_name,
   const char* channel_name,
   dslin_node_p* node);
```

Include file

`dslin.h`

Purpose

To search for an existing LIN node.

Note

- The search is limited to the selected LIN channel.
- Use the `dslin_node_lookup` function only during the initialization phase of the system.

Parameters

node_name Name of node that is searched for. The name is limited to 63 characters.

channel_name Name of the LIN channel on which the node is searched for.

node Returned pointer to the LIN node.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The node searched for was found.
DSLIN_NODE_NOT_FOUND	There is no node with the specified name installed on this channel.

Example

The following example shows how to search for a LIN node. For a detailed example of LIN node handling, refer to Example of Initializing a LIN Slave Node on page 99.

```
enum DSLIN_ERROR error = DSLIN_OK;
dslin_node_p lin_node = NULL;
// Look if there is a node with that name
// installed on the appropriate LIN channel.
error = dslin_node_lookup( "LIN Node0","LIN Interface0", &lin_node );
if( error == DSLIN_ERR_NODE_NOT_FOUND )
{
// There is no LIN node with the name "LIN Node0" working
// on the LIN channel with the name "LIN Interface0".
}
if( error == DSLIN_OK )
{
// We got it!
}
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

dslin_node_create.....102
dslin_node_error_print.....106
dslin_node_init.....104
LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_create

Syntax

```
enum DSLIN_ERROR dslin_node_create(
   dslin_channel_p channel,
   const char* name,
   dslin_node_p* node);
```

Include file

`dslin.h`

Purpose

To create a LIN node.

Note

Use the `dslin_node_create` function only during the initialization phase of the system.

Description

The `dslin_node_create` function allocates memory for a new LIN node. If there is already a node with the same name connected to the channel, the pointer to the node already created is returned. A newly created channel is disabled and configured as slave node by default. If an existing node is returned the enabling state depends on the reused node.

Parameters

channel Pointer to a LIN channel.

name Name of the node. The name is limited to 63 characters. If the string is NULL or empty (" "), the LIN node cannot be found with `dslin_node_parity_offset_set`.

node Returned pointer to the LIN node. Returns NULL if the function fails.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new node has been created successfully.
`DSLIN_OBJECT_REUSED`	A pointer to an existing node with the same name on the same channel was returned.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_ERR_MALLOC`	Memory allocation error. There is not enough memory available on the master processor board.
`DSLIN_ERR_NODE_COUNT`	Illegal node number. The number of LIN nodes on a channel is limited to 16.

Example

The example shows how to create a LIN node. For a detailed example of LIN node handling, refer to Example of Initializing a LIN Slave Node on page 99.

```
enum DSLIN_ERROR error = DSLIN_OK;
dslin_node_p lin_node = NULL;
// Create the handle for the LIN node.
error = dslin_node_create( lin_channel, "LIN-Node0", &lin_node );
dslin_node_error_print( lin_node, error );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_error_print 106
dslin_node_init 104
dslin_node_parity_offset_set 110

dslin_node_init

Syntax

```
enum DSLIN_ERROR dslin_node_init(
   dslin_node_p node,
   enum DSLIN_NODE_TYPE type,
   UInt8 tx_error_threshold,
   UInt8 rx_error_threshold);
```

Include file

`dslin.h`

Purpose

To initialize and configure a LIN node.

Note

- Use the `dslin_node_init` function only during the initialization of the system.
- A master node always has an internal slave task. This means a master node can receive and transmit responses like a slave node.
- After initialization, the node is enabled by default.
- The `dslin_node_init` function resets all node error counters.

Description

The function specifies the type of the node and resets all its error counters. Currently only slaves are available. The rx error threshold is used for error handling. You can also set the threshold value with the `dslin_node_rx_error_threshold_set` function. For detailed information on error handling, see LIN Error Handling on page 26.

Parameters

node Pointer to a LIN node

type Specifies the type of the LIN node:

Symbol	Meaning
`DSLIN_NODE_SLAVE`	The node acts as a slave node.
`DSLIN_NODE_MASTER`	The node acts as a master node.

tx_error_threshold Specifies the initialization value of the `tx_error_threshold` parameter in the range 0 … 255. You can change the

value with the `dslin_node_tx_error_threshold_set` function. The transmit error counter can trigger an interrupt when the error threshold is exceeded. The TX error counter is only available for a master node.

Note

The TX error threshold is related to the TX error counter, which is only provided by a LIN master node.

rx_error_threshold Specifies the initialization value of the `rx_error_threshold` parameter in the range 0 … 255. You can change the value with the `dslin_node_rx_error_threshold_set` function. The receive error counter can trigger an interrupt when the error threshold is exceeded. The RX error counter is available for master and slave nodes.

Note

The interrupts must be explicitly enabled with DSLIN_NODE_INT_TX_ERROR_THRESHOLD_ EXCEEDED or DSLIN_NODE_INT_RX_ERROR_THRESHOLD_ EXCEEDED. See dslin_node_interrupt_init on page 235.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The node has been initialized successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_NODE_MASTER_ALREADY_PRESENT`	Only one master node per LIN bus is allowed.
`DSLIN_ERR_NODE_TYPE_ILLEGAL`	An illegal node type was defined. Currently only slave nodes are available.

Example

The example shows how to initialize a LIN node. For a detailed example of LIN node handling, refer to Example of Initializing a LIN Slave Node on page 99.

```
enum DSLIN_ERROR error = DSLIN_OK;
dslin_node_p lin_node = NULL;
// Initialize the LIN node.
error = dslin_node_init( lin_node, DSLIN_NODE_SLAVE, 64, 64);
dslin_node_error_print( lin_node, error );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_node_create 102
dslin_node_error_print 106
dslin_node_parity_offset_set 110
dslin_node_rx_error_threshold_set 109
dslin_node_tx_error_threshold_set 107
LIN Node Handling 97
Standard Defines 15

dslin_node_error_print

Syntax

```
define dslin_node_error_print(
   dslin_node_p node,
   enum DSLIN_ERROR error);
```

Include file

`dslin.h`

Purpose

To report errors to the dSPACE log file.

If `error==DSLIN_OK`, nothing is written to the log file.

Note

- Reporting the error information to the log file is a time-consuming process. Consider this when using the function within your task.
- The dSPACE log file can be opened in the dSPACE experiment software.

Parameters

node Pointer to a LIN node

error Error code to be written to the log file as plain text.

Return value

None

Example

For examples on how to use the `dslin_node_error_print` function, refer to Example of Initializing a LIN Frame to Receive and Transmit a Response on page 181 and Example of Reading Response Data on page 182.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node..... 99

References

dslin_node_create..... 102
dslin_node_init..... 104
dslin_node_parity_offset_set..... 110
LIN Error Handling..... 26
Standard Defines..... 15

dslin_node_tx_error_threshold_set

Syntax

```
enum DSLIN_ERROR dslin_node_tx_error_threshold_set(
   dslin_node_p master_node,
   UInt8 threshold);
```

Include file

`dslin.h`

Purpose

To set the transmit error threshold for the selected LIN node.

Note

- The settings specified by the `dslin_node_tx_error_threshold_set` function have to be applied with the `dslin_node_apply_settings` function.
- The default value for the threshold is 64.
- The `dslin_node_tx_error_threshold_set` function is related to a LIN master node. Only a LIN master node provides a TX error counter.

Description

If an error occurs during transmission a message, the tx error counter is increased (+8). If no error occurs it is decreased (–1). If the number of errors reaches the threshold an interrupt can be triggered. Refer to `dslin_node_interrupt_init` on page 235.

Parameters

master_node Pointer to a LIN master node

threshold Specifies the threshold in the range 0 … 255. "0" disables the check for the threshold and also the interrupt generation in the event of an exceeded threshold.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The threshold was successfully set.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

dslin_node_interrupt_init.....235
dslin_node_tx_error_count_get.....119
LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_rx_error_threshold_set

Syntax

```
enum DSLIN_ERROR dslin_node_rx_error_threshold_set(
   dslin_node_p node,
   UInt8 threshold);
```

Include file

`dslin.h`

Purpose

To set the receive error threshold for the selected LIN node.

Note

- The settings specified by the `dslin_node_rx_error_threshold_set` function have to be applied with the `dslin_node_apply_settings` function.
- The default value for the threshold is 64.

Description

The rx error counter has a threshold that can be used to trigger events if the number of errors exceeds the threshold. If an error occurs in receiving a frame, the rx error counter is increased (+8). If no error occurs it is decreased (–1). For detailed information, see LIN Error Handling on page 26.

Parameters

node Pointer to a LIN node

threshold Specifies the threshold in the range 0 … 255. "0" disables the check for the threshold and also the interrupt generation in the event of an exceeded threshold.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The threshold has been set successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

dslin_node_apply_settings.....111
dslin_node_info_rx_err_get.....139
dslin_node_rx_error_count_get.....118
LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_parity_offset_set

Syntax

```
enum DSLIN_ERROR dslin_node_parity_offset_set(
   dslin_node_p master_node,
   UInt8 identifier,
   UInt8 offset);
```

Include file

`dslin.h`

Purpose

To specify the parity offset of the identifier.

Note

- The settings specified by the `dslin_node_parity_offset_set` function have to be applied with the `dslin_node_apply_settings` function.
- The `dslin_node_parity_offset_set` function is related to a LIN master node. Only a master node can send a frame header.

Description

You can use the `dslin_node_parity_offset_set` function to simulate a wrong parity in the header. The ID parity bits are part of the identifier field which is used to denote the content and the length of a frame.

Parameters

master_node Pointer to a LIN master node

identifier Specifies the header to which the offset must be applied.

offset Specifies the offset. The offset is added to the parity before transmitting the header.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The threshold has been set successfully.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_ERR_NODE_TYPE_ILLEGAL	The node has the wrong type. The command is only valid for a master node.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

dslin_node_apply_settings.....111
LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_apply_settings

Syntax

```
enum DSLIN_ERROR dslin_node_apply_settings(
   dslin_node_p node);
```

Include file

`dslin.h`

Purpose

To apply the settings for the node. The function transfers the data to the slave processor of the LIN board.

Note

After you have specified the settings with the corresponding dslin_xxxx_set functions, you can call the `dslin_node_apply_settings` function once to transfer all settings to the slave processor of the LIN board.

Parameters

node Pointer to a LIN node

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The settings have been transferred successfully to the slave processor.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_init 104
LIN Error Handling 26
LIN Node Handling 97
Standard Defines 15

dslin_node_enable

Syntax

```
enum DSLIN_ERROR dslin_node_enable(
   dslin_node_p node);
```

Include file

`dslin.h`

Purpose

To enable the selected LIN node.

Description

The `dslin_node_enable` function allows you to enable a disabled LIN node. In HIL simulation this is useful to switch on and off different ECUs and test them without changing the hardware configuration.

Parameters

node Pointer to a LIN node

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The node is enabled.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

dslin_node_disable.....114
LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_disable

Syntax

```
enum DSLIN_ERROR dslin_node_disable(
   dslin_node_p node);
```

Include file

`dslin.h`

Purpose

To disable the selected LIN node.

Description

The `dslin_node_disable` function allows you to disable an enabled LIN node. In HIL simulations this is useful to switch on and off different ECUs and test them without changing the hardware configuration.

Parameters

node Pointer to a LIN node

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The node is disabled.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Error Code	Meaning
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

dslin_node_enable.....113
LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_command_wakeup

Syntax

```
enum DSLIN_ERROR dslin_node_command_wakeup(
   dslin_node_p node);
```

Include file

`dslin.h`

Purpose

To send a wake-up byte over the LIN bus.

Description

If the bus is in sleep mode the `dslin_node_command_wakeup` function terminates the sleep mode and wakes up the bus by sending the corresponding wake-up byte (0x80). A wake-up signal can be sent by any slave node on the bus.

Parameters

node Pointer to a LIN node

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The wake-up byte was sent successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node..... 99

References

dslin_node_command_sleep..... 116
LIN Error Handling..... 26
LIN Node Handling..... 97
Standard Defines..... 15

dslin_node_command_sleep

Syntax

```
enum DSLIN_ERROR dslin_node_command_sleep(
   dslin_node_p master_node);
```

Include file

`dslin.h`

Purpose

To send a sleep command over the LIN bus.

Note

The `dslin_node_command_sleep` function is related to a master node. Only a master node can send the go to sleep command.

Description

The `dslin_node_command_sleep` function is used by the master to broadcast the sleep mode to all bus nodes. There is no more bus activity after completion of this message until a wake-up signal on the bus ends the sleep mode. The sleep mode command is a download command frame with 0x00 as the first data field. For detailed information, refer to LIN specification 2.0.

Parameter

master_node Pointer to a LIN master node

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The node is disabled.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_NODE_TYPE_ILLEGAL`	The node has the wrong type. The command is only valid for a master node.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_channel_is_wake 83
dslin_node_command_wakeup 115
LIN Error Handling 26
LIN Node Handling 97
Standard Defines 15

dslin_node_rx_error_count_get

Syntax

```
enum DSLIN_ERROR dslin_node_rx_error_count_get(
   dslin_node_p node,
   UInt8 identifier,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

`dslin.h`

Purpose

To get the receive error count for the specific frame identifier on the selected node.

Note

This counter has no threshold and is never decreased. Use the `dslin_node_init` function to reset the counter to 0.

Description

The counter provides the total count of receive errors for a specific LIN response. It is increased by 1 in the following cases:

- A bit error in a data or checksum field in reading back a LIN response sent by the node.
- A slave-not-responding error when expecting or reading a response from the bus.
- A checksum error when reading a response from the bus.
- A framing error when reading a response from the bus.

Parameters

node Pointer to a LIN node

identifier Identifier of the frame

value Address where the returned value of the rx error counter is stored.

timestamp Address where the value of the time stamp of the last increment action is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The function has successfully returned the count value.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_info_rx_err_get 139
dslin_node_rx_error_threshold_set 109
LIN Error Handling 26
LIN Node Handling 97
Standard Defines 15

dslin_node_tx_error_count_get

Syntax

```
enum DSLIN_ERROR dslin_node_tx_error_count_get(
   dslin_node_p master_node,
   UInt8* identifier,
   UInt32* value,
   dsfloat* termination);
```

Include file

dslin.h

Purpose

To get the TX error count of the selected node.

> **Note**
>
> The `dslin_node_tx_error_count_get` function is related to a LIN master node. Only a master node provides a TX error counter.

Description

The TX error counter is used to count errors that occur during the transmission of messages. The error counter is increased if an error occurs and decreased if the message was sent without error. When the counter exceeds the TX error threshold, the application can trigger an interrupt. For information on how to set the threshold, refer to `dslin_node_tx_error_threshold_set` on page 107.

Parameters

master_node Pointer to a LIN master node

identifier The `identifier` parameter is not valid for slave nodes.

value Address where the returned value of the TX error counter is stored.

termination Address where the time stamp of the last increment is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the count value.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node..... 99

References

dslin_node_rx_error_count_get..... 118

dslin_node_initial_nad_set

Syntax

```
enum DSLIN_ERROR dslin_node_initial_nad_set(
   dslin_node_p node,
   UInt8 nad);
```

Include file

`dslin.h`

Purpose

To set the initial NAD (node address for diagnostics).

> **Note**
>
> The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Description

The node starts with the initial node address, so the address must be fixed. The NAD is used in a request to address a slave node. The NAD is also used to indicate the source of a response.

Parameters

node Pointer to a LIN node

nad Specifies a diagnostic node address in the range 1 … 126.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully set the initial NAD.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node..... 99

References

dslin_node_initial_nad_get..... 122
LIN Error Handling..... 26
LIN Node Handling..... 97
Standard Defines..... 15

dslin_node_initial_nad_get

Syntax

