DS4302 CAN 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

The DS4302 Real-Time Library (RTLib) provides the C functions and macros you need to program the DS4302 CAN Interface Board.

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.

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.

Slave CAN Access Functions

Where to go from here

Information in this section

Basics on Slave CAN Access Functions

Introduction

Provides basics on the communication principles between the master processor board and the slave CAN subsystem, and on the CAN error message types.

Where to go from here

Information in this section

Basic Principles of Master-Slave Communication

Introduction

The master processor board uses slave access functions to control the slave CAN subsystem and exchange data with it.

Note

You have to initialize the communication between the master and the slaves. Refer to `ds4302_can_communication_init` on page 21.

Communication process

- The master application initializes the required slave functions based on the CAN controller.
- The message register functions write all required values to the appropriate handle, e.g. (ds4302_canMsg). The appropriate request and read functions get the information from this handle later on.
- To perform a read operation, the master processor board requests that the previously registered slave function be carried out. The slave then performs the required functions independently and writes the results back to the dual-port memory. If more than one function is required simultaneously – for example, as a result of different tasks on the processor board – priorities must be considered.
- The master processor board application reads/writes the input/output data from/to the slave.

Note

The master processor board reads the slave results from the dual-port memory in the order in which they occur, and then reads them into a buffer, regardless of whether a particular result is needed. The read functions copy data results from the buffer into the processor board application variables.

Function classes

Slave applications are based on communication functions that are divided into separate classes as follows:

- *Initialization functions* initialize the slave functions.
- *Register functions* make the slave functions known to the slave.
- *Request functions* require that the previously registered slave function be carried out by the slave.
- *Read functions* fetch data from the dual-port memory and convert or scale the data, if necessary.
- *Write functions* convert or scale the data if necessary and write them into the dual-port memory.

Error handling

When an error occurs with initialization or register functions, an error message appears from the global message module. Then the program ends.

Request, read, and write functions return an error code. The application can then handle the error code.

Communication channels and priorities

This communication method, along with the command table and the transfer buffer, can be initialized in parallel for the statically defined communication channels with fixed priorities (0 … 6). Like communication buffers, each communication channel has access to memory space in the dual-port memory so that slave error codes can be transferred.

Related topics

Basics

Basics on the RTI CAN Blockset (RTI CAN Blockset Reference 🕮)
CAN Support (DS4302 Features 🕮)

CAN Error Message Types

Introduction

The functions of the CAN environment report error, warning, and information messages if a problem occurs. These messages are displayed by the **Message Viewer** of the experiment software. The message consists of an error number, the function name, the board index (offset of the PHS-bus address) and the message text. For example:

```
Error[121]: ds4302_can_channel_init (6,..) baudrate: too low
(min. 10 kBaud)!
```

Message Number	Message Type
100 … 249	Error
250 … 349	Warning
400 … 500	Information

Related topics

References

Data Structures for CAN

Introduction

The data structures provide information on channels, services, and messages to be used by other functions. Using CAN RTLib functions, you access the structures *automatically*. You do not have to access them explicitly in your application.

Where to go from here

Information in this section

Information in other sections

ds4302_canChannel

Purpose

The `ds4302_canChannel` structure contains information on the CAN channel capabilities.

Syntax

```
typedef struct
{
   UInt32 base;
   Int32 index;
   UInt32 channel;
   UInt32 btr0;
   UInt32 btr1;
   UInt32 frequency;
   UInt32 mb15_format;
   UInt32 busoff_int_number;
} ds4302_canChannel;
```

Include file `Ds4302.h`

Members

base The PHS-bus base address is provided by the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced`. This parameter is read-only.

index Table index allocated by the message register function. This parameter is read-only.

channel Number of the used CAN channel. This parameter is provided by the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced`. This parameter is read-only.

btr0 Value of Bit Timing Register 0. This parameter is provided by the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced`. This parameter is read-only.

btr1 Value of Bit Timing Register 1. This parameter is provided by the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced`. This parameter is read-only.

frequency Frequency of the CAN controller. This parameter is provided by the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced`. This parameter is read-only.

mb15_format Format of mailbox 15. Mailbox 15 is a double-buffered receive unit of the CAN. Use this mailbox for the message type most frequently used in your application. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_STD`	11-bit standard format, CAN 2.0A
`DS4302_CAN_EXT`	29-bit extended format, CAN 2.0B

This parameter is provided by the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced`. This parameter is read-only.

busoff_int_number Subinterrupt generated when the CAN channel goes bus off. This parameter is provided by the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced`. This parameter is read-only.