```
enum DSLIN_ERROR dslin_node_initial_nad_get(
   dslin_node_p node,
   UInt8* nad);
```

Include file

`dslin.h`

Purpose

To get the initial NAD (node address for diagnostics).

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Parameters

node Pointer to a LIN node

nad Pointer to the returned initial node address

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the initial NAD.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_initial_nad_set 121
LIN Error Handling 26
LIN Node Handling 97
Standard Defines 15

dslin_node_current_nad_set

Syntax

```
enum DSLIN_ERROR dslin_node_current_nad_set(
   dslin_node_p node,
   UInt8 nad);
```

Include file

`dslin.h`

Purpose

To set the current NAD (node address for diagnostics).

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Description

The current NAD is in most cases identical to the initial NAD. You can change the current NAD by the requests "Assign NAD" and "Conditional change NAD".

Parameters

node Pointer to a LIN node

nad Specifies a NAD with the range 1...126.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully set the NAD.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

dslin_node_current_nad_get.....124
LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_current_nad_get

Syntax

```
enum DSLIN_ERROR dslin_node_current_nad_get(
   dslin_node_p node,
   UInt8* nad);
```

Include file

`dslin.h`

Purpose

To get the current NAD (node address for diagnostics).

> **Note**
>
> The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Parameters

node Pointer to a LIN node

nad Pointer to the returned NAD

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the NAD.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_current_nad_set 123
LIN Error Handling 26
LIN Node Handling 97
Standard Defines 15

dslin_node_supplier_id_set

Syntax

```
enum DSLIN_ERROR dslin_supplier_id_set(
   dslin_node_p node,
   UInt16 supplier_id);
```

Include file

`dslin.h`

Purpose

To set the supplier ID.

> **Note**
>
> The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Description

The supplier ID is part of the LIN product identification. The LIN Consortium assigns each supplier an ID.

Parameters

node Pointer to a LIN node

supplier_id Specifies a supplier ID in the range 0 … 0x7FFF.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully set the supplier ID.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_supplier_id_get 127
LIN Error Handling 26
LIN Node Handling 97
Standard Defines 15

dslin_node_supplier_id_get

Syntax

```
enum DSLIN_ERROR dslin_node_supplier_id_get(
   dslin_node_p node,
   UInt16* supplier_id);
```

Include file

`dslin.h`

Purpose

To get the supplier ID.

> **Note**
>
> The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Parameters

node Pointer to a LIN node

supplier_id Pointer to the returned supplier ID.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the supplier ID.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_supplier_id_set 125
LIN Error Handling 26
LIN Node Handling 97
Standard Defines 15

dslin_node_function_id_set

Syntax

```
enum DSLIN_ERROR dslin_function_id_set(
   dslin_node_p node,
   UInt8 function_id);
```

Include file

`dslin.h`

Purpose

To set the function ID.

> **Note**
>
> The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Description

The function ID is part of the LIN product identification. The function ID is assigned by each supplier. If two products have different functionalities, their function IDs must differ.

Parameters

node Pointer to a LIN node

function_id Specifies a function ID in the range 0 … 0xFFFF.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully set the function ID.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node..... 99

References

dslin_node_function_id_get..... 129
LIN Error Handling..... 26
LIN Node Handling..... 97
Standard Defines..... 15

dslin_node_function_id_get

Syntax

```
enum DSLIN_ERROR dslin_node_function_id_get(
   dslin_node_p node,
   UInt16* function_id);
```

Include file

`dslin.h`

Purpose

To get the function ID.

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Parameters

node Pointer to a LIN node

function_id Pointer to the returned function ID.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the function ID.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_function_id_set 128
LIN Error Handling 26
LIN Node Handling 97
Standard Defines 15

dslin_node_variant_id_set

Syntax

```
enum DSLIN_ERROR dslin_variant_id_set(
   dslin_node_p node,
   UInt8 variant_id);
```

Include file

`dslin.h`

Purpose

To set the variant ID.

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Description

The variant ID can be assigned by each supplier as a part of the LIN product identification. The variant ID must be changed whenever the product is changed but not its functionality.

Parameters

node Pointer to a LIN node

variant_id Specifies a variant ID in the range 0 … 0xFF.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully set the variant ID.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_variant_id_get 131
LIN Error Handling 26
LIN Node Handling 97
Standard Defines 15

dslin_node_variant_id_get

Syntax

```
enum DSLIN_ERROR dslin_node_variant_id_get(
   dslin_node_p node,
   UInt8* variant_id);
```

Include file

`dslin.h`

Purpose

To get the variant ID.

> **Note**
>
> The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Parameters

node Pointer to a LIN node

variant_id Pointer to the returned variant ID.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the variant ID.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_variant_id_set 130
LIN Error Handling 26
LIN Node Handling 97
Standard Defines 15

dslin_node_readbyid_positive_response_set

Syntax

```
enum DSLIN_ERROR dslin_node_readbyid_positive_response_set(
   dslin_node_p node,
   UInt8 id,
   UInt8* response);
```

Include file

`dslin.h`

Purpose

To set the positive response for the "read by identifier" request.

> **Note**
>
> The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Description

It is possible to read the supplier identity and other properties from a slave node using the "read by identifier" request. A response is sent only if the NAD, the supplier ID, and the function ID match. For more information, refer to LIN specification 2.0.

Parameters

node Pointer to a LIN node

id Selects the row in the table of positive responses. The valid values are 0, 1 and values in the range 16 ... 31.

response Response array with a length of 7 bytes which is sent for the "read by identifier" request. You can define the response array yourself.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully set the positive response.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Example

The following example shows how you can use the function generally:

```
UInt8 response[7];
response[0] = PCI;
response[1] = RSID;
response[2] = D1;
response[3] = D2;
response[4] = D3;
response[5] = D4;
response[6] = D5;
dslin_node_readbyid_positive_response_set(node, id, response);
```

The following example shows how to set the positive response field with the ID = 0:

```
UInt8 response[7];
UInt16 supplier_id = 0x0123;
UInt16 function_id = 0x4567;
UInt9 variant_id = 0x89;
response[0] = 0x06; //PCI
response[1] = 0xF2; //RSID
response[2] = 0xFF & supplier_id; //D1
response[3] = supplier_id >> 8; //D2
response[4] = 0xFF & function_id; //D3
response[5] = function_id >> 8; //D4
response[6] = variant_id; //D5
dslin_node_readbyid_positive_response_set(node, 0, response);
```

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_readbyid_positive_response_get 134
LIN Error Handling 26
LIN Node Handling 97
Standard Defines 15

dslin_node_readbyid_positive_response_get

Syntax

```
enum DSLIN_ERROR dslin_node_readbyid_positive_response_get(
   dslin_node_p node,
   UInt8 id,
   UInt8* response);
```

Include file

`dslin.h`

Purpose

To return the positive response for the "read by identifier" request.

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Parameters

node Pointer to a LIN node

id Selects the row in the table of positive responses. The range is 0, 1 and 16 … 31.

response Returns the response array with a length of 7 bytes which is sent for the "read by identifier" request. You can define the response array yourself.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the positive response.
`DSLIN_NO_DATA_AVAILABLE`	The error code can be returned by the first read or get operation. It is a message and indicates no error.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node..... 99

References

dslin_node_readbyid_positive_response_set..... 132
LIN Error Handling..... 26
LIN Node Handling..... 97
Standard Defines..... 15

dslin_node_configuration_init

Syntax

```
enum DSLIN_ERROR dslin_node_configuration_init(
   dslin_node_p node);
```

Include file

`dslin.h`

Purpose

To initialize the node configuration services for a slave node.

> **Note**
>
> The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Description

This function configures a slave node which can receive the frame ID 0x3c and transmit a response with the frame ID0x3d. The received frame is named "MasterReqRx" and the transmitted frame is named "SlaveRespTx".

After initialization, the node configuration service uses these frames to handle the node configuration requests from a LIN master node. Refer to `dslin_node_configuration_service` on page 137.

Parameters

node Pointer to a LIN node

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully initialized the configuration service.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_ERR_MALLOC`	Memory allocation error on the processor board. There is not enough memory available on the processor board.
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. The error indicates that the LIN board is overloaded with command processing.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_configuration_service

Syntax

```
enum DSLIN_ERROR dslin_node_configuration_service(
   dslin_node_p node);
```

Include file

`dslin.h`

Purpose

To process the requests from the LIN master node.

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Description

Use this function after receiving the response of the master request frame (ID: 0x3c). The best way to call the function is a task which is called when the response is received using the `dslin_node_frame_interrupt_init` function. This ensures that the service for the node configuration is executed directly after receiving the master request. The service for the node configuration prepares internally the slave response (IS: 0x3d).

The preconditions for a working node configuration are:

- An initial node address is set using `dslin_node_initial_nad_set`,
- A current node address is set using `dslin_node_current_nad_set`,
- The node is initialized using `dslin_node_configuration_init`.

If one of the preconditions is not fulfilled, the `dslin_node_configuration_service` function issues this error DSLIN_ERR_NODE_CONF_NOT_INIT.

Parameter

node Pointer to a LIN node

Return value

The function returns the following error code:

Error Code	Meaning
DSLIN_OK	The function was successfully.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_ERR_NODE_CONF_NOT_INIT	The node configuration is not completely initialized.
DSLIN_ERR_NODE_CONF_SID_NOT_SUPPORTED	The service identifier (SID) is not supported. The node configuration service detected an unsupported service identifier (SID).

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node..... 99

References

dslin_node_configuration_init..... 136
dslin_node_current_nad_set..... 123
dslin_node_frame_interrupt_init..... 238
dslin_node_initial_nad_set..... 121
LIN Error Handling..... 26
LIN Node Handling..... 97
Standard Defines..... 15

dslin_node_info_rx_err_get

Syntax

```
enum DSLIN_ERROR dslin_node_info_rx_err_get(
   dslin_node_p node,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

`dslin.h`

Purpose

To read the rx error counter for LIN responses.

Description

The rx error counter is increased by 8 in the following cases:

- A bit error in a data or checksum field occurs in reading back a LIN response sent by the node.
- A slave-not-responding error occurs when expecting or reading a response from the bus.
- A checksum error occurs when reading a response from the bus.

The rx error counter is decreased by 1 (not below 0) each time a response is received properly. For detailed information, see LIN Error Handling on page 26.

Note

- The valid range for the error counter value is within 0 … 255. If no interrupt is defined the error counter is not reset. But keep in mind that the counter cannot exceed the 255 value. If more errors occur the counter value remains at 255.
- The `dslin_node_init` function resets the counter to 0.

Parameters

node Pointer to a LIN node

value Address where the value of the rx error counter is stored.

timestamp Address where the value of the time stamp of the last increment or decrement action is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The function has successfully returned the count value.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node 99

References

dslin_node_info_tx_err_get 140
dslin_node_rx_error_threshold_set 109
LIN Node Handling 97
Standard Defines 15

dslin_node_info_tx_err_get

Syntax

```
enum DSLIN_ERROR dslin_node_info_tx_err_get(
   dslin_node_p master_node,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

dslin.h

Purpose

To read the TX error counter for LIN headers.

Note

- The valid range for the error counter value is within 0 … 255. If no interrupt is defined the error counter is not reset. But keep in mind that the counter cannot exceed the maximum value. If more errors occur the counter value remains at 255.
- The `dslin_node_init` function resets the counter to "0". See `dslin_node_init` on page 104.

Description

The TX error counter is increased by 8 each time the transmitted header has a bit error. It is decreased by 1 (not below 0) each time the header is read back properly.

Parameters

master_node Pointer to a LIN master node

value Address where the value of the TX error counter is stored.

timestamp Address where the value of the time stamp of the last increment or decrement action is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the count value.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

dslin_node_info_rx_err_get.....139
dslin_node_rx_error_count_get.....118
dslin_node_tx_error_threshold_set.....107
LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_info_reset

Syntax

```
enum DSLIN_ERROR dslin_node_info_reset(
   dslin_node_p node);
```

Include file

`dslin.h`

Purpose

To reset the node error counters to 0.

Parameters

node Pointer to the LIN node.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the count value.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_node_info_rx_err_get 139
dslin_node_info_tx_err_get 140
LIN Error Handling 26
Standard Defines 15

dslin_node_info_checksum_err_get

Syntax

```
enum DSLIN_ERROR dslin_node_info_checksum_err_get(
   dslin_node_p node,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

`dslin.h`

Purpose

To get the total number of checksum errors.

Note

The counter is only incremented, never decremented or reset. Use the `dslin_node_init` function to reset the counter to `"0"`.

Parameters

node Pointer to a LIN node

value Address where the number of the checksum errors is stored.

timestamp Address where the value of the time stamp of the last increment action is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the count value.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node is NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

LIN Error Handling.....26
Standard Defines.....15

dslin_node_info_bit_err_get

Syntax

```
enum DSLIN_ERROR dslin_node_bit_err_get(
   dslin_node_p node,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

`dslin.h`

Purpose

To get the number of bit errors.

Note

The counter is only incremented, never decremented or reset. Use the `dslin_node_init` function to reset the counter to "0".

Description

A bit error is detected by the sending node if the bit value that is monitored is different from the bit value that was sent.

Parameters

node Pointer to a LIN node

value Address where the number of the bit errors is stored.

timestamp Address where the value of the time stamp of the last increment action is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The function has successfully returned the count value.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_info_idpar_err_get

Syntax

```
enum DSLIN_ERROR dslin_node_info_idpar_err_get(
   dslin_node_p node,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

`dslin.h`

Purpose

To get the number of ID parity errors.

Note

The counter is only incremented, never decremented or reset. Use the `dslin_node_init` function to reset the counter to "0".

Description

The ID parity error is detected when a slave node receives a wrong parity bit in a header.

Parameters

node Pointer to a LIN node

value Address where the number of the parity ID errors is stored.

timestamp Address where the value of the time stamp of the last increment action is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the count value.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_info_framing_err_get

Syntax

```
enum DSLIN_ERROR dslin_node_info_framing_err_get(
   dslin_node_p node,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

`dslin.h`

Purpose

To get the total number of framing errors.

Description

The framing error is an error in the response field of a received LIN frame. A framing error indicates a bus collision.

Parameters

node Pointer to a LIN node

value Address where the returned number of framing errors is stored.

timestamp Address where the value of the last framing error's time stamp is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the count value.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access. It occurs if node == NULL. It can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Related topics

Basics

LIN Bus Handling (DS4330 Features 🕮)

References

LIN Node Handling 97
Standard Defines 15

dslin_node_info_header_bit_err_get

Syntax

```
enum DSLIN_ERROR dslin_node_info_header_bit_err_get(
   dslin_node_p master_node,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

`dslin.h`

Purpose

To read the number of bit errors for all transmitted LIN headers.

Note

The `dslin_node_info_header_bit_err_get` function is related to a LIN master node. Only a master node can send a frame header.

Description

The bit error counter is increased by 1 each time a sent header has a bit error

Parameters

master_node Pointer to a LIN master node

value Address where the value of the returned bit error count is stored.

timestamp Address where the time stamp of the last error increment is stored.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the count value.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_info_no_bus_activity_err_get

Syntax

```
enum DSLIN_ERROR dslin_node_info_no_bus_activity_err_get(
   dslin_node_p node,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

`dslin.h`

Purpose

To get the number of detected no-bus-activity errors.

Note

The counter is only incremented, never decremented or reset. Use the `dslin_node_init` function to reset the counter to "0".

Description

The error "no bus activity" is issued if for 25000 bit times no bus traffic is registered.

Parameters

node Pointer to a LIN node

value Address where the number of the "no bus activity" errors is stored.

timestamp Address where the time stamp of the last error increment is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The function has successfully returned the count value.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

dslin_node_init.....104
LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_info_snr_err_get

Syntax

```
enum DSLIN_ERROR dslin_node_info_snr_err_get(
   dslin_node_p node,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

`dslin.h`

Purpose

To get the number of slave-not-responding errors.

> **Note**
>
> The counter is only incremented, never decremented or reset. Use the `dslin_node_init` function to reset the counter to "0".