Related topics

References

ds4302_can_channel_init.. 25
ds4302_can_channel_init_advanced.. 27

ds4302_canService

Purpose

The `ds4302_canService` structure contains information on the CAN service. The CAN service provides information on errors and status information (see the `type` parameter).

Syntax

```
typedef struct
{
   UInt32 busstatus;
   UInt32 stdmask;
   UInt32 extmask;
   UInt32 msg_mask15;
   UInt32 tx_ok;
   UInt32 rx_ok;
   UInt32 crc_err;
   UInt32 ack_err;
   UInt32 form_err;
   UInt32 stuffbit_err;
   UInt32 bit1_err;
   UInt32 bit0_err;
   UInt32 rx_lost;
   UInt32 data_lost;
   UInt32 mailbox_err;
   UInt32 c52_err;
   UInt32 p2in;
   UInt32 data0;
   UInt32 data1;
   UInt16 txqueue_overflowcnt_std;
   UInt16 txqueue_overflowcnt_ext;
   UInt32 module;
   UInt32 queue;
   UInt32 type;
   Int32 index;
} ds4302_canService;
```

Include file

`Ds4302.h`

Members

data0 Contains returned data from the function `ds4302_can_service_read`.

data1 Contains returned data from the function `ds4302_can_service_read`.

Note

For each service, the structure provides its own member. For the meaning of the services, refer to the `type` parameter. The members `data0` and `data1` remain in the structure for compatibility reasons.

module The CAN module is provided by the function `ds4302_can_service_register`. This parameter is read-only.

queue This parameter is provided by the function `ds4302_can_service_register`. This parameter is read-only.

type Type of the service already allocated by the previously performed register function. Once a service is registered on the slave, it can deliver a value. The return value will be stored in the structure members `data0` and `data1`. This parameter is provided by the `ds4302_can_service_register` function. This parameter is read-only.

Note

Start the CAN channel with the enabled status interrupt to use the following predefined services (see `ds4302_can_channel_start` on page 32).

Predefined Symbol	Meaning
`DS4302_CAN_SERVICE_TX_OK`	Number of successfully sent TX/RM/RQTX messages
`DS4302_CAN_SERVICE_RX_OK`	Number of successfully received RX/RQRX messages
`DS4302_CAN_SERVICE_CRC_ERR`	Number of CRC errors
`DS4302_CAN_SERVICE_ACK_ERR`	Number of acknowledge errors
`DS4302_CAN_SERVICE_FORM_ERR`	Number of format errors
`DS4302_CAN_SERVICE_BIT1_ERR`	Number of Bit1 errors
`DS4302_CAN_SERVICE_BIT0_ERR`	Number of Bit0 errors
`DS4302_CAN_SERVICE_STUFFBIT_ERR`	Number of stuff bit errors

Note

It is not necessary to start the CAN channel with the enabled status interrupt if you are using only the following predefined services (see `ds4302_can_channel_start` on page 32).

Predefined Symbol	Meaning
`DS4302_CAN_SERVICE_RX_LOST`	Number of lost RX messages. The RX lost counter is incremented when a received message is overwritten in the receive mailbox before the message has been read.
`DS4302_CAN_SERVICE_DATA_LOST`	Number of data lost errors. The data lost counter is incremented when the data of a message is overwritten before the data has been written to the communication queue.
`DS4302_CAN_SERVICE_MAILBOX_ERR`	Number of mailbox errors. If a message to be sent cannot be assigned to a mailbox, the mailbox error counter is increased by one. For possible error reasons, see below.
`DS4302_CAN_SERVICE_BUSSTATUS`	Status of the CAN controller. For the predefined values, see below.
`DS4302_CAN_SERVICE_STDMASK`	Status of the global standard mask register

Predefined Symbol	Meaning
`DS4302_CAN_SERVICE_EXTMASK`	Status of the global extended mask register
`DS4302_CAN_SERVICE_MSG_MASK15`	Status of the message 15 mask register
`DS4302_CAN_SERVICE_TXQUEUE_ OVERFLOW_COUNT`	Overflow counter of the transmit queue. The overflow counter (STD or XTD message format) is incremented when the queue is filled (64 messages) and a new message arrives. Depending on the `overrun_policy` parameter set with `ds4302_can_msg_txqueue_init`, the new message overwrites the oldest message entry or is ignored. The overflow counters are 16-bit counters. The wraparound occurs after 65535 overflows.
`DS4302_CAN_SERVICE_C252_ERR`	Number of C252 transceiver errors. The value will be increased when one of the following CAN bus events occurs: ▪ CAN-H wire interrupted, ▪ CAN-L wire interrupted, ▪ CAN-H shorted to battery, ▪ CAN-L shorted to ground, ▪ CAN-L shorted to battery, ▪ CAN-H shorted to ground, ▪ CAN-L mutually shorted to CAN-H.
`DS4302_CAN_SERVICE_P2OUT`	Contents of the P2OUT register on the slave DS4302 CAN controller.

index Table index already allocated by the register function `ds4302_can_service_register`. This parameter is read-only.