Description

A slave-not-responding error is detected when a slave node expects a message from another slave node (depending on the identifier) but no valid message appears on the bus within the time allowed to transmit a message frame (T_{FRAME_MAX}). The maximum respond time is (N · 10 + 44) · 1.4 bit times to a certain header (N = Number of bytes).

For more information on T_{FRAME_MAX}, refer to *LIN specification 1.2*. When a slave does not expect a message it does not need to detect this error.

Parameters

node Pointer to a LIN node

value Address where the number of slave-not-responding errors is stored.

timestamp Address where the value of the time stamp of the last increment action is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the count value.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_info_synch_err_get

Syntax

```
enum DSLIN_ERROR dslin_node_info_synch_err_get(
   dslin_node_p node,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

`dslin.h`

Purpose

To get the number of inconsistent-synchronization-field errors.

Note

The counter is only incremented, never decremented or reset. Use the `dslin_node_init` function to reset the counter to "0".

Description

An inconsistent-synchronization-field error is detected when a slave node receives a synchronization byte in a header different from 0x55. Such errors usually occur when the transceiver and receiver are not working with the same baud rates.

Parameters

node Pointer to a LIN node

value Address where the number of the synch field errors is stored.

timestamp Address where the value of the time stamp of the last increment action is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The function has successfully returned the count value.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

LIN Error Handling.....26
LIN Node Handling.....97
Standard Defines.....15

dslin_node_info_extrabytes_err_get

Syntax

```
enum DSLIN_ERROR dslin_node_info_extrabytes_err_get(
   dslin_node_p node,
   UInt32* value,
   dsfloat* timestamp);
```

Include file

dslin.h

Purpose

To get the number of extrabyte errors.

Note

The counter is only incremented, never decremented or reset. Use the `dslin_node_init` function to reset the counter to "0".

Description

Extrabyte errors are caused if the receive length of the RX frame is shorter than the length of the TX frame with the same identifier on the LIN bus. Further reasons are, for example:

- If a wake-up byte is sent by another node and the LIN bus is not in sleep mode, the byte is not interpreted as a wake-up byte but as an extrabyte.
- Interference on the LIN bus

Parameters

node Pointer to a LIN node

value Address where the number of extrabyte errors is stored.

timestamp Address where the time stamp of the last increment is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the count value.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_node_init..... 104
LIN Error Handling..... 26
LIN Node Handling..... 97
Standard Defines..... 15

dslin_node_board_get

Syntax

```
enum DSLIN_ERROR dslin_node_board_get(
   dslin_node_p node,
   dslin_board_p* board);
```

Include file

`dslin.h`

Purpose

To get the pointer to the LIN board used.

Description

The `dslin_node_board_get` function allows you to get a pointer to the LIN board used. That is useful if you want to execute a board-specific function within a node function, for example, if you want to perform a board update with the `dslin_board_update` function.

Parameters

node Pointer to a LIN node

board Returned pointer to the LIN board.

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the board settings.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node.....99

References

dslin_board_update.....42
LIN Error Handling.....26

dslin_node_channel_get

Syntax

```
enum DSLIN_ERROR dslin_node_channel_get(
   dslin_node_p node,
   dslin_channel_p* channel);
```

Include file

`dslin.h`

Purpose

To get the pointer to the LIN channel used.

Description

The `dslin_node_channel_get` function allows you to get a pointer to the LIN channel used. This is useful if you want to execute a channel-specific function within a node function, for example, if you want to restore the settings of a channel with the `dslin_channel_restore_settings` function.

Parameters

node Pointer to a LIN node

channel Pointer to the returned LIN channel

Return value

The function returns the following error code:

Error Code	Meaning
`DSLIN_OK`	The function has successfully returned the channel settings.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Defining LIN Nodes (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Slave Node..... 99

References

dslin_channel_restore_settings..... 78
LIN Error Handling..... 26
LIN Node Handling..... 97
Standard Defines..... 15

LIN Schedule Handling

Introduction

The following functions are used to implement LIN schedules in LIN applications.

Where to go from here

Information in this section

dslin_schedule_error_print

Syntax

```
define dslin_schedule_error_print(
   schedule,
   error);
```

Include file

`dslin.h`

Purpose

To write errors to the dSPACE log file. If `error==DSLIN_OK` nothing is written to the output.

Note

- Reporting the error information to the log file is a time-consuming process. Consider this when using the function within your task.
- The dSPACE log file can be opened in the dSPACE experiment software.

Parameters

schedule Pointer to a LIN schedule

error Error code to be written to the log file as plain text.

Return value

None

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_schedule_lookup

Syntax

```
enum DSLIN_ERROR dslin_schedule_lookup(
   const char* schedule_name,
   const char* node_name,
   const char* channel_name,
   dslin_schedule_p* schedule);
```

Include file

`dslin.h`

Purpose

To search for an existing LIN schedule.

Note

- The search is limited by the selected LIN node and LIN channel.
- Use the `dslin_schedule_lookup` function only during the initialization phase of the system.

Parameters

schedule_name Name of the LIN schedule that is searched for. The name is limited to 63 characters.

node_name Name of the node that is searched for the schedule. The name is limited to 63 characters.

channel_name Name of the channel containing the node and the schedule. The name is limited to 63 characters.

schedule Returned pointer to the searched LIN schedule. Returns NULL if the function fails to find the schedule.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The searched schedule has been found.
`DSLIN_ERR_SCHEDULE_NOT_FOUND`	There is no schedule with this name installed on the selected node and channel.

Example

The example shows how to search for a LIN schedule.

```
enum DSLIN_ERROR error = DSLIN_OK;
dslin_schedule_p S1 = NULL;
error = dslin_schedule_lookup( "S1","node","channel", &S1 );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_schedule_create 161
dslin_schedule_entry_append 163
dslin_schedule_error_print 159
LIN Error Handling 26
LIN Schedule Handling 158
Standard Defines 15

dslin_schedule_create

Syntax

```
enum DSLIN_ERROR dslin_schedule_create(
   dslin_node_p master_node,
   const char* name,
   dslin_frame_p* schedule);
```

Include file

`dslin.h`

Purpose

To create a LIN frame.

Note

- Use the `dslin_schedule_create` function only during initialization of the system.
- A newly created LIN schedule cannot be interrupted. Use `dslin_schedule_breakable` to allow the schedule to be interrupted.
- Only a LIN master node provides LIN schedules.

Description

The `dslin_schedule_create` function allocates the memory for a new LIN schedule or returns the pointer to an existing LIN schedule if there is already a schedule with the name connected to the node. A newly created schedule is disabled by default. If an existing schedule was returned, the state (enabled or disabled) depends on the reused schedule.

Parameters

master_node Pointer to the LIN master node

name Name of the LIN schedule. The name is limited to 63 characters. If the string is NULL or empty (“ “) the LIN schedule cannot be found with `dslin_schedule_lookup`.

schedule Returned pointer to the created LIN schedule. Returns NULL if the function fails.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	A new schedule has been created successfully.
DSLIN_OBJECT_REUSED	A pointer to an existing schedule with the same name on the same node has been returned.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_ERR_MALLOC	Memory allocation error. There is not enough memory available on the master processor board.
DSLIN_ERR_SCHEDULE_COUNT	Illegal schedule number. The number of LIN schedules on a channel is limited to 32.

Example

The example shows how to create a LIN schedule.

```
enum DSLIN_ERROR error = DSLIN_OK;
dslin_schedule_p S1 = NULL;
// Create the handle for the LIN schedule.
error = dslin_schedule_create( lin_node, "S1", &S1 );
     dslin_schedule_error_print( S1, error );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_schedule_breakable..... 172
dslin_schedule_entry_append..... 163
dslin_schedule_error_print..... 159
dslin_schedule_lookup..... 160
LIN Error Handling..... 26

dslin_schedule_entry_append

Syntax

```
enum DSLIN_ERROR dslin_schedule_entry_append(
   dslin_schedule_p schedule,
   UInt8 frame_id,
   UInt8 frame_length,
   dsfloat frame_time);
```

Include file

`dslin.h`

Purpose

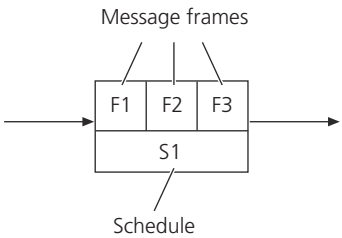
To append a frame to a schedule.

Note

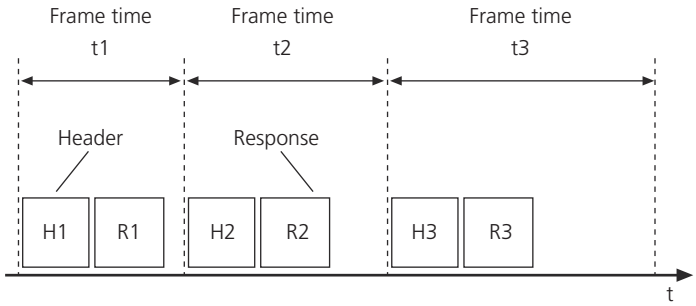
Use this function only during initialization time of the system.

Description

Defined frames can be added to existing schedules. This is the way to build up LIN schedules that define a sequence in which the frames are sent over the LIN bus. See the following illustration:



Each frame consists of a header and a response. Because LIN schedules are related to master nodes, the master sends the frame headers over the LIN bus and the master itself or a slave node sends the frame response. The time to send the header and to get the response including a delay time can be defined by the `frame_time` parameter. For detailed information on how to specify the frame time, refer to *LIN specification 1.2* and see the following illustration:



Parameters

schedule Pointer to a LIN schedule

frame_id Identifier of the frame that is added to the schedule.

frame_length Expected length of the response that is related to the frame header.

frame_time Specifies the time interval between two adjacent frames. The frame time consists of the time to send the header and to get the response (whole frame time).

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	A new schedule has been created successfully.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
DSLIN_ERR_SCHEDULE_FRAME_TIME_ILLEGAL	Illegal frame time specified. The valid range for frame times is specified by DSLIN_LIN_SCHEDULE_MIN_FRAME_TIME and DSLIN_LIN_SCHEDULE_MAX_FRAME_TIME.

Example

The following example shows how to add several frames to a schedule.

```
// Returned schedule position.
UInt8 pos = 0;
// Identifiers of the frames.
UInt8 frame_id1 = 0x01;
UInt8 frame_id2 = 0x02;
UInt8 frame_id3 = 0x03;
// Delay for the first frame (t1).
dsfloat frame_time1 = 0.015;
// Delay between the first and the second frame (t2).
dsfloat frame_time2 = 0.020;
```

```
// Delay between the second and the last scheduled frame (t3).
dsfloat frame_time3 = 0.030;
dslin_schedule_p S1 = NULL;
dslin_schedule_create( lin_node, "S1", &S1 );
dslin_schedule_entry_append( S1, frame_id1, 2, frame_time1 );
dslin_schedule_entry_append( S1, frame_id2, 4, frame_time2 );
dslin_schedule_entry_append( S1, frame_id3, 8, frame_time3 );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_schedule_create 161
dslin_schedule_error_print 159
dslin_schedule_lookup 160
LIN Error Handling 26
LIN Schedule Handling 158
Standard Defines 15

dslin_schedule_start

Syntax

```
enum DSLIN_ERROR dslin_schedule_start(
   dslin_schedule_p schedule,
   UInt8 repetitions);
```

Include file

`dslin.h`

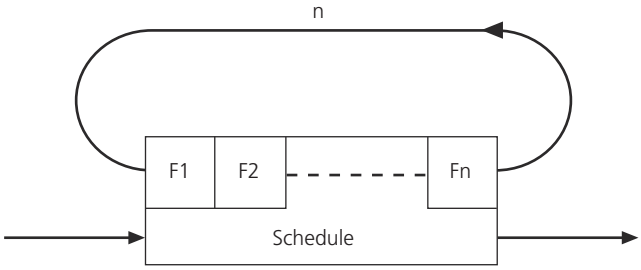
Purpose

To start a LIN schedule and repeat it *n* times.

Description

If a LIN schedule is already running, the `dslin_schedule_start` function terminates the execution of the running schedule. If the running schedule cannot be interrupted, the schedule is executed until the last header is sent. Then the new schedule is started.

The schedule is repeated as long as specified by the `repetitions` parameter and is executed in a loop. See the following illustration:



Parameters

schedule Pointer to a LIN schedule

repetitions Specifies the number of repetitions the schedule is executed for. Two modes are available:

Number of Repetitions	Meaning
0	The schedule is endlessly executed.
> 0	The schedule is executed as often as specified.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new schedule has been created successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_CHANNEL_IS_IN_SLEEP_MODE`	The error occurs when you try to start a schedule or a header and the LIN bus is in sleep mode. Send the wake-up command to the LIN bus. Refer to dslin_node_command_wakeup on page 115.
`DSLIN_ERR_CHANNEL_IS_DISABLED`	The error occurs when you try to start a schedule or a header and the LIN channel is disabled. Enable the LIN channel with `dslin_channel_enable` .

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_channel_enable..... 60
dslin_schedule_restart_at..... 171
dslin_schedule_stop..... 167
LIN Error Handling..... 26
Standard Defines..... 15

dslin_schedule_stop

Syntax

```
enum DSLIN_ERROR dslin_schedule_stop(
   dslin_schedule_p schedule);
```

Include file

`dslin.h`

Purpose

To stop the execution of a LIN schedule.

Description

The termination behavior depends on whether or not the schedule is interruptible.

- If the schedule is interruptible it is stopped after the currently executed schedule task (frame) has finished.
- If the schedule is not interruptible it cannot be terminated until the last schedule task is executed.

You can set the termination behavior of a schedule with the `dslin_schedule_breakable` and `dslin_schedule_unbreakable` functions.

Parameters

schedule Pointer to a LIN schedule

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	A new schedule has been created successfully.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_schedule_breakable 172
dslin_schedule_restart_at 171
dslin_schedule_start 165
dslin_schedule_unbreakable 174
LIN Error Handling 26
Standard Defines 15

dslin_schedule_resume_enable

Syntax

```
enum DSLIN_ERROR dslin_schedule_resume_enable(
   dslin_schedule_p schedule);
```

Include file

dslin.h

Purpose

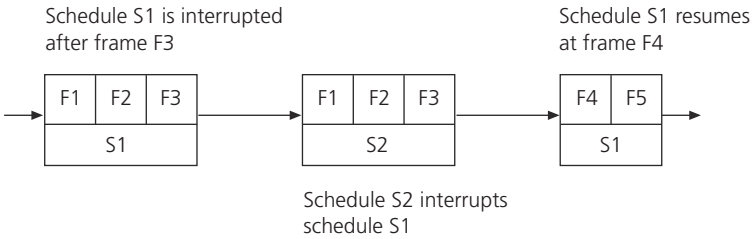
To enable a schedule to be resumed after a break.

Note

A schedule can only be resumed if it is interruptible.

Description

After the schedule is interrupted you can resume it at the position where it was interrupted. See the following illustration:



Parameters

schedule Pointer to a LIN schedule

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new schedule has been created successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_schedule_restart_at 171
dslin_schedule_resume_disable 170
dslin_schedule_start 165
dslin_schedule_stop 167
LIN Error Handling 26
Standard Defines 15

dslin_schedule_resume_disable

Syntax

```
enum DSLIN_ERROR dslin_schedule_resume_disable(
   dslin_schedule_p schedule);
```

Include file

`dslin.h`

Purpose

To disable the restart or resumption of a schedule if it is interruptible.

Description

After a schedule was interrupted by another schedule, you can restart or resume the interrupted schedule. If the schedule is not to be restarted or resumed, execute the `dslin_schedule_resume_disable` function. Schedules can be set to interruptible by the `dslin_schedule_breakable` function.