Parameter type

Additional information on the service functions provided by the type parameter:

DS4302_CAN_SERVICE_MAILBOX_ERR Provides the number of mailbox errors. The following table describes possible error reasons and how to you can avoid these errors:

Error reason	Description	Workaround
All mailboxes are filled.	The messages are not removed from a mailbox fast enough.	Decrease the timeout value of all messages of the corresponding CAN channel and restart the application.
Conflict between two message IDs.	This error can occur if standard and extended messages are used on a CAN channel simultaneously. Check whether all messages are sent according to your requirements. It is not possible to remove remote messages temporarily from a mailbox. Check for a possible problem	Try the first element of the following list. If the error counter still increases, try the next one: ▪ Decrease the timeout value for messages with the same format as mailbox 14 – i.e., with the opposite format of mailbox 15 (refer to `ds4302_can_channel_init`). ▪ Initialize the `mb15_format` parameter with the other format when calling `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced`. ▪ Choose different message IDs for messages of mailbox 14 format. ▪ Do not use standard and extended messages on one CAN channel simultaneously.

Error reason	Description	Workaround
	with a registered remote message.	

DS4302_CAN_SERVICE_BUSSTATUS Provides bus status information; the following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_BUSOFF_STATE`	The CAN channel disconnects itself from the CAN bus. Use `ds4302_can_channel_BOff_return` to recover from the bus off state.
`DS4302_CAN_WARN_STATE`	The CAN controller is still active. The CAN controller recovers from this state automatically.
`DS4302_CAN_ACTIVE_STATE`	The CAN controller is active.

Note

After calling `ds4302_can_channel_BOff_return`, the service DS4302_CAN_SERVICE_BUSSTATUS will not return DS4302_CAN_BUSOFF_STATE.

Related topics

References

ds4302_can_channel_BOff_go	35
ds4302_can_channel_BOff_return	36
ds4302_can_channel_init	25
ds4302_can_channel_init_advanced	27
ds4302_can_channel_start	32
ds4302_can_msg_txqueue_init	71
ds4302_can_service_read	88
ds4302_can_service_register	85

ds4302_canMsg

Purpose

The `ds4302_canMsg` structure contains information on the CAN message capabilities.

Syntax

```
typedef struct{
   double timestamp;
   Float32 deltatime;
   Float32 delaytime;
   Int32 processed;
   UInt32 datalen;
   UInt32 data[8];
   UInt32 identifier;
   UInt32 format;
   UInt32 module;
   UInt32 queue;
   Int32 index;
   UInt32 msg_no;
   UInt32 type;
   UInt32 inform;
   UInt32 timecount;
   ds4302_canChannel*canChannel;
   ds4302_canService *msgService;
    } ds4302_canMsg;
```

Include file

`Ds4302.h`

Members

timestamp This parameter contains the following values:

- For transmit or remote messages: The point in time the last message was successfully sent (given in seconds).
- For receive messages: The point in time the last message was received (given in seconds).

This parameter is updated by the function `ds4302_can_msg_read` if the message was registered using the `inform` parameter `DS4302_CAN_TIMECOUNT_INFO`.

deltatime Time difference in seconds between the old and the new timestamp

This parameter is updated by the function `ds4302_can_msg_read` if the message was registered with the `inform` parameter `DS4302_CAN_TIMECOUNT_INFO`.

Note

If several CAN identifiers are received with a single RX message, the `deltatime` parameter delivers useless values. For this reason, it is recommended to use the `deltatime` parameter only if one CAN identifier is received per registered CAN message.

delaytime Time difference between the update and the sending of a message (for TX, RQTX, and RM messages only). For cyclic sending, the delay time between the update and the sending of a message is used. For acyclic sending, the delay time between the trigger and the successful sending of a message is used. The valid range is 0.0 ... 100.0 seconds.