Parameters

schedule Pointer to a LIN schedule

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new schedule has been created successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_schedule_breakable 172
dslin_schedule_restart_at 171
dslin_schedule_resume_enable 168
dslin_schedule_start 165

dslin_schedule_restart_at

Syntax

```
enum DSLIN_ERROR dslin_schedule_restart_at(
   dslin_schedule_p schedule,
   UInt8 schedule_pos);
```

Include file

`dslin.h`

Purpose

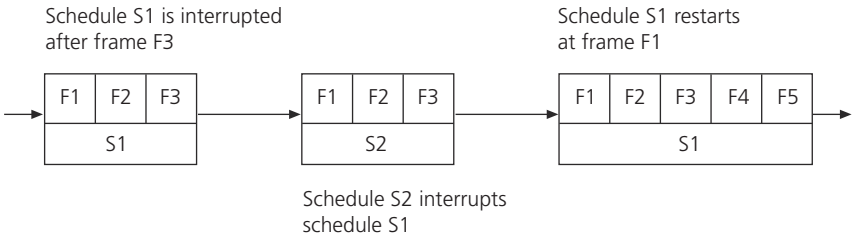
To restart an interrupted schedule at a defined position.

Note

- When the LIN schedule is already interrupted, you cannot start the schedule with the `dslin_schedule_restart` function. Use `dslin_schedule_start` instead.
- To define a schedule as interruptible, use `dslin_schedule_breakable`.

Description

If a LIN schedule is interrupted you can resume it (see `dslin_schedule_resume_enable` on page 168) or restart it at a defined position within it. The restart position can be defined by the `schedule_pos` parameter. After the interrupting schedule has finished, the schedule is restarted at the specified position. See the following illustration:



Schedules that are often interrupted and restarted from the beginning can never be finished.

Parameters

schedule Pointer to a LIN schedule

schedule_pos Lets you enter the position where the schedule must be restarted. The valid range is 1 … DSLIN_SCHEDULE_MAX_ENTRIES (64).

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new schedule has been created successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_SCHEDULE_POSITION_ILLEGAL`	An incorrect restart position was entered. The valid range is within 1 … `DSLIN_SCHEDULE_MAX_ENTRIES`.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_schedule_breakable 172
dslin_schedule_resume_disable 170
dslin_schedule_resume_enable 168
dslin_schedule_start 165
dslin_schedule_stop 167
LIN Error Handling 26
Standard Defines 15

dslin_schedule_breakable

Syntax

```
enum DSLIN_ERROR dslin_schedule_breakable(
   dslin_schedule_p schedule);
```

Include file

`dslin.h`

Purpose

To allow a schedule to be interrupted immediately.

Description

Schedules can be interrupted by other schedules that are started by the `dslin_schedule_start` function. Two modes are available to continue the interrupted schedule:

- To resume the schedule automatically, use the `dslin_schedule_resume_enable` function.
- To restart the schedule at a defined position, use the `dslin_schedule_restart_at` function.

If the schedule is stopped by the `dslin_schedule_stop` function it is not restarted or resumed automatically. Use the `dslin_schedule_start` function to restart the schedule again from the beginning.

Parameters

schedule Pointer to a LIN schedule

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new schedule has been created successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_schedule_restart_at..... 171
dslin_schedule_resume_disable..... 170
dslin_schedule_resume_enable..... 168
dslin_schedule_start..... 165

dslin_schedule_unbreakable

Syntax

```
enum DSLIN_ERROR dslin_schedule_unbreakable(
   dslin_schedule_p schedule);
```

Include file

`dslin.h`

Purpose

To protect a schedule against interruption by another schedule.

Description

After the `dslin_schedule_unbreakable` function is executed, the schedule cannot be interrupted by another schedule. It is executed until finished.

Parameters

schedule Pointer to a LIN schedule

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new schedule has been created successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_schedule_breakable 172
dslin_schedule_restart_at 171
dslin_schedule_resume_disable 170
dslin_schedule_resume_enable 168
dslin_schedule_start 165
LIN Error Handling 26
Standard Defines 15

dslin_schedule_status_get

Syntax

```
enum DSLIN_ERROR dslin_schedule_status_get(
   dslin_schedule_p schedule,
   dslin_schedule_status_get_t* status,
   UInt8* current_schedule_pos,
   dsfloat* timestamp);
```

Include file

`dslin.h`

Purpose

To get the current status of a LIN schedule.

Description

The status of the currently running schedule can be read by the `dslin_schedule_status_get` function. Different parameters are available:

- With the `dslin_schedule_status_get_t` structure you can get information on whether the schedule is active, pending, completed or aborted.
- The `current_schedule_pos` and the time stamp parameters show you the schedule task currently being executed and its time stamp.
- If the schedule is not running or pending, the position is `"0"`. A running schedule shows a position within 1 … DSLIN_SCHEDULE_MAX_ENTRIES. If a running resumable schedule is interrupted the last position is shown. If the schedule is not resumable, the position 0 is shown.

Parameters

schedule Pointer to a LIN schedule

status Pointer to the `dslin_schedule_status_get_t` structure that returns the status of the schedule. See Data Structures: dslin_schedule_status_t on page 25.

current_schedule_pos Address where the current schedule position is stored. The valid range for schedule positions is 1 … DSLIN_SCHEDULE_MAX_ENTRIES (see Predefined Symbols on page 15).

timestamp Address where the time stamp of the currently occurring event is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new schedule has been created successfully.
`DSLIN_NO_DATA_AVAILABLE`	The error code can be returned by the first read or get operation. It is a message and indicates no error.
`DSLIN_DATA_LOST`	Data was lost before it was read.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

LIN Error Handling.....26
Standard Defines.....15

dslin_schedule_board_get

Syntax

```
enum DSLIN_ERROR dslin_schedule_board_get(
   dslin_frame_p schedule,
   dslin_board_p* board);
```

Include file

`dslin.h`

Purpose

To get the pointer to the LIN board used.

Description

The `dslin_schedule_board_get` function allows you to get a pointer to the LIN board used. This is useful if you want to execute a board-specific function within a schedule function, for example, if you want to perform a board update with the `dslin_board_update` function.

Parameters

schedule Returned pointer to the created LIN schedule. Returns NULL if the function fails.

board Returned pointer to the LIN board.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new schedule has been created successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Schedules (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_board_update.....42
LIN Error Handling.....26
Standard Defines.....15

LIN Frame Handling

Introduction

The following functions and definitions are used to initialize and handle LIN frames.

Where to go from here

Information in this section

Information in other sections

Data Types and Enumerations..... 17
Provides definition of the data type and enumerations used by the LIN board functions.

Example of Initializing a Single Slave Node

Example

The example describes the simplest LIN slave node. It sends 8 bytes if a header with the identifier 0x01 is received. The expected baud rate is 9600 baud.

```
#include <brtenv.h>
#include <dslin.h>
#include <ds4330.h>
dslin_board_p board = NULL;
dslin_channel_p channel = NULL;
dslin_node_p node = NULL;
dslin_frame_p frame = NULL;
UInt8 data[8] = { 1,2,3,4,5,6,7,8 };
void main( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  init();
  // Board init
  error = dslin4330_board_init( DS4330_1_BASE, &board );
  dslin_board_error_print( board, error);
  // Channel init
  error = dslin_channel_create( board, "CH1", 1, &channel );
  dslin_channel_error_print( channel, error );
  error = dslin_channel_init( channel, 9600, 20, 20,
    DSLIN_TRANSCEIVER_ISO9141, DSLIN_TERMINATION_SLAVE_30K );
  dslin_channel_error_print( channel, error );
  error = dslin_channel_enable( channel );
  dslin_channel_error_print( channel, error );
  // Node init
  error = dslin_node_create( channel, "Node", &node );
  dslin_node_error_print( node, error );
  error = dslin_node_init( node, DSLIN_NODE_SLAVE, 64, 64 );
  dslin_node_error_print( node, error );
  // Frame init
  error = dslin_frame_create( node, "TX", &frame );
  dslin_frame_error_print( frame, error );
  error = dslin_frame_tx_init( frame, 0x01, 0.001 );
  dslin_frame_error_print( frame, error );
  dslin_frame_tx_data_set( frame, 8, data );
  dslin_frame_apply_settings( frame );
  error = dslin_channel_enable( channel );
  dslin_channel_error_print( channel, error );
```

```
  for(;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

LIN Error Handling 26
LIN Frame Handling 178
Standard Defines 15

Example of Initializing a LIN Frame to Receive and Transmit a Response

Preconditions

Transmit LIN frame You need the following information to transmit a LIN frame:

- The identifier of the header to react to
- The handle or name of the LIN node (slave)
- The name of the LIN frame (mostly from the LIN database file)
- The delay of the response in seconds

Receive LIN frame You need the following information to receive a LIN frame:

- The identifier of the header to react to
- The handle or name of the LIN node (only slave)
- The name of the LIN frame (mostly from the LIN database file)

Example

The following example shows you how to initialize a LIN frame:

```
#include <dslin.h>
void lin_node_frame_init( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  dslin_node_p lin_node = NULL;
  dslin_frame_p lin_frame_tx = NULL;
  dslin_frame_p lin_frame_rx = NULL;
// Get the handle to the node.
  error = dslin_node_parity_offset_set("LIN Node0","LIN Interface0", &lin_node );
  dslin_node_error_print( lin_node, error );
```

```
// Create the handle for the LIN tx frame.
  error = dslin_frame_create( lin_node, "LIN-FrameTX",
    &lin_frame_tx );
  dslin_node_error_print( lin_node_tx, error );
// Create the handle for the LIN rx frame.
  error = dsline_frame_create( lin_node, "LIN-FrameRX",
    &lin_frame_rx );
  dslin_node_error_print( lin_node_rx, error );
// Initialize the LIN tramsmit response
  error = dslin_frame_tx_init( lin_frame_tx, 0x12, 0.001 );
  dslin_frame_error_print( lin_frame_tx, error );
// Initialize the LIN receive response.
  error = dslin_frame_rx_init( lin_frame_rx, 0x12 );
  dslin_frame_error_print( lin_frame_rx, error );
}
```

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

LIN Error Handling 26
LIN Frame Handling 178
Standard Defines 15

Example of Reading Response Data

Example

To read the response data of a frame the following operations are necessary:

- Updating the data for the main processor with `dslin_board_update`.
- Getting the time stamp for the possibly received response with `dslin_frame_info_timestamp_get`.
- Checking if this is a new time stamp.
- Reading the response data.

The example demonstrates the reading of response data:

```
#include <dslin.h>
extern dslin_frame_p lin_frame_rx;
void lin_node_response_read( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  // We need the old timestamp to detect a new timestamp.
  static dsfloat ts_rx = 0.0;
  // Storage for one temp timestamp.
  dsfloat ts = 0.0;
  UInt8 len_rx = 0;
  UInt8 buffer_rx[8] = {0,0,0,0,0,0,0,0};
```

```
  // Update the data for the LIN software on the processor board.
  error = dslin_board_update( lin_board );
  // Test if we have received some new data.
  if( DSLIN_OK == error )
  {
    // We have received new data, but we don't know
    // which data was updated! So we must read all possible LIN data.
    // First we get the timestamp of the possible received response.
    error = dslin_frame_info_timestamp_get( lin_frame_rx, &ts );
    // Test if a valid timestamp is available.
    if( DSLIN_OK == error )
    {
      // We have a valid timestamp!
      // But is this a new timestamp?
      if( ts > ts_rx )
      {
        // OK, this is a new timestamp. Save the
        // timestamp for the next run.
        ts_rx = ts;
        // Now we are sure to read some new data.
        error = dslin_frame_info_data_get( lin_frame_rx,
               8,             // Limit to 8 data bytes.
               &len_rx,       // Returned byte count.
               &buffer_rx  ); // Returned data.
        msg_info_printf( 0, 0,
          "received response data[%d](%fs): 0x%x,0x%x,0x%x,0x%x,0x%x,0x%x,0x%x,0x%x",
          len_rx, ts_rx, buffer_rx[0],buffer_rx[1],buffer_rx[2],buffer_rx[3],
          buffer_rx[4],buffer_rx[5],buffer_rx[6],buffer_rx[7] );
      }
      else
      {
        // No new timestamp!
      }
    }
    else
    {
      //No timestamp available!
    }
  }
  else
  {
    // No data from dslin_board_update().
  }
}
```

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

Example of Updating the Outgoing Response Data

Example

The example shows how to update response data before transmitting the frame.

```
#include <dslin.h>
extern dslin_frame_p lin_frame_tx;
void lin_node_response_update( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  UInt8 data[8] = { 1,2,3,4,5,6,7,8 };
  // Update the response data.
  error = dslin_frame_tx_data_set( lin_frame_tx, 8, data );
  if( DSLIN_OK == error )
  {
     // Transfer the response data to the LIN board.
    error = dsline_frame_apply_settings( lin_frame_tx );
    if( DSLIN_OK == error )
    {
      // Successfully updated!
    }
    else
    {
      // We have a serious problem!
      // Possible an communication overload
      // to the LIN board.
    }
  }
  else
  {
  // We have a serious problem!
  // Possible an uninitialized LIN frame.
  }
}
```

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_frame_apply_settings 212
dslin_frame_tx_data_set 205
LIN Error Handling 26
LIN Frame Handling 178
Standard Defines 15

dslin_frame_lookup

Syntax

```
enum DSLIN_ERROR dslin_frame_lookup(
   const char* frame_name,
   const char* node_name,
   const char* channel_name,
   dslin_frame_p* frame);
```

Include file

`dslin.h`

Purpose

To search for an existing LIN frame.

Note

- The search is limited by the selected LIN node and LIN channel.
- Use the `dslin_frame_lookup` function only during the initialization phase of the system.

Purpose

frame_name Name of the LIN frame that is searched for. The name is limited to 63 characters.

node_name Name of the node that is searched for the frame. The name is limited to 63 characters.

channel_name Name of the channel containing the node and the frame. The name is limited to 63 characters.

frame Returned pointer to the searched LIN frame. Returns NULL if the function fails to find the frame.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The searched frame has been found.
DSLIN_ERR_FRAME_NOT_FOUND	There is no frame with this name installed on the selected node and channel.

Example

The example shows how to search for a LIN frame. For a detailed example of LIN frame handling, refer to Example of Initializing a LIN Frame to Receive and Transmit a Response on page 181.

```
enum DSLIN_ERROR error = DSLIN_OK;
dslin_frame_p lin_frame_tx = NULL;
error = dslin_frame_lookup( "LIN FrameTX", "LIN Node0", "LIN
Interface0", &lin_frame_tx );
```

```
dslin_frame_error_print( lin_frame, error );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_create..... 189
dslin_frame_error_print..... 196
dslin_frame_rx_init..... 190
dslin_frame_tx_init..... 192
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_rx_msgid_lookup

Syntax

```
enum DSLIN_ERROR dslin_frame_rx_msgid_lookup(
   dslin_node_p node,
   UInt16 msgid,
   dslin_frame_p* rx_frame);
```

Include file

`dslin.h`

Purpose

To search for a LIN frame specified by the message ID.

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Purpose

node Pointer to a LIN node

msgid Message identifier of the frame that is searched for.

rx_frame Returned pointer to the LIN frame. The value is NULL if the function cannot find the frame.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The frame was found.
DSLIN_ERR_FRAME_NOT_FOUND	There is no frame with the message identifier on the selected node.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_create..... 189
dslin_frame_rx_init..... 190
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_tx_msgid_lookup

Syntax

```
enum DSLIN_ERROR dslin_frame_tx_msgid_lookup(
   dslin_node_p node,
   UInt16 msgid,
   dslin_frame_p* tx_frame);
```

Include file

dslin.h

Purpose

To search for a transmitting LIN frame specified by the message ID.

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Parameter

node Pointer to a LIN node

msgid Message identifier of the frame that is searched for.

tx_frame Returned pointer to the LIN frame. The value is NULL if the function cannot find the frame.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The frame was found.
`DSLIN_ERR_FRAME_NOT_FOUND`	There is no frame with the message identifier on the selected node.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_create..... 189
dslin_frame_tx_init..... 192
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_create

Syntax

```
enum DSLIN_ERROR dslin_frame_create(
   dslin_node_p node,
   const char* name,
   dslin_frame_p* frame);
```

Include file

`dslin.h`

Purpose

To create a LIN frame.