This parameter is updated by the function `ds4302_can_msg_read` if the message was registered with the `inform` parameter `DS4302_CAN_DELAYCOUNT_INFO`.

processed Processed flag of the message. This parameter is updated by the function `ds4302_can_msg_read`. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_PROCESSED`	The message has been sent/received since the last execution call.
`DS4302_CAN_NOT_PROCESSED`	The message has not been sent/received since the last execution call.

datalen Length of the data in the CAN message in bytes. This parameter is updated by the function `ds4302_can_msg_read` if the message was registered with the `inform` parameter `DS4302_CAN_DATA_INFO`.

data[8] Buffer for CAN message data. This data is updated by the function `ds4302_can_msg_read` if the message was registered with the `inform` parameter `DS4302_CAN_DATA_INFO`.

identifier Identifier of the message. This parameter is provided by the message register functions and is read-only.

format Specifies the message format. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_STD`	11-bit standard format, CAN 2.0A
`DS4302_CAN_EXT`	29-bit extended format, CAN 2.0B

module Address of the registered message. This parameter is provided by the message register functions and is read-only.

queue Communication channel within the range of 0 … 5. This parameter is provided by the message register functions and is read-only.

index Table index already allocated by the previously performed register function. This parameter is provided by the message register functions and is read-only.

msg_no Number of the message. This parameter is provided by the message register functions and is read-only.

type Type of the CAN message. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_TX`	Transmit message registered by `ds4302_can_msg_tx_register`
`DS4302_CAN_RX`	Receive message registered by `ds4302_can_msg_rx_register`
`DS4302_CAN_RM`	Remote message registered by `ds4302_can_msg_rm_register`
`DS4302_CAN_RQTX`	RQTX message registered by `ds4302_can_msg_rqtx_register`
`DS4302_CAN_RQRX`	RQRX message registered by `ds4302_can_msg_rqrx_register`

This parameter is provided by the message register functions and is read-only.

inform Specifies the kind of information returned by the function `ds4302_can_msg_read`. You have to register a message with the appropriate `inform` parameter to get the requested information. You can combine the predefined symbols with the logical operator OR. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_INFO`	Returns no information.
`DS4302_CAN_DATA_INFO`	Updates the data and datalen parameters (needed for receive and request (RQRX) messages).
`DS4302_CAN_MSG_INFO`	Updates the message identifier and the message format for RM, RQ, TX, and RX messages.
`DS4302_CAN_TIMECOUNT_INFO`	Updates the timestamp and the deltatime parameters.
`DS4302_CAN_DELAYCOUNT_INFO`	Updates the delaytime parameter.

Note

If you modify the `inform` parameter after the message was registered, your message data will be corrupted.

This parameter is provided by the message register functions and is read-only.

timecount Internally used parameter. This parameter is read-only.

canChannel Pointer to the used `ds4302_canChannel` structure where the message object is installed. This parameter is read-only. Refer to `ds4302_canChannel` on page 11.

msgService Only used by the message processed functions to read the processed status (sent or received) of a message. This parameter is read-only.

Related topics

References

ds4302_can_msg_read........77
ds4302_can_msg_rm_register........58
ds4302_can_msg_rqrx_register........55
ds4302_can_msg_rqtx_register........50
ds4302_can_msg_rx_register........47
ds4302_can_msg_tx_register........42

Initialization

Introduction

Before you can use a CAN controller, you have to perform an initialization process that resets the slave DSP and sets up the communication channels between master and slave (parameter `queue`).

Where to go from here

Information in this section

ds4302_init

Syntax

```
void ds4302_init(const UInt32 base)
```

Include file

`Ds4302.h`

Purpose

To initialize the DS4302 and to load the firmware from the PROM. This function performs the following actions:

- Resets the slave DSP and the FIFO
- Disables all on-board transceivers and the 120-Ω termination resistor
- Selects the interrupt line 0 for PHS-bus interrupt line expansion
- Enables I/O-error reset.

Parameters

base	Specifies the PHS-bus base address of the DS4302.

Return Value

None.

Messages

The following message is defined:

ID	Type	Message	Description
100	Error	ds4302_init(x,..) DS4302 board at offset X not found!	The board DS4302 is not found at the given address.
106	Error	ds4302_can_communication_init(x,..) slave: not responding	The slave did not finish the initialization of the communication within one second due to a wrong firmware version or a hardware failure. This error may be caused by an active I/O error line (/IOERR) of the PHS-bus due to a wrong initialization order of the I/O boards connected to the PHS-bus. You have to initialize the I/O boards with the appropriate board_init functions and the DS4302 has to be the last board to be initialized.
109	Error	ds4302_can_communication_init(x,..) slave: wrong firmware version	The firmware version of the CAN module is incompatible with the Real-Time Library (RTLib) used.