> **Note**
>
> Use the `dslin_frame_create` function only during initialization of the system.

Description

The `dslin_frame_create` function allocates the memory for a new LIN frame or returns the pointer to an existing LIN frame if there is already a frame with the name connected to the node. A newly created frame is disabled by default. If an existing frame was returned, the state (enabled or disabled) depends on the reused frame.

Parameters

node Pointer to a LIN node

name Name of the LIN frame. The name is limited to 63 characters. If the string is NULL or empty (“”), the LIN frame cannot be found with `dslin_frame_lookup`.

frame Returned pointer to the created LIN frame. Returns NULL if the function fails.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new frame has been created successfully.
`DSLIN_OBJECT_REUSED`	A pointer to an existing frame with the same name on the same node has been returned.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_ERR_MALLOC`	Memory allocation error. There is not enough memory available on the master processor board.

Error Code	Meaning
DSLIN_ERR_FRAME_COUNT	Illegal frame number. The number of LIN frames on a channel is limited to 64.
DSLIN_ERR_NAME_TOO_LONG	The name of the frame is too long. Valid names are up to 63 characters long.

Execution times

For information, refer to Function Execution Times on page 249.

Example

The example shows how to create a LIN frame. For a detailed example of LIN frame handling, refer to Example of Initializing a LIN Frame to Receive and Transmit a Response on page 181.

```
enum DSLIN_ERROR error = DSLIN_OK;
dslin_frame_p lin_frame_tx = NULL;
// Create the handle for the LIN frame.
error = dslin_frame_create( lin_node, "LIN-FrameTX", &lin_frame_tx );
dslin_frame_error_print( lin_frame_tx, error );
```

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_frame_error_print 196
dslin_frame_lookup 185
dslin_frame_rx_init 190
dslin_frame_tx_init 192
LIN Error Handling 26
LIN Frame Handling 178
Standard Defines 15

dslin_frame_rx_init

Syntax

```
enum DSLIN_ERROR dslin_frame_rx_init(
   dslin_frame_p frame,
   UInt8 identifier,
   UInt8 length);
```

Include file

dslin.h

Purpose

To prepare a LIN frame to receive data.

> **Note**
>
> Use the `dslin_frame_rx_init` function only during the initialization phase of the system.

Description

The function configures a frame to receive data. By default, the frame is enabled after initialization. It is not necessary to run the `dslin_frame_enable` function.

The identifier labels the frame. If the identifier of the frame is specified later during the node configuration, you can prepare the frame without identifier. In this case, set the `identifier` parameter to 0xFF.

Parameters

frame Pointer to a LIN frame

identifier Identifier of the LIN frame. To specify the identifier later, set the value to 0xFF.

length Expected number of receive bytes, without the checksum byte. Use a range within 0 … DSLIN_MAX_FRAME_LENGTH (255 bytes).

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The frame has been initialized successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Example

The example shows how to prepare a LIN frame to receive data. For a detailed example of LIN frame handling, refer to Example of Initializing a LIN Frame to Receive and Transmit a Response on page 181.

```
error = dslin_frame_rx_init( lin_frame_rx, 0x12, 4);
dslin_frame_error_print( lin_frame_rx, error );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_create..... 189
dslin_frame_error_print..... 196
dslin_frame_lookup..... 185
dslin_frame_tx_init..... 192
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_tx_init

Syntax

```
enum DSLIN_ERROR dslin_frame_tx_init(
   dslin_frame_p frame,
   UInt8 identifier,
   dsfloat response_delay);
```

Include file

`dslin.h`

Purpose

To prepare a frame to send response data.

Note

Use the `dslin_frame_tx_init` function only during initialization of the system.

Description

The function configures a frame to send data. By default, the frame is enabled after initialization. It is not necessary to run the `dslin_frame_enable` function.

The identifier labels the frame. If the identifier of the frame is specified later during the node configuration, you can prepare the frame without identifier. In this case, set the `identifier` parameter to 0xFF.

Parameters

frame Pointer to a LIN frame

identifier Identifier of the transmit frame. To specify the identifier later, set the value to 0xFF.

response_delay Delay of the response after a header has been received within the range 0 … DSLIN_MAX_FRAME_RESPONSE_DELAY (10 seconds).

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The frame has been initialized successfully.
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
DSLIN_ERR_RESPONSE_DELAY_ILLEGAL	Valid values for the response delay are within the range 0 … 10 seconds.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Example

The example shows how to prepare a LIN frame to transmit data.

```
// Delay the response about 1ms after receiving the header
// with the identifier 0x12.
error = dslin_frame_tx_init( lin_frame_tx, 0x12, 0.001 );
dslin_frame_error_print( lin_frame_tx, error );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_frame_create 189
dslin_frame_error_print 196
dslin_frame_lookup 185
dslin_frame_rx_init 190
LIN Error Handling 26
LIN Frame Handling 178
Standard Defines 15

dslin_frame_rx_eventtrig_init

Syntax

```
enum DSLIN_ERROR dslin_frame_rx_eventtrig_init(
   dslin_frame_p frame,
   dslin_frame_p unconditional_frame,
   dslin_frame_p* frame_list,
   UInt16 list_size);
```

Include file

`dslin.h`

Purpose

To configure a frame to fetch the newest data or status from a list of event-triggered frames and one unconditional frame.

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Note

You can use the `dslin_frame_rx_eventtrig_init` function only for frames configured with the `dslin_frame_rx_init` function.

Description

During run time the data or status of a frame configured with this function are read just like a frame configured with the `dslin_frame_rx_init` function. The frame ID of the unconditional frame is used to filter out the unwanted event-triggered frame responses by examining the first byte of the response.

You can use the following functions in conjunction with the `dslin_frame_rx_eventtrig_init` function:

- `dslin_frame_lookup` on page 185
- `dslin_frame_lock` on page 199 to update all frame information for the get functions and lock the frame against update.
- `dslin_frame_enable` on page 197 to enable updating with new data or status.
- `dslin_frame_disable` on page 198 to disable updating with new data or status.
- `dslin_frame_info_checksum_get` on page 218 to get the checksum of the fetched response.
- `dslin_frame_info_data_get` on page 214 to get the data of the fetched response.
- `dslin_frame_info_id_get` on page 217 to get the frame ID of the fetched response.

- `dslin_frame_info_status_get` on page 219 to get the status of the frame.
- `dslin_frame_info_timestamp_get` on page 215 to get the time stamp of the data or status.
- `dslin_frame_unlock` on page 201 to enable updating of the frame.

Parameters

frame Pointer to a LIN frame. This frame fetches the response data from the event-triggered frames and the unconditional frame.

unconditional_frame Pointer to the unconditional frame. This frame defines the frame ID.

frame_list Array of pointers to the event-triggered frames

list_size Number of frame pointers in the list

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The frame is configured successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_ERR_MALLOC`	Memory allocation error on the processor board. There is not enough memory available on the processor board.

Example

The example shows how to configure the frame.

```
dslin_frame_p EventTriggeredFrameList[16];
dslin_frame_p EventTriggeredFrame;
dslin_frame_p UnconditionalFrame;

dslin_frame_rx_eventtrig_init(EventTriggeredFrame,
                              UnconditionalFrame,
                              EventTriggeredFrameList,
                              16);
```

Related topics

Basics

Handling Frames (DS4330 Features 🕮)

Examples

References

dslin_frame_error_print

Syntax

```
define dslin_frame_error_print(
   dslin_frame_p frame,
   enum DSLIN_ERROR error);
```

Include file

`dslin.h`

Purpose

To write errors to the dSPACE log file. If `error==DSLIN_OK` nothing is written to the output.

Note

Reporting the error information to the log file is a time-consuming process. Consider this when using the function within your task.

The dSPACE log file can be opened in the dSPACE experiment software.

Parameters

frame Pointer to a LIN frame

error Error code to be written to the log file as plain text.

Return value

None

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

LIN Error Handling.....26
Standard Defines.....15

dslin_frame_enable

Syntax

```
enum DSLIN_ERROR dslin_frame_enable(
   dslin_frame_p frame);
```

Include file

`dslin.h`

Purpose

To enable a LIN frame.

Note

Only a previously disabled frame can be enabled.

Parameters

frame Pointer to a LIN frame

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The frame is enabled.
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
DSLIN_ERR_FRAME_NOT_INITIALIZED	The frame is not initialized.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_disable..... 198
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_disable

Syntax

```
enum DSLIN_ERROR dslin_frame_disable(
   dslin_frame_p frame);
```

Include file

`dslin.h`

Purpose

To disable a LIN frame. The frame does not send or receive data any more.

Note

A TX frame also does not response to a received master header.

Parameters

frame	Pointer to a LIN frame

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The frame is disabled.
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
DSLIN_ERR_FRAME_NOT_INITIALIZED	The frame is not initialized.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

References

dslin_frame_enable 197
LIN Error Handling 26
LIN Frame Handling 178
Standard Defines 15

dslin_frame_lock

Syntax

```
enum DSLIN_ERROR dslin_frame_lock(
   dslin_frame_p frame);
```

Include file

`dslin.h`

Purpose

To lock a frame.

Description

All data that you can read with a dslin_frame_xxxx_get function from a frame is locked. It cannot be updated. The lock ensures consistency for the different data bytes of the frame. If you do not lock the frame, data may be overwritten by the `dslin_board_update` function and old data may be mixed with new data.

You can use the `dslin_frame_lock` function during evaluation of the LIN frame-specific data.

Note

After you have locked and evaluated the data, you have to execute the `dslin_frame_unlock` function to get new data.

Parameters

frame Pointer to a LIN frame

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The frame is locked.
DSLIN_ERR_FRAME_NOT_INITIALIZED	The frame is initialized.
DSLIN_ERR_NO_DATA_AVAILABLE	There is no data available with this frame.
DSLIN_ERR_DATA_LOST	The frame is currently locked and cannot accept new data. New data is rejected.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Example

The example shows how to prepare to lock data on a LIN frame. For a detailed example of LIN frame handling, refer to Example of Reading Response Data on page 182.

```
// Lock the frame data against updates.
dslin_frame_lock( frame );
// Now we can read consistent data.
dslin_frame_info_timestamp_get( frame, &ts );
dslin_frame_info_data_get( frame, sizeof(buffer), &len, buffer );
// Unlock the frame.
dslin_frame_unlock( frame );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_info_data_get..... 214
dslin_frame_info_timestamp_get..... 215
dslin_frame_unlock..... 201
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_unlock

Syntax

```
enum DSLIN_ERROR dslin_frame_unlock(
   dslin_frame_p frame);
```

Include file

`dslin.h`

Purpose

To unlock a LIN frame and allow updates of the data.

Description

All data of a frame that you can get with `dslin_frame_xxxx_get` functions is overwritten by new data. After a lock you have to execute the `dslin_frame_unlock` function to get new data.

Parameters

frame Pointer to a LIN frame

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The frame is unlocked.
`DSLIN_ERR_FRAME_NOT_INITIALIZED`	The frame is not initialized.

Error Code	Meaning
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_lock..... 199
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_id_set

Syntax

```
enum DSLIN_ERROR dslin_frame_id_set(
   dslin_frame_p frame,
   UInt8 identifier);
```

Include file

dslin.h

Purpose

To set the identifier of a frame.

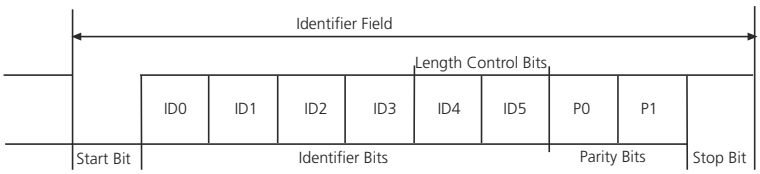
Note

- The `dslin_frame_id_set` function configures the frame and the slave processor. To avoid data loss, execute this function only during the initialization phase or if you set up a new configuration.
- The settings specified by the `dslin_frame_id_set` function have to be applied with the `dslin_frame_apply_settings` function.

Description

You can set the identifier of the frame explicitly if you want to change the previous settings made with `dslin_frame_rx_init` or `dslin_frame_tx_init`.

See the following illustration for information on the identifier field.



For detailed information on the identifier field, refer to the LIN specification.

Parameters

frame Pointer to a LIN frame

identifier Identifier of the frame.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The ID is successfully set.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_ERR_FRAME_NOT_INITIALIZED`	The frame is not initialized.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_apply_settings..... 212
dslin_frame_info_id_get..... 217
dslin_frame_rx_init..... 190
dslin_frame_tx_init..... 192
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_msgid_set

Syntax

```
enum DSLIN_ERROR dslin_frame_msgid_set(
   dslin_frame_p frame,
   UInt16 msgid);
```

Include file

`dslin.h`

Purpose

To set the message ID for a frame.

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Description

Each frame has a unique 16-bit message ID. During node configuration, the message ID is associated with a protected identifier, which is used in normal communication with the node.

Note

A sending response data frame and the receiving response data frame on the same node must have the same message identifier.

Parameters

frame Pointer to a LIN frame

msgid Message identifier of the frame

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The message ID is set successfully.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_ERR_FRAME_NOT_INITIALIZED	The frame is not initialized.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_apply_settings..... 212
dslin_frame_msgid_get..... 221
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_tx_data_set

Syntax

```
enum DSLIN_ERROR dslin_frame_tx_data_set(
   dslin_frame_p frame,
   UInt8 length,
   UInt8* data);
```

Include file

dslin.h

Purpose

To update the response data of the frame.

Note

The settings specified by the `dslin_frame_tx_data_set` function have to be applied by the `dslin_frame_apply_settings` function.

Description

The maximum length allowed is within the range 0 … DSLIN_MAX_FRAME_LENGTH (255 bytes).

Parameters

frame Pointer to a LIN frame

length Byte count of the response without checksum. The allowed length is within the range 0 … DSLIN_MAX_FRAME_LENGTH (255 bytes).

data Address where the source data is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The response data is successfully updated.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_ERR_FRAME_NOT_INITIALIZED`	The frame is not initialized.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_apply_settings..... 212
dslin_frame_info_data_get..... 214
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_tx_response_delay_set

Syntax

```
enum DSLIN_ERROR dslin_frame_tx_response_delay_set(
   dslin_frame_p frame,
   dsfloat delay);
```

Include file

`dslin.h`

Purpose

To set the response delay of the frame.

> **Note**
>
> The settings specified by the `dslin_frame_tx_response_delay_set` function have to be applied with the `dslin_frame_apply_settings` function.

Description

The response delay is the time between the slave detecting a header and the sending of the response. The response delay is related to the "in-frame response space" which separates the header and the response of a message.

Parameters

frame Pointer to a LIN frame

delay Delay of the response after a received header within the range 0 … DSLIN_MAX_FRAME_RESPONSE_DELAY (10 seconds).

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The delay time of the response has been set successfully.
`DSLIN_ERR_FRAME_NOT_INITIALIZED`	The frame is not initialized.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_ERR_RESPONSE_DELAY_ILLEGAL`	Valid values for the response delay are within the range 0 … 10 seconds.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_apply_settings..... 212
dslin_frame_info_timestamp_get..... 215
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_length_set

Syntax

```
enum DSLIN_ERROR dslin_frame_length_set(
   dslin_frame_p frame,
   UInt8 length);
```

Include file

`dslin.h`

Purpose

To set the response byte count (length) of the LIN frame.

Note

The settings specified by the `dslin_frame_length_set` function have to be applied with the `dslin_frame_apply_settings` function.

Description

For the total byte count of tx_frames and rx_frames, you must add one byte for the checksum field.

Parameters

frame Pointer to a LIN frame

length Byte count of the response without checksum. The allowed length is within the range 0 … DSLIN_MAX_FRAME_LENGTH (255 bytes).

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The delay time of the response has been set successfully.
DSLIN_ERR_FRAME_NOT_INITIALIZED	The frame is not initialized.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_apply_settings..... 212
dslin_frame_info_data_get..... 214
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_tx_checksum_set

Syntax