Example

For examples, refer to:
- Example of Handling Transmit and Receive Messages on page 94
- Example of Handling Request and Remote Messages on page 96
- Example of Using Subinterrupts on page 98

Related topics

References

ds4302_can_channel_init..25
ds4302_can_communication_init..21

ds4302_can_communication_init

Syntax

```
void ds4302_can_communication_init(
      const UInt32 base,
      const UInt32 bufferwarn)
```

Include file

`Ds4302.h`

Purpose

To initialize communication between the master and the slave DS4302.

Description

This function also initializes seven communication channels with fixed queues (0 … 6) for the master-slave communication. The communication channel QUEUE0 has the highest priority. The slave initializes the communication with the master itself and sends an acknowledgment code if the initialization was successful. If the master does not receive this acknowledgment code within one second, the program is aborted. The timer of the slave DSP is reset.

Parameters

base Specifies the PHS-bus base address of the DS4302 board.

bufferwarn Enables the bufferwarn subinterrupt. The subinterrupt handler is installed automatically. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_INT_DISABLE`	The bufferwarn subinterrupt is disabled.
`DS4302_CAN_INT_ENABLE`	The bufferwarn subinterrupt is enabled.

Return value

None

Messages

The following messages are defined:

ID	Type	Message	Description
101	Error	ds4302_can_communication_init(x,..) memory: allocation error on master	Memory allocation error. No free memory on the master.
104	Error	ds4302_can_communication_init(x,..) queue: master to slave overflow	Not enough memory space between the master write pointer and the slave read pointer. The operation will be aborted.
105	Error	ds4302_can_communication_init(x,..) subint: init failed by master	Master subinterrupt initialization failed. There is not enough memory available.
106	Error	ds4302_can_communication_init(x,..) slave: not responding	The slave did not finish the initialization of the communication within one second due to a wrong firmware version or a hardware failure. This error may be caused by an active I/O error line (/IOERR) of the PHS-bus due to a wrong initialization order of the I/O boards connected to the PHS-bus. You have to initialize the I/O boards with the appropriate board_init functions and the DS4302 has to be the last board to be initialized.
107	Error	ds4302_can_communication_init(x,..) slave: memory allocation error	Memory allocation error on the slave. There are too many functions registered.
108	Error	ds4302_can_communication_init(x,..) queue: slave to master overflow	Not enough memory space between the slave write pointer and the master read pointer. The slave tries to write data in a filled queue. To prevent this error deactivate all messages with `ds4302_can_msg_sleep` or `ds4302_can_channel_all_sleep` when registering messages or services.

Example

For examples, refer to:

- Example of Handling Transmit and Receive Messages on page 94
- Example of Handling Request and Remote Messages on page 96
- Example of Using Subinterrupts on page 98

Related topics

References

ds4302_can_channel_all_sleep	33
ds4302_can_channel_init	25
ds4302_can_msg_sleep	75
ds4302_init	20

CAN Channel Handling

Introduction

Provides information on handling CAN interfaces, called *CAN channels*.

Where to go from here

Information in this section

ds4302_can_channel_init

Syntax

```
ds4302_canChannel* ds4302_can_channel_init(
      const UInt32 base,
      const UInt32 channel,
      const UInt32 baudrate,
      const UInt32 mb15_format,
      const Int32 busoff_subinterrupt);
```

Include file

`Ds4302.h`

Purpose

To perform the basic initialization of the specified CAN channel, that is, to reset the CAN controller and set its baud rate.

Note

You have to call the **`ds4302_can_channel_start`** function to complete the CAN channel initialization.

Description

If no error occurs, `ds4302_can_channel_init` returns a pointer to the `ds4302_canChannel` structure.

If an interrupt is to be sent for the bus off state of the CAN controller, you have to specify a subinterrupt number and a subinterrupt handler.