```
enum DSLIN_ERROR dslin_frame_tx_checksum_set(
   dslin_frame_p frame,
   UInt8 checksum);
```

Include file

dslin.h

Purpose

To set the checksum for a tx_frame.

Note

- The settings specified by the `dslin_frame_tx_checksum_set` function have to be applied with the `dslin_frame_apply_settings` function.
- The external checksum is transmitted only once. The next time the frame is transmitted, the automatically generated checksum is sent again.

Description

You can replace the valid checksum before transmitting the frame. This is useful to simulate a wrong checksum, for example.

Parameters

frame Pointer to a LIN frame

checksum Checksum that replaces the valid checksum.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The checksum has been set successfully.
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_apply_settings..... 212
dslin_frame_info_checksum_get..... 218
LIN Error Handling..... 26

dslin_frame_mode_set

Syntax

```
enum DSLIN_ERROR dslin_frame_mode_set(
   dslin_frame_p frame,
   enum DSLIN_FRAME_MODE mode);
```

Include file

`dslin.h`

Purpose

To set the mode for the frame.

> **Note**
>
> The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Parameters

frame Pointer to a LIN frame

mode Specifies the mode of the LIN frame. For detailed information on modes, refer to Data Types and Enumerations on page 17. Valid values are:

Symbol	Meaning
DSLIN_FRAME_MODE_CLASSIC_CHECKSUM	The default checksum mode for LIN frames.
DSLIN_FRAME_MODE_ENHANCED_CHECKSUM	The checksum mode for LIN 2.0 frames.
DSLIN_FRAME_MODE_EVENT_TRIGGRED	Event-triggered frame Note: When this mode is set for an identifier, all TX frames with that identifier installed on the same LIN channel use this mode.
DSLIN_FRAME_MODE_UNCONDITIONAL	The default for a TX or RX frame.
DSLIN_FRAME_MODE_TX_ONCE	By default a TX frame is transferred every time the LIN master requests the frame response.
DSLIN_FRAME_MODE_TX_ALWAYS	The frame response data is always sent when a matching LIN master node request is received. This is the default for a TX frame.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The mode is set successfully.
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type
DSLIN_ERR_FRAME_NOT_INITIALIZED	The frame is not initialized.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_apply_settings..... 212
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_apply_settings

Syntax

```
enum DSLIN_ERROR dslin_frame_apply_settings(
   dslin_frame_p frame);
```

Include file

dslin.h

Purpose

To apply the settings made with the frame set functions. The function transfers the data to the slave processor.

Note

After you have specified the settings with the appropriate dslin_xxxx_set functions, you can call the `dslin_frame_apply_settings` function once to transfer all the settings to the slave processor of the LIN board. Consider that only the settings of the corresponding frame are transferred. For each frame, you have to call the `dslin_frame_apply_settings` function separately.

Parameters

frame Pointer to a LIN frame

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	Data has been transferred successfully to the slave processor.
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_info_data_get

Syntax

```
enum DSLIN_ERROR dslin_frame_info_data_get(
   dslin_frame_p frame,
   UInt8 maxlen,
   UInt8* len,
   UInt8* buffer);
```

Include file

`dslin.h`

Purpose

To copy the frame data to the target data buffer.

> **Tip**
>
> You can use time stamps to define whether or not the data is new.

Description

The available data is always copied to the target buffer. It makes no difference whether the data is new or not, but you can check this by using the `dslin_frame_info_timestamp_get` function. The target buffer with the appropriate length has to be defined by the user. If you want to get 8 bytes, you have to define a buffer with a length of "8". For example,

```
UInt8 buffer[8] = {0,0,0,0,0,0,0,0};
```

Parameters

frame Pointer to a LIN frame

maxlen Maximum number of bytes to be copied.

len Address where the number of received bytes is stored.

buffer Address where the target buffer is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	Data has been successfully copied to the buffer.
`DSLIN_ERR_FRAME_NOT_INITIALIZED`	The frame is not initialized.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Example

The example shows how to copy frame data to a buffer. For a detailed example of LIN frame handling, refer to Example of Reading Response Data on page 182.

```
UInt8 len = 0;
UInt8 buffer[8] = { 0,0,0,0,0,0,0,0 };
enum DSLIN_ERROR error = DSLIN_OK;
error = dslin_frame_info_data_get( frame, sizeof(buffer), &len, buffer );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_info_timestamp_get..... 215
dslin_frame_length_set..... 208
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_info_timestamp_get

Syntax

```
enum DSLIN_ERROR dslin_frame_info_timestamp_get(
   dslin_frame_p frame,
   dsfloat* ts);
```

Include file

`dslin.h`

Purpose

To get the time stamp of the last frame event that occurred.

Description

The time stamp also depends on whether or not an error occurs:

Frame	Time Stamp
Frame without error	The time is sampled when the message frame is completed by the response.
Frame with error	The time is sampled when the error is detected.

Parameters

frame Pointer to a LIN frame

ts Address where the returned time stamp is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	Data was successfully copied to the buffer.
`DSLIN_ERR_FRAME_NOT_INITIALIZED`	The frame is not initialized.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_tx_response_delay_set..... 207
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_info_id_get

Syntax

```
enum DSLIN_ERROR dslin_frame_info_id_get(
   dslin_frame_p frame,
   UInt8* id);
```

Include file

`dslin.h`

Purpose

To get the identifier of the current message frame.

Description

Returns the identifier set during the initialization with `dslin_frame_rx_init`, `dslin_frame_tx_init` or set after initialization with `dslin_frame_id_set`.

Parameters

frame Pointer to a LIN frame

id Address where the returned identifier is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The identifier has been read successfully.
`DSLIN_ERR_FRAME_NOT_INITIALIZED`	The frame is not initialized.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_id_set..... 202
dslin_frame_rx_init..... 190
dslin_frame_tx_init..... 192
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_info_checksum_get

Syntax

```
enum DSLIN_ERROR dslin_frame_info_checksum_get(
   dslin_frame_p frame,
   UInt8* checksum);
```

Include file

`dslin.h`

Purpose

To get the checksum of the current message frame.

Description

In the case of a receive frame, this is the checksum received. For a transmit frame, it is the checksum sent.

Parameters

frame Pointer to a LIN frame

checksum Address where the returned checksum is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The checksum has been read successfully.
`DSLIN_ERR_FRAME_NOT_INITIALIZED`	The frame was not initialized.

Error Code	Meaning
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_tx_checksum_set..... 209
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_info_status_get

Syntax

```
enum DSLIN_ERROR dslin_frame_info_status_get(
   dslin_frame_p frame,
   dslin_frame_status_t* status);
```

Include file

`dslin.h`

Purpose

To get the frame status.

Description

You can use the `dslin_frame_info_status_get` function to read the current status of a LIN frame, which is provided by the `Data Structures: dslin_frame_status_t` structure. Evaluating the members of the structure provides more information on the current frame status.

Parameters

frame Pointer to a LIN frame

status Address where the returned status is stored.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The status has been read successfully.
DSLIN_ERR_FRAME_NOT_INITIALIZED	The frame is not initialized.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Example

The example shows how to get the status of a frame. For a detailed example of LIN frame handling, refer to Example of Initializing a LIN Frame to Receive and Transmit a Response on page 181.

```
// This example shows the evaluation of the error
// for a tx frame. This is mainly only the bit error.
dslin_frame_status_t status;
// Get the status and error information.
error = dslin_frame_info_status_get( tx_frame, &status );
// If data was available, evaluate the data.
  if( DSLIN_OK == error )
  {
    // At least one error is occurred.
    if( status.error )
    {
     // Test for a bit error
     if( status.error_bit )
     {
        // Error Handling
     }
    }
  }
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

Data Structures: dslin_frame_status_t..... 23
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_msgid_get

Syntax

```
enum DSLIN_ERROR dslin_frame_msgid_get(
   dslin_frame_p frame,
   UInt16* msgid);
```

Include file

`dslin.h`

Purpose

To get the message ID used for a frame.

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Description

Each frame has a unique 16-bit message ID. During node configuration, the message ID is associated with a protected identifier, which is used in normal communication with the node.

Parameters

frame Pointer to a LIN frame

msgid Returned 16-bit message identifier of the frame.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The function has returned the message ID successfully.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_ERR_WRONG_TYPE`	Wrong input pointer type
`DSLIN_ERR_FRAME_NOT_INITIALIZED`	The frame is not initialized.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_msgid_set..... 204
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_frame_board_get

Syntax

```
enum DSLIN_ERROR dslin_frame_board_get(
   dslin_frame_p frame,
   dslin_board_p* board);
```

Include file

`dslin.h`

Purpose

To get the pointer to the LIN board used.

Description

The `dslin_frame_board_get` function allows you to get a pointer to the LIN board used. This is useful if you want to execute a board-specific function within a frame function, for example, if you want to perform a board update with the `dslin_board_update` function.

Parameters

frame Pointer to a LIN frame

board Returned pointer to the LIN board.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The pointers to the frame and the LIN board are returned successfully.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_ERR_WRONG_TYPE	Wrong input pointer type

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_board_update..... 42
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_checksum_calc_enhanced

Syntax

```
enum DSLIN_ERROR dslin_checksum_calc_enhanced(
   enum DSLIN_FRAME_MODE checksum,
   UInt8 frame_id,
   UInt8* data,
   UInt8 dlc,
   UInt8* checksum);
```

Include file

dslin.h

Purpose

To calculate the checksum with the specified data and frame identifier.

Note

The function is available only for LIN protocol 2.0 and the DS4330 with firmware version 1.3 and higher.

Description

The function returns the checksum for the input data. The calculated checksum depends on the checksum mode passed by the `checksum` parameter.

Parameters

checksum Specifies the checksum. The valid values are:

Symbol	Meaning
`DSLIN_FRAME_MODE_CLASSIC_CHECKSUM`	The default checksum mode for LIN frames.
`DSLIN_FRAME_MODE_ENHANCED_CHECKSUM`	The checksum mode for LIN 2.0 frames.

frame_id Frame identifier

data Pointer to the input data field

dlc Length of the data array

checksum Pointer to the returned checksum

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The checksum is calculated.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_info_checksum_get..... 218
dslin_frame_tx_checksum_set..... 209
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

dslin_checksum_calc

Syntax

```
enum DSLIN_ERROR dslin_checksum_calc(
   UInt8* data,
   UInt8 datalength,
   UInt8* checksum);
```

Include file

`dslin.h`

Purpose

To calculate the checksum.

Description

The function returns the checksum for the input data.

Parameters

data Pointer to the input data field

datalength Length of the data array.

checksum Pointer to the returned checksum

Return value

None

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Handling Frames (DS4330 Features 🕮)
LIN Bus Handling (DS4330 Features 🕮)

Examples

Example of Initializing a LIN Frame to Receive and Transmit a Response..... 181

References

dslin_frame_info_checksum_get..... 218
dslin_frame_tx_checksum_set..... 209
LIN Error Handling..... 26
LIN Frame Handling..... 178
Standard Defines..... 15

LIN Interrupt Handling

Introduction

The following functions are used to implement interrupts in LIN applications.

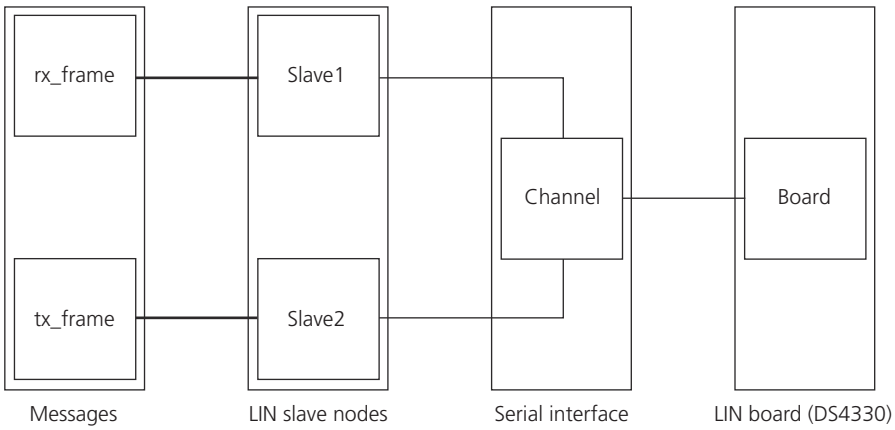
Where to go from here

Information in this section

Example of Using LIN Frame Interrupts

Example

The example shows how to implement a LIN bus with two LIN slaves and use a frame interrupt. The following illustration shows you a scheme of the specified LIN bus:




```
#include <brtenv.h>
#include <dslin.h>
#include <ds4330.h>
dslin_board_p board = NULL;
dslin_channel_p channel;
dslin_node_p slave1;
dslin_node_p slave2;
dslin_frame_p rx_frame;
dslin_frame_p tx_frame;
dsfloat period = 1.0;
void lin_interrupt_handler( void );
void lin_board_init( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  // Initialize the RTLIB.
  init();
  // Initialize the IO board.
  error = dsline4330_board_init( DS4330_1_BASE, &board );
  dslin_board_error_print( board, error);
}
// Initializes the LIN channel.
void lin_channel_init( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  error = dslin_channel_create( board, "LIN channel", 1, &channel );
  dslin_channel_error_print( channel, error );
  error = dslin_channel_init( channel, 20000, 13, 1, DSLIN_TRANSCEIVER_ISO9141, DSLIN_TERMINATION_SLAVE_30K );
  dslin_channel_error_print( channel, error );
}
// Initializes the LIN node.
void lin_node_init( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  // Connect the first slave node to the LIN channel.
  error = dslin_node_create( channel, "slaven node 1", &slave1 );
```

```
  dslin_node_error_print( slave1, error );
  error = dslin_node_init( slave1, DSLIN_NODE_SLAVE, 64, 64 );
  dslin_node_error_print( slave1, error );
  // Connect the second slave node to the LIN channel.
  error = dslin_node_create( channel, "slaven node 2", &slave2 );
  dslin_node_error_print( slave2, error );
  error = dslin_node_init( slave2, DSLIN_NODE_SLAVE, 64, 64 );
  dslin_node_error_print( slave2, error );
}
// Initializes the LIN frames.
void lin_frame_init( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  UInt8 frame_data[8] = {1,2,3,4,5,6,7,8};
  // Inizialize one rx frame on the first slave node.
  error = dslin_frame_create( slave1, "rx_frame", &rx_frame );
  dslin_frame_error_print( rx_frame, error );
  error = dslin_frame_rx_init( rx_frame, 0x01, 8 );
  dslin_frame_error_print( rx_frame, error );
  // Inizialize one tx frame on the second slave node.
  error = dslin_frame_create( slave2, "tx_frame", &tx_frame );
  dslin_frame_error_print( tx_frame, error );
  error = dsline_frame_tx_init( tx_frame, 0x01, 0.001 );
  dslin_frame_error_print( tx_frame, error );
  error = dslin_frame_tx_data_set(tx_frame, 8, frame_data );
  dslin_frame_error_print( tx_frame, error );
  error = dsline_frame_apply_settings( tx_frame );
  dslin_frame_error_print( tx_frame, error );
}
// Initialize frame interrupts.
void lin_interrupt_init( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  // Install the interrupt vector.
  install_phs_int_vector( DS4330_1_BASE, 0, lin_interrupt_handler );
  // Clear the DS4330 interrupt.
  ds4330_dpmem_interrupt_clear( DS4330_1_BASE );
  error = dslin_node_frame_interrupt_init( slave1, 0, DSLIN_FRAME_INT_RESPONSE_RECEIVED, 0x1 );
  dslin_node_error_print( slave1, error );
  error = dslin_node_frame_interrupt_init( slave2, 1, DSLIN_FRAME_INT_RESPONSE_SEND, 0x1 );
  dslin_node_error_print( slave1, error );
}
void lin_start( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  // Enable the global interupt,
  RTLIB_INT_ENABLE();
  // Enable the IO board interrupts.
  error = dslin_board_interrupt_enable( board );
  dslin_board_error_print( board, error );
  error = dslin_channel_enable( channel );
  dslin_channel_error_print( channel, error );
}
void lin_interrupt_handler( void )
{
  Int16 subint = 0;
  // Acknowledge the interrupt.
  ds4330_dpmem_interrupt_clear( DS4330_1_BASE );
```