Parameters

base Specifies the PHS-bus base address of the DS4302 board.

channel Specifies the CAN channel within the range 0 … 3

baudrate Specifies the baud rate of the CAN bus within the range 10 kBd … 1 MBd.

mb15_format Specifies the format for mailbox 15. Mailbox 15 is a double-buffered receive unit of the CAN. Use this mailbox for the message type most frequently used in your application. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_STD`	11-bit standard format, CAN 2.0A
`DS4302_CAN_EXT`	29-bit extended format, CAN 2.0B

busoff_subinterrupt Specifies the Subinterrupt number for the bus off state. The valid range is 0 … 30. Use the following predefined symbol to disable the bus off interrupt:

Predefined Symbol	Meaning
`DS4302_CAN_NO_SUBINT`	No interrupt for bus off

Return value

canChannel	Pointer to the `ds4302_canChannel`

Messages

The following messages are defined:

ID	Type	Message	Description
120	Error	ds4302_can_channel_init(x,..) baudrate: illegal	The function **`ds4302_can_channel_init`** is unable to calculate the bit timing parameter for the given baud rate. Use the function **`ds4302_can_channel_init_advanced`** instead.
124	Error	ds4302_can_channel_init(x,..) frequency < DS4302_CAN_MINCLOCK Hz!	The clock frequency of the DS4302 CAN clock generator limited by `DS4302_CAN_MINCLOCK` is too low.
125	Error	ds4302_can_channel_init(x,..) frequency > DS4302_CAN_MAXCLOCK Hz!	The clock frequency of the DS4302 CAN clock generator limited by `DS4302_CAN_MAXCLOCK` is too high.
300	Warning	ds4302_can_channel_init(x,..) frequency < DS4302_CAN_LOWCLOCK: low Performance!	The clock frequency of the CAN clock generator is lower than DS4302_CAN_LOWCLOCK (low performance).

Example

For examples, refer to:

- Example of Handling Transmit and Receive Messages on page 94
- Example of Handling Request and Remote Messages on page 96
- Example of Using Subinterrupts on page 98

Related topics

References

ds4302_can_channel_all_sleep	33
ds4302_can_channel_all_wakeup	34
ds4302_can_channel_BOff_go	35
ds4302_can_channel_BOff_return	36
ds4302_can_channel_init_advanced	27
ds4302_can_channel_set	37
ds4302_can_channel_start	32
ds4302_can_msg_rm_register	58
ds4302_can_msg_rqrx_register	55
ds4302_can_msg_rqtx_register	50
ds4302_can_msg_rx_register	47
ds4302_can_msg_sleep	75
ds4302_can_msg_tx_register	42
ds4302_can_service_register	85
ds4302_canChannel	11

ds4302_can_channel_init_advanced

Syntax

```
ds4302_canChannel* ds4302_can_channel_init_advanced(
      const UInt32 base,
      const UInt32 channel,
      const UInt32 frequency,
      const UInt32 bit_timing0,
      const UInt32 bit_timing1,
      const UInt32 mb15_format,
      const Int32 busoff_subinterrupt);
```

Include file

`Ds4302.h`

Purpose

To perform the initialization of a CAN channel with parameters.

If no error occurs, the function returns a pointer to the `ds4302_canChannel` structure.

Note

You have to call `ds4302_can_channel_start` to complete the CAN channel initialization.

Description

Use the returned handle when calling one of the following functions: `ds4302_can_channel_start`, `ds4302_can_channel_all_sleep`, `ds4302_can_channel_all_wakeup`, `ds4302_can_channel_BOff_go`, `ds4302_can_channel_BOff_return`, `ds4302_can_channel_set`, `ds4302_can_msg_tx_register`, `ds4302_can_msg_rx_register`, `ds4302_can_msg_rqtx_register`, `ds4302_can_msg_rqrx_register`.

If an interrupt should be sent for the bus off state of the CAN controller, you have to specify a subinterrupt number.

The function `ds4302_can_channel_start` completely initializes the CAN controller. All mailbox-independent initializations are done by this function. After the hardware-dependent registers are set, the CAN controller interrupts are disabled.

Parameters

base Specifies the PHS-bus base address of the DS4302 board.

channel Specifies the CAN channel 0 … 3

frequency Specifies the frequency for the CAN clock generator within the range 5 … 16 MHz. If you use a value below 16 MHz the performance will decrease. If you choose a value below 10 MHz a warning message will be generated.

bit_timing0 Specifies the value for the bit timing register 0

bit_timing1 Specifies the value for the bit timing register 1

mb15_format Specifies the format for mailbox 15. Mailbox 15 is a double-buffered receive unit of the CAN. Use this mailbox for the message type most frequently used in your application. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_STD`	11-bit standard format, CAN 2.0A
`DS4302_CAN_EXT`	29-bit extended format, CAN 2.0B

busoff_subinterrupt Specifies the Subinterrupt number for bus off. Valid range is 0 … 30. Use the following predefined symbol to disable the bus off interrupt:

Predefined Symbol	Meaning
`DS4302_CAN_NO_SUBINT`	No interrupt for bus off

Return value

canChannel Specifies the pointer to the `ds4302_canChannel` structure.

Messages

The following messages are defined:

ID	Type	Message	Description
101	Error	ds4302_can_channel_init_advanced (x,..) memory allocation error on master	Memory allocation error. No free memory on the master.
123	Error	ds4302_can_channel_init_advanced (x,..) channel: use range 0..3!	Use a CAN channel within the range of 0 … 3.
124	Error	ds4302_can_channel_init_advanced(x,..) frequency < DS4302_CAN_MINCLOCK Hz!	The clock frequency of the DS4302 CAN clock generator limited by `DS4302_CAN_MINCLOCK` is too low.
125	Error	ds4302_can_channel_init_advanced(x,..) frequency > DS4302_CAN_MAXCLOCK Hz!	The clock frequency of the DS4302 CAN clock generator limited by `DS4302_CAN_MAXCLOCK` is too high.
140	Error	ds4302_can_channel_init_advanced (x,..) format: wrong format	Only the `DS4302_CAN_STD` and `DS4302_CAN_EXT` symbols are allowed for the mb15_format parameter.

ID	Type	Message	Description
141	Error	ds4302_can_channel_init_advanced (x,..) subint: use range 0..30!	The subinterrupt number must be within the range of 0 … 30.
300	Warning	ds4302_can_channel_init_advanced(x,..) frequency < DS4302_CAN_LOWCLOCK : low Performance!	Warning message from the `ds4302_can_channel_init` and `ds4302_can_channel_init_advanced` functions. The clock frequency of the CAN clock generator is lower than DS4302_CAN_LOWCLOCK (low performance).

Example

```
ds4302_canChannel* CH;
CH = ds4302_can_channel_init_advanced(
   DS4302_1_BASE,      /* PHS-bus base address  */
   0,                  /* channel 0 */
   0x80,               /* BTR0      */
   0x6F,               /* BTR1*     */
   DS4302_CAN_STD,     /* use mailbox 15 to receive only    */
                       /* CAN messages with standard format */
   DS4302_CAN_NO_SUBINT /* generate no subinterrupt when  */
                       /* the CAN controller goes in the    */
                       /* bus off state                     */
);
```

Related topics

References

ds4302_can_channel_transceiver

Syntax

```
void ds4302_can_channel_transceiver(
      const ds4302_canChannel* canCh,
      const UInt32 transceiver,
      const UInt32 termination);
```

Include file

`Ds4302.h`

Purpose

To select the CAN transceiver and the CAN termination for the CAN channel determined by the pointer ds4302_canChannel on page 11.

Use the returned handle from the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to call this function.

Parameters

canCh Specifies the pointer to the `ds4302_canChannel` structure.

transceiver Specifies the CAN transceiver; the following symbols are predefined:

Predefined Symbols	Selected Transceiver	Termination Resistors
DS4302_CAN_NO_TRANSCEIVER	No transceiver selected	Using the termination parameter you can enable or disable the 120-Ω termination resistor in order to ensure a defined bus termination.
DS4302_CAN_ISO11898_TRANSCEIVER	ISO 11898 transceiver PCA82C251	Using the termination parameter you can enable or disable the 120-Ω termination resistor.
DS4302_CAN_RS485_TRANSCEIVER	Modified RS485 transceiver	Using the termination parameter you can enable or disable the 120-Ω termination resistor.
DS4302_CAN_C252_TRANSCEIVER	Fault-tolerant transceiver PCA82C252	Using the termination parameter you can choose between an 1.6-kΩ or a 10-kΩ termination resistor.

termination Controls the on-board bus termination; the following symbols are predefined:

Predefined Symbols	Selected Transceiver	Meaning
DS4302_CAN_TERMINATION_ON DS4302_CAN_TERMINATION_OFF	DS4302_CAN_NO_TRANSCEIVER, DS4302_CAN_ISO11898_TRANSCEIVER, DS4302_CAN_RS485_TRANSCEIVER	Enables or disables the 120-Ω termination resistor between the CAN-H and CAN-L line of the CAN bus. If termination is set to DS4302_CAN_TERMINATION_ON the 120-Ω termination resistor is enabled. Termination set to DS4302_CAN_TERMINATION_OFF will disable the termination resistor.
DS4302_CAN_C252_LOW_RESISTOR DS4302_CAN_C252_HIGH_RESISTOR	DS4302_CAN_C252_TRANSCEIVER	The fault-tolerant transceiver PCA82C252 has its own termination resistors. The DS4302 allows you to select 10-kΩ termination resistors or 1.6-kΩ termination resistors between CAN-L and RTL and CAN-H and RTH. If termination is set to DS4302_CAN_C252_LOW_RESISTOR the 1.6-kΩ termination resistor is selected. If termination is set to DS4302_CAN_C252_HIGH_RESISTOR the 10-kΩ termination resistor is selected.