```
  // Call the decode function until no interrupt is pending.
  while( -1 != (subint = dslin_interrupt_decode( RTLIB_IO_MOD_IDX(DS4330_1_BASE) ) ) )
  {
    if( subint > -1   )
    {
      msg_info_printf( 0,0, " Ds4330 LIN interrupt received: %d", subint );
    }
  }
}
void lin_action( void )
{
  enum DSLIN_ERROR error = DSLIN_OK;
  RTLIB_SRT_ISR_BEGIN();
  dslin_board_update( board );
  // Use this function only for test purposes when no real LIN master available.
  error = dslin_channel_header_send( channel, 0x01 );
  dslin_channel_error_print( channel, error );
  RTLIB_SRT_ISR_END();
}
void main( void )
{
  lin_board_init();
  lin_channel_init();
  lin_node_init();
  lin_frame_init();
  lin_interrupt_init();
  lin_start();
  RTLIB_TIC_START();
  RTLIB_SRT_START( period, lin_action );
  RTLIB_INT_ENABLE();
  for(;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

References

LIN Error Handling.....26
LIN Interrupt Handling.....226

Example of Requesting LIN Interrupts

Example

The `dslin_interrupt_request` function is used to test the implemented interrupts and interrupt handles. The function requests an interrupt with a desired number. The advantage is that you can trigger interrupts but no LIN

communication is implemented or running. The example shows you how to request LIN interrupts.

```
#include <brtenv.h>
#include <dslin.h>
#include <ds4330.h>
void ds4330_lin_interrupt_handler( void )
{
  UInt16 next_subint = 0;
  Int16 subint = 0;
  // Acknowledge the interrupt.
  ds4330_dpmem_interrupt_clear( DS4330_1_BASE );
  // Call the decode function until no interrupt is pending.
  while( -1 != (subint = dslin_interrupt_decode( RTLIB_IO_MOD_IDX(DS4330_1_BASE) ) ) )
  {
    if( subint > -1   )
    {
      msg_info_printf( 0,0, " Ds4330 LIN interrupt received: %d", subint );
      // Calculate the next requestet interrupt number : 0,1,2,...2047,0,1,...
      if( subint < DSLIN_INTERRUPT_COUNT_MAX-1 )
      {
        next_subint = subint + 1;
      }
      else
      {
        next_subint = 0;
      }
      // Request the next interrupt.
      dslin_interrupt_request( RTLIB_IO_MOD_IDX(DS4330_1_BASE), next_subint );
    }
  }
}
void main( void )
{
  dslin_board_p board = NULL;
  enum DSLIN_ERROR error = DSLIN_OK;
  // Initialize the RTLIB.
  init();
  // Initialize the IO board.
  error = dslin4330_board_init( DS4330_1_BASE, &board );
  dslin_board_error_print( board, error);
  // Install the interrupt vector.
  install_phs_int_vector( DS4330_1_BASE, 0, ds4330_lin_interrupt_handler );
  // Clear the DS4330 interrupt.
  ds4330_dpmem_interrupt_clear( DS4330_1_BASE );
  // Enable the global interrupt,
  RTLIB_INT_ENABLE();
  // Enable the IO board interrupts.
  dslin_board_interrupt_enable( board );
  // Request the first interrupt.
  dslin_interrupt_request( RTLIB_IO_MOD_IDX(DS4330_1_BASE), 0 );
  for(;;)
  {
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

References

dslin_board_interrupt_enable

Syntax

```
define dslin_board_interrupt_enable(
      dslinboard_p board);
```

Include file

`dslin.h`

Purpose

To globally enable all initialized interrupts for a LIN board.

Description

Interrupts can be initialized and enabled for the different modules of the DSLIN API:

- Use `dslin_schedule_interrupt_init` to initialize schedule interrupts.
- Use `dslin_node_interrupt_init` to initialize node-related interrupts.
- Use `dslin_node_frame_interrupt_init` to initialize frame-related interrupts.
- Use `dslin_interrupt_enable` to enable the single interrupts.

Parameters

board Pointer to the LIN board. The board must be already initialized by the corresponding initialization function, for example, `dslin4330_board_init`.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The channel is enabled.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_WRONG_TYPE	Wrong input pointer type
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Example

For examples on how to enable interrupts globally, see Example of Using LIN Frame Interrupts on page 227 and Example of Requesting LIN Interrupts on page 229.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

References

dslin_board_interrupt_disable..... 232
dslin_interrupt_disable..... 243
dslin_interrupt_enable..... 242
dslin_node_frame_interrupt_init..... 238
dslin_node_interrupt_init..... 235
dslin_schedule_interrupt_init..... 240
dslin4330_board_init..... 35
LIN Error Handling..... 26
Standard Defines..... 15

dslin_board_interrupt_disable

Syntax

```
define dslin_board_interrupt_disable(
      dslinboard_p board);
```

Include file

dslin.h

Purpose

To globally disable all interrupts for a LIN board.

Description

Although the single interrupts are enabled by the `dslin_interrupt_enable` function, you can disable all interrupts of a LIN board at once.

Single interrupts can also disabled by using the `dslin_interrupt_disable` function.

Parameters

board Pointer to the LIN board. The board must be already initialized by the corresponding initialization function, for example, `dslin4330_board_init`.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The channel is enabled.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if channel == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
`DSLIN_WRONG_TYPE`	Wrong input pointer type
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

References

dslin_board_interrupt_enable..... 231
dslin_interrupt_disable..... 243
dslin_interrupt_enable..... 242
dslin_node_frame_interrupt_init..... 238
dslin_node_interrupt_init..... 235
dslin_schedule_interrupt_init..... 240
dslin4330_board_init..... 35
LIN Error Handling..... 26
LIN Interrupt Handling..... 226
Standard Defines..... 15

dslin_frame_interrupt_init

Syntax

```
enum DSLIN_ERROR dslin_frame_interrupt_init(
   dslin_frame_p frame,
   UInt32 interrupt_number,
   enum DSLIN_FRAME_INTERRUPT_TYPE type);
```

Include file

`dslin.h`

Purpose

To assign the number of a subinterrupt to a LIN frame event independently from a frame identifier.

Description

You can setup interrupts which are independent from the frame identifier. So it is possible to call a task even if the frame identifier changes. The interrupts are event-triggered. The frame events are defined in the `FRAME_INTERRUPT_TYPE` enumeration. The following frame events are used as trigger sources:

Frame Event	Meaning
`DSLIN_FRAME_INT_HEADER_RECEIVED`	The interrupt is triggered if the header was received correctly.
`DSLIN_FRAME_INT_HEADER_SEND`	The interrupt is triggered if the header was transmitted correctly.
`DSLIN_FRAME_INT_HEADER_SEND_BIT_ERROR`	The interrupt is triggered if the header was transmitted and a bit error was detected.
`DSLIN_FRAME_INT_RESPONSE_RECEIVED`	The interrupt is triggered if the response was received correctly.
`DSLIN_FRAME_INT_RESPONSE_SEND`	The interrupt is triggered if the response was sent correctly.
`DSLIN_FRAME_INT_RESPONSE_SEND_BIT_ERROR`	The interrupt is triggered if the response was transmitted and a bit error was detected.
`DSLIN_FRAME_INT_RESPONSE_CHECKSUM_ERROR`	The subinterrupt is triggered if an error in the checksum field of a received response (RX frame) was detected.
`DSLIN_FRAME_INT_SNR_ERROR`	The subinterrupt is triggered if a slave-not-responding error was detected. This error occurs if a response is not fully completed within the maximum frame length.

Parameters

frame Pointer to a LIN frame.

interrupt_number Specifies the interrupt number to be triggered. The value is returned by the `dslin_interrupt_decode` function when the specified event occurs. For a DS4330, valid values are within 0 … 2047.

type Specifies the event source. The available event sources are specified in the FRAME_INTERRUPT_TYPE enumeration (see above).

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The interrupt has been successfully initialized.
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if frame == NULL. The error can occur if a pointer is used which is not initialized or the initialization has failed.
DSLIN_ERR_INTERRUPT_COUNT	The specified interrupt number is not valid. The valid range is within 0 … 2047.
DSLIN_ERR_FRAME_INTERRUPT_ILLEGAL	The defined interrupt is not valid.

Example

The following example shows how to specify a frame event triggered interrupt.

```
// Generate an interrupt with the number '1' when the response
// for the frame were received correctly
dslin_frame_interrupt_init( frame, 1,
                 DSLIN_FRAME_INT_RESPONSE_RECEIVED);
```

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

References

Data Types and Enumerations 17
dslin_interrupt_decode 244
dslin_node_frame_interrupt_init 238
dslin_node_interrupt_init 235
LIN Error Handling 26

dslin_node_interrupt_init

Syntax

```
enum DSLIN_ERROR dslin_node_interrupt_init(
   dslin_node_p node,
   UInt16 interrupt_number,
   enum DSLIN_NODE_INTERRUPT_TYPE type);
```

Include file

dslin.h

Purpose

To assign a subinterrupt number to a LIN node event.

Description

The interrupts are triggered if the specified node events occur. The node events are defined in the NODE_INTERRUPT_TYPE enumeration. Call the `dslin_interrupt_decode` function to get the interrupt number of the triggered interrupt. See dslin_interrupt_decode on page 244.

The following node events are used as trigger sources:

Node Event	Meaning
`DSLIN_NODE_INT_IDPAR_ERROR`	The ID parity error indicates that the slave node received a wrong ID parity bit.
`DSLIN_NODE_INT_SYNCH_FIELD_ERROR`	The inconsistent field error indicates that the node has received a header with a synchronization byte different from 0x55.
`DSLIN_NODE_INT_NO_BUS_ACTIVITY`	A No-Bus-Activity error is detected when no synchronization break was received for more than 25000 bit times since the reception of the last synchronization break byte.
`DSLIN_NODE_INT_EXTRABYTES_DETECTED`	The event indicates bytes on the LIN bus that cannot assigned to a certain LIN header or LIN response. This may be caused if the receive length of the RX frame is shorter than the length of the TX frame with the same identifier on the same LIN bus.
`DSLIN_NODE_INT_SLEEP_CMD_RECEIVED`	The interrupt is triggered after a sleep mode command has been received from the LIN master. The sleep mode command is a master command frame that contains 0x00 as the first data field.
`DSLIN_NODE_INT_WAKE_CMD_RECEIVED`	The interrupt is triggered when a node is in sleep mode and receives a wake-up command from any node in the LIN bus.
`DSLIN_NODE_INT_TX_ERROR_THRESHOLD_EXCEEDED`	Indicates an overflow of the TX error counter. The TX threshold is set with `dslin_node_init` and `dslin_node_tx_error_threshold_set`.
`DSLIN_NODE_INT_RX_ERROR_THRESHOLD_EXCEEDED`	Indicates an overflow of the RX error counter. The RX threshold is set with `dslin_node_init` and `dslin_node_rx_error_threshold_set`.
`DSLIN_NODE_INT_TIMEOUT_AFTER_WAKEUP`	Indicates a timeout after a wake-up command was received. The timeout occurs if no header was received within 128 bit-times after a wake-up.

Parameters

node Pointer to a LIN node

interrupt_number Specifies the interrupt number which is returned by the `dslin_interrupt_decode` function if the specified event occurs. Valid values are within 0 … DSLIN_BOARD_INTERRUPT_COUNT – 1 (see Standard Defines on page 15).

type Specifies the event source. The available event sources are specified in the NODE_INTERRUPT_TYPE enumeration. See Data Types and Enumerations on page 17.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	The interrupt has been successfully initialized.
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or the initialization has failed.
DSLIN_ERR_INTERRUPT_COUNT	The specified interrupt number is not valid.
DSLIN_ERR_NODE_INTERRUPT_ILLEGAL	The defined interrupt is not valid.

Example

The following example shows how to specify a node event triggered interrupt.

```
// Generate an interrupt with the number '2' if the node
// receives a sleep command.
dslin_node_interrupt_init( node, 2,
                           DSLIN_NODE_INT_SLEEP_CMD_RECEIVED );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

References

dslin_board_interrupt_disable..... 232
dslin_board_interrupt_enable..... 231
dslin_interrupt_decode..... 244
dslin_interrupt_disable..... 243
dslin_interrupt_enable..... 242
dslin_node_frame_interrupt_init..... 238
dslin_node_init..... 104
dslin_node_rx_error_threshold_set..... 109
dslin_node_tx_error_threshold_set..... 107
dslin_schedule_interrupt_init..... 240
LIN Error Handling..... 26
LIN Interrupt Handling..... 226
Standard Defines..... 15

dslin_node_frame_interrupt_init

Syntax

```
enum DSLIN_ERROR dslin_node_frame_interrupt_init(
   dslin_node_p node,
   UInt16 interrupt_number,
   enum DSLIN_FRAME_INTERRUPT_TYPE type,
   UInt8 identifier);
```

Include file

`dslin.h`

Purpose

To assign the number of a subinterrupt to a LIN frame event occurring on a LIN node.

Description

The interrupts are event-triggered. The frame events are defined in the FRAME_INTERRUPT_TYPE enumeration. The following frame events are used as trigger sources:

Frame Event	Meaning
`DSLIN_FRAME_INT_HEADER_RECEIVED`	The interrupt is triggered if the header was received correctly.
`DSLIN_FRAME_INT_HEADER_SEND`	The interrupt is triggered if the header was transmitted correctly.
`DSLIN_FRAME_INT_HEADER_SEND_BIT_ERROR`	The interrupt is triggered if the header was transmitted and a bit error was detected.
`DSLIN_FRAME_INT_RESPONSE_RECEIVED`	The interrupt is triggered if the response was received correctly.
`DSLIN_FRAME_INT_RESPONSE_SEND`	The interrupt is triggered if the response was sent correctly.
`DSLIN_FRAME_INT_RESPONSE_SEND_BIT_ERROR`	The interrupt is triggered if the response was transmitted and a bit error was detected.
`DSLIN_FRAME_INT_RESPONSE_CHECKSUM_ERROR`	The subinterrupt is triggered if an error in the checksum field of a received response (RX frame) was detected.
`DSLIN_FRAME_INT_SNR_ERROR`	The subinterrupt is triggered if a slave-not-responding error was detected. This error occurs if a response is not fully completed within the maximum frame length.

Tip

This function cannot be used if the frame identifier changes during run-time. In this case, use the `dslin_frame_interrupt_init` function.

Parameters

node Pointer to a LIN node.

interrupt_number Specifies the interrupt number to be triggered if the specified event occurs. The valid values are within 0 … `DSLIN_BOARD_INTERRUPT_COUNT_MAX` – 1 (see Standard Defines on page 15).

type Specifies the event source. The available event sources are specified in the FRAME_INTERRUPT_TYPE enumeration (see above).

identifier Specifies an identifier of a frame. The function only generates an interrupt for frames with this identifier.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	The interrupt has been successfully initialized.
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_NULL_POINTER`	NULL pointer access; occurs if node == NULL. The error can occur if a pointer is used which is not initialized or the initialization has failed.
`DSLIN_ERR_INTERRUPT_COUNT`	The specified interrupt number is not valid. The valid range is within 0 … `DSLIN_INTERRUPT_COUNT_MAX` – 1.
`DSLIN_ERR_FRAME_INTERRUPT_ILLEGAL`	The defined interrupt is not valid.

Example

The following example shows how to specify a frame event triggered interrupt. For a detailed example of using frame interrupts, see Example of Using LIN Frame Interrupts on page 227.

```
// Generate an interrupt with the number '1' when the response
// for identifier '0x12' were received correctly
// by the LIN node. The response length is '8' bytes.
dslin_node_frame_interrupt_init( node, 1,
                 DSLIN_FRAME_INT_RESPONSE_RECEIVED, 0x12, 8 );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

References

Data Types and Enumerations..... 17
dslin_frame_interrupt_init..... 234
dslin_node_interrupt_init..... 235
LIN Error Handling..... 26
LIN Interrupt Handling..... 226
Standard Defines..... 15

dslin_schedule_interrupt_init

Syntax

```
enum DSLIN_ERROR dslin_schedule_interrupt_init(
   dslin_schedule_p schedule,
   UInt16 interrupt_number,
   enum DSLIN_SCHEDULE_INTERRUPT_TYPE type);
```

Include file

`dslin.h`

Purpose

To connect a subinterrupt number with a LIN schedule event.

Description

The `dslin_schedule_interrupt_init` function is used to initialize schedule-related interrupts. The specified subinterrupt number is returned by the `dslin_interrupt_decode` function when the specified schedule event occurs. The available events are specified in the SCHEDULE_INTERRUPT_TYPE enumeration:

Predefined Symbol	Meaning
`DSLIN_SCHEDULE_INT_STARTED`	The schedule is started from the beginning. The interrupt is triggered at the header's beginning, as soon as the synchronization break signal is generated.
`DSLIN_SCHEDULE_INT_COMPLETED`	The schedule was successfully executed. The interrupt is triggered when the frame was successfully transferred. If the slave-not-responding error (illegal delay) occurs the interrupt is also triggered. It is always triggered before the next schedule starts.
`DSLIN_SCHEDULE_INT_ABORTED`	The schedule was interrupted before completion.
`DSLIN_SCHEDULE_INT_RESTARTED`	The schedule was restarted after termination.

Parameters

schedule Pointer to a LIN schedule

interrupt_number Specifies the interrupt number within the range 0 … DSLIN_BOARD_INTERRUPT_COUNT_MAX –1 (see Standard Defines on page 15).

type Specifies the event source that triggers the interrupt.

Return value

The function returns the following error codes:

Error Code	Meaning
DSLIN_OK	A new schedule has been created successfully.
DSLIN_ERR_NULL_POINTER	NULL pointer access; occurs if schedule == NULL. The error can occur if a pointer is used which is not initialized or for which the initialization has failed.
DSLIN_WRONG_TYPE	Wrong input pointer type
DSLIN_COMMUNICATION_OVERLOAD	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
DSLIN_ERR_INTERRUPT_COUNT	Too many interrupts are defined. The maximum number of interrupts is 2048.
DSLIN_ERR_SCHEDULE_INTERRUPT_ILLEGAL	Wrong interrupt type selected. The valid types are specified in the DSLIN_SCHEDULE_INTERRUPT_TYPE enumeration. See above.

Example

```
// Generate an interrupt with the number '2' if the schedule terminates.
dslin_schedule_interrupt_init( schedule, 2,
                               DSLIN_SCHEDULE_INT_COMPLETED);
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

References

dslin_interrupt_enable

Syntax

```
enum DSLIN_ERROR dslin_interrupt_enable(
   UInt16 board_index,
   UInt16 interrupt_number);
```

Include file

`dslin.h`

Purpose

To enable an interrupt with a specific interrupt number for a LIN board.

> **Note**
>
> You also have to enable the board interrupt with the `dslin_board_interrupt_enable` function.

Parameters

board_index Index of the board on the PHS bus within the range 0 … 15.

interrupt_number Specifies the interrupt number within the range 0 … DSLIN_BOARD_INTERRUPT_COUNT_MAX – 1 (see Standard Defines on page 15.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new schedule has been created successfully.
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_BOARD_NOT_INITIALIZED`	The board was not initialized. Use `dslin4330_board_init` to initialize the LIN board.
`DSLIN_ERR_INTERRUPT_COUNT`	Too many interrupts are defined. The maximum number of interrupts is 2048.
`DSLIN_ERR_INTERRUPT_NOT_INITIALIZED`	The interrupt was not initialized. Use the interrupt initialization function of the corresponding interrupt type.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

References

dslin_interrupt_disable

Syntax

```
enum DSLIN_ERROR dslin_interrupt_disable(
   UInt16 board_index,
   UInt16 interrupt_number);
```

Include file

`dslin.h`

Purpose

To disable an enabled interrupt.

Parameters

board_index Index of the board on the PHS bus within the range 0 … 15.

interrupt_number Specifies the interrupt number within the range 0 … `DSLIN_BOARD_INTERRUPT_COUNT_MAX` – 1.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new schedule has been created successfully.
`DSLIN_COMMUNICATION_OVERLOAD`	The LIN board was not ready to accept the command. This indicates that the LIN board is overloaded.
`DSLIN_ERR_BOARD_NOT_INITIALIZED`	The board was not initialized. Use `dslin4330_board_init` to initialize the LIN board.
`DSLIN_ERR_INTERRUPT_COUNT`	The interrupt numbers are limited to DSLIN_BOARD_INTERRUPT_COUNT_MAX – 1.

Error Code	Meaning
DSLIN_ERR_INTERRUPT_NOT_INITIALIZED	The interrupt was not initialized. Use the interrupt initialization function of the corresponding interrupt type.

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

References

dslin_interrupt_enable 242
dslin_node_frame_interrupt_init 238
dslin_node_interrupt_init 235
dslin_schedule_interrupt_init 240
LIN Error Handling 26
LIN Interrupt Handling 226
Standard Defines 15

dslin_interrupt_decode

Syntax

```
Int32 DSLIN_ERROR dslin_interrupt_decode(
   UInt16 board_index);
```

Include file

`dslin.h`

Purpose

To read the interrupt number of the interrupt triggered by the LIN board.

Description

The interrupt handling works according to the FIFO principle. The first triggered interrupt is the oldest interrupt and is read out first. The `dslin_interrupt_decode` function delivers the number of the interrupt triggered by the LIN board. After the number is read, the interrupt is processed and the corresponding interrupt number does not reappear until the next interrupt with that number is triggered.

Parameters

board_index Index of the board on the PHS bus within the range 0 … 15.

Return value

The function returns the following values:

Value	Meaning
– 1	No interrupt number available on the specified board.
0 … DSLIN_BOARD_INTERRUPT_COUNT_MAX – 1	Oldest interrupt number.

Example

The following example shows how to use the `dslin_interrupt_decode` function. For detailed examples on using interrupts, see Example of Using LIN Frame Interrupts on page 227 and Example of Requesting LIN Interrupts on page 229.

```
void ds4330_lin_interrupt_handler( void )
  {
    Int16 subint = 0;
    // Acknowledge the interrupt.
    ds4330_dpmem_interrupt_clear( DS4330_1_BASE );
    // Call the decode function until no interrupt is pending.
    while( -1 != (subint = dslin_interrupt_decode
                    ( RTLIB_IO_MOD_IDX(DS4330_1_BASE) ) ) )
    {
      if( subint > -1   )
      {
        // Call your interrupt driven functions here.
      }
    }
  }
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

References

ds4330_dpmem_interrupt_clear..... 38
dslin_node_frame_interrupt_init..... 238
dslin_node_interrupt_init..... 235
dslin_schedule_interrupt_init..... 240
LIN Error Handling..... 26
LIN Interrupt Handling..... 226
Standard Defines..... 15

dslin_interrupt_request

Syntax

```
UInt32 DSLIN_ERROR dslin_interrupt_request(
   UInt16 board_index,
   UInt16 interrupt_number);
```

Include file

`dslin.h`

Purpose

To request an interrupt of a LIN board.

> **Note**
>
> The `dslin_interrupt_request` function is only used for test purposes.

Description

The `dslin_interrupt_request` function is used for testing the implemented interrupts and interrupt handles. The function requests an interrupt with a specified number. The advantage is that you can trigger interrupts but no LIN communication is implemented or is running.

Parameters

board_index Index of the board on the PHS bus within the range 0 … 15.

interrupt_number Specifies the subinterrupt generated by the LIN board.

Return value

The function returns the following error codes:

Error Code	Meaning
`DSLIN_OK`	A new schedule has been created successfully.
`DSLIN_ERR_BOARD_NOT_INITIALIZED`	The board was not initialized. Use `dslin4330_board_init` to initialize the LIN board.

Example

The following example shows how to request the subinterrupt ‘1’ from a DS4330 board.

```
dslin_interrupt_request( RTLIB_IO_MOD_IDX(DS4330_1_BASE), 1 );
```

Execution times

For information, refer to Function Execution Times on page 249.

Related topics

Basics

Using Interrupts (DS4330 Features 🕮)

Examples

Example of Requesting LIN Interrupts..... 229

References

dslin_node_frame_interrupt_init..... 238
dslin_node_interrupt_init..... 235
dslin_schedule_interrupt_init..... 240
dslin4330_board_init..... 35
LIN Error Handling..... 26
LIN Interrupt Handling..... 226
Standard Defines..... 15

Function Execution Times

Introduction

To give you the mean function execution times and basic information on the test environment used.

Where to go from here

Information in this section

Information in other sections

Local Interconnect Network (LIN) (DS4330 Features 🕮)
Provides basic information about Local Interconnect Network (LIN).

Information on the Test Environment

Introduction

The execution times of the C functions can vary, since they depend on different factors. The measured execution times are influenced by the test environment used.

Test environment

The execution time of a function can vary, since it depends on different factors, for example:

- CPU clock and bus clock frequency of the processor board used
- Optimization level of the compiler
- Use of inlining parameters

The test programs that are used to measure the execution time of the functions listed below have been generated and compiled with the default settings of the `down<xxxx>` tool (optimization and inlining). The execution times in the tables below are always the mean measurement values.

The properties of the processor boards used are:

	DS1006
CPU clock	2.6 GHz / 3.0 GHz
Bus clock	133 MHz

Measured Execution Times

Execution times

Execution times are available for the following RTLib units:

- Board-related functions
- Channel-related functions
- Node-related functions
- Frame-related functions

Note

The following execution times contain mean values for a sequence of I/O accesses. The execution time of a single call might be lower because of buffered I/O access.

Board-related functions

The following execution times have been measured for updating the LIN board:

Function	**Load**	**Mean Execution Time**	
		DS1006 with 2.6 GHz	**DS1006 with 3.0 GHz**
`dslin_board_update`	No data	1.45 µs	1.19 µs
	1xTX, 2xRX, 8 byte	12.29 µs	12.06 µs
	1xTX, 4xRX, 8 byte	18.93 µs	18.64 µs
	1xTX, 16xRX, 8 byte	59.78 µs	58.53 µs

Function	Load	Mean Execution Time	
		DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`dslin_board_error_print`	DSLIN_OK	0.21 µs	0.056 µs
	Error	14.71 µs	14.19 µs

Channel-related functions

The following execution times have been measured for handling LIN channels:

Function	Mean Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`dslin_channel_lookup` (16 channels)	0.09 µs	0.077 µs
`dslin_channel_create`	8.34 µs	7.75 µs
`dslin_channel_init`	1.7 µs	2.125 µs
`dslin_channel_enable`	3.56 µs	3.56 µs
`dslin_channel_disable`	0.16 µs	0.19 µs
`dslin_channel_transceiver_set`	0.16 µs	0.030 µs
`dslin_channel_transceiver_sleep`	1.15 µs	1.73 µs
`dslin_channel_termination_set`	0.04 µs	0.031 µs
`dslin_channel_baudrate_set`	18.69 µs	14.118 µs
`dslin_channel_apply_settings`	1.55 µs	1.50 µs
`dslin_channel_restore_settings`	16.80 µs	1.68 µs
`dslin_channel_is_wake`	0.19 µs	0.03 µs
`dslin_channel_rx_monitor_init`	0.55 µs	0.41 µs
`dslin_channel_rx_monitor_clear`	0.24 µs	0.16 µs
`dslin_channel_rx_monitor_client_init`	0.15 µs	0.10 µs
`dslin_channel_rx_monitor_client_read`	0.17 µs	0.18 µs
`dslin_channel_tx_response_write`	0.21 µs	0.18 µs
`dslin_channel_board_get`	0.04 µs	0.03 µs

Node-related functions

The following execution times have been measured for handling LIN nodes:

Function	Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`dslin_node_parity_offset_set` (16 channels and 16 nodes)	0.75 µs	–
`dslin_node_create`	16.16 µs	17.79 µs
`dslin_node_init`	1.17 µs	1.24 µs
`dslin_node_enable`	0.31 µs	0.26 µs
`dslin_node_disable`	1.46 µs	1.36 µs

Function	Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`dslin_node_rx_error_threshold_set`	0.035 µs	0.030 µs
`dslin_node_apply_settings`	0.25 µs	0.03 µs
`dslin_node_command_wakeup`	1.54 µs	1.56 µs
`dslin_node_info_rx_err_get`	0.17 µs	0.20 µs
`dslin_node_configuration_init`	0.14 µs	0.16 µs
`dslin_node_initial_nad_set`	0.26 µs	0.29 µs
`dslin_node_initial_nad_get`	0.053 µs	0.046 µs
`dslin_node_current_nad_set`	0.14 µs	0.10 µs
`dslin_node_current_nad_get`	0.054 µs	0.045 µs
`dslin_node_supplier_id_set`	0.054 µs	0.049 µs
`dslin_node_supplier_id_get`	0.051 µs	0.12 µs
`dslin_node_function_id_set`	0.054 µs	0.048 µs
`dslin_node_function_id_get`	0.059 µs	0.046 µs
`dslin_node_variant_id_set`	0.064 µs	0.048 µs
`dslin_node_variant_id_get`	0.053 µs	0.046 µs
`dslin_node_readbyid_positive_response_set`	0.11 µs	0.12 µs
`dslin_node_readbyid_positive_response_get`	0.28 µs	0.19 µs
`dslin_node_configuration_service` (without data communication)	0.19 µs	0.17 µs
`dslin_node_configuration_service` (with data communication)	7.43 µs	7.417 µs

Frame-related functions

The following execution times have been measured for handling LIN frames:

Function	Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`dslin_frame_lookup` (16 channels and 16 nodes)	2.15 µs	0.42 µs
`dslin_frame_create`	3.84 µs	3.63 µs
`dslin_frame_tx_init`	2.0 µs	8.28 µs
`dslin_frame_rx_init`	5.87 µs	6.22 µs
`dslin_frame_rx_eventtrig_init`	0.31 µs	0.24 µs
`dslin_frame_enable`	0.047 µs	0.038 µs
`dslin_frame_disable`	0.11 µs	0.084 µs
`dslin_frame_lock`	0.15 µs	0.035 µs
`dslin_frame_unlock`	0.20 µs	0.032 µs
`dslin_frame_id_set`	5.3 µs	0.155 µs
`dslin_frame_tx_data_set`	0.19 µs	0.054 µs

Function	Execution Time	
	DS1006 with 2.6 GHz	DS1006 with 3.0 GHz
`dslin_frame_tx_response_delay_set`	0.05 µs	0.036 µs
`dslin_frame_tx_checksum_set`	0.05 µs	0.043 µs
`dslin_frame_apply_settings`	1.60 µs	1.526 µs
`dslin_frame_info_data_get`	0.13 µs	0.0423 µs
`dslin_frame_info_timestamp_get`	0.051 µs	0.043 µs
`dslin_frame_info_id_get`	0.16 µs	0.11 µs
`dslin_frame_info_checksum_get`	0.045 µs	0.042 µs
`dslin_frame_info_status_get`	0.11 µs	0.16 µs
`dslin_checksum_calc`	0.12 µs	0.12 µs

C

D

E

F