Return value

None

Messages

The following error and warning messages are defined:

ID	Type	Message	Description
104	Error	ds4302_can_channel_init(x,..) baudrate: illegal	The function `ds4302_can_channel_init` is unable to calculate the bit timing parameter for the given baud rate. Use the function `ds4302_can_channel_init_advanced` instead.
106	Error	ds4302_can_channel_transceiver(x,..) slave: not responding	The slave did not finish the initialization of the communication within one second due to a wrong firmware version or a hardware failure. This error may be caused by an active I/O error line (/IOERR) of the PHS-bus due to a wrong initialization order of the I/O boards connected to the PHS-bus. You have to initialize the I/O boards with the appropriate board_init functions and the DS4302 has to be the last board to be initialized.
107	Error	ds4302_can_channel_transceiver(x,..) slave: memory allocation error	Memory allocation error on slave. There are too many functions registered.
108	Error	ds4302_can_channel_transceiver(x,..) queue: slave to master overflow	Not enough memory space between the slave write pointer and the master read pointer. The slave tries to write data in a filled queue. To prevent this error deactivate all messages with `ds4302_can_msg_sleep` or `ds4302_can_channel_all_sleep` during registering messages or services.
120	Error	ds4302_can_channel_transceiver(x,..) queue: master to slave overflow	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted.
126	Error	ds4302_can_channel_transceiver(x,..) baudrate: too high for the transceiver	The given baud rate exceeds the maximum value for the specified transceiver.

Example

For examples, refer to:
- Example of Handling Transmit and Receive Messages on page 94
- Example of Handling Request and Remote Messages on page 96

Related topics

References

ds4302_can_channel_all_sleep	33
ds4302_can_channel_init	25
ds4302_can_channel_init_advanced	27
ds4302_can_communication_init	21
ds4302_can_msg_sleep	75

ds4302_can_channel_start

Syntax

```
void ds4302_can_channel_start(
      const ds4302_canChannel* canCh,
      const UInt32 status_int);
```

Include file

`Ds4302.h`

Purpose

To complete the initialization and start the CAN channel referenced by the canCh pointer.

Description

The CAN channel will change to the bus on state and the DS4302 slave interrupts will be enabled. Use the returned handle from the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to call this function.

Parameters

canCh Specifies the pointer to the `ds4302_canChannel` structure.

status_int Enables the status change interrupt; the following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_INT_DISABLE`	No status interrupt will be generated.
`DS4302_CAN_INT_ENABLE`	A status change interrupt can be generated when a CAN bus event is detected in the Status Register. A status change interrupt occurs on each successful reception or transmission on the CAN bus, regardless of whether the DS4302 slave has configured a message object to receive that particular message identifier. This interrupt is useful to detect bus errors caused by physical layer issues, such as noise. In most applications, it is recommended to not set this bit. Because this interrupt occurs for each message, the DS4302 would be unnecessarily burdened.

Return value

None

Messages

The following messages are defined:

ID	Type	Message	Description
104	Error	ds4302_can_channel_start(x,..) queue: master to slave overflow	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted.
126	Error	ds4302_can_channel_start(x,..) baudrate: too high for the transceiver	The given baud rate exceeds the maximum value for the specified transceiver.

Example

For examples, refer to:

- Example of Handling Transmit and Receive Messages on page 94
- Example of Handling Request and Remote Messages on page 96
- Example of Using Subinterrupts on page 98

Related topics

References

ds4302_can_channel_all_sleep

Syntax

```
Int32 ds4302_can_channel_all_sleep(
      const ds4302_canChannel* canCh);
```

Include file

`Ds4302.h`

Purpose

To stop the transmission of all previously registered transmit, request transmission, and remote messages and the data transfer from all registered messages to the master processor board.

Description

The messages are deactivated and set to sleep mode until they are reactivated by `ds4302_can_channel_all_wakeup`.

Use the returned handle from the `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` function to call this function.

Parameters

canCh	Specifies the pointer to the `ds4302_canChannel` structure.

Return value

This function returns the error code; the following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	An overflow of the master to slave communication buffer occurred. Repeat the function until it returns DS4302_CAN_NO_ERROR.

Example

```
ds4302_can_channel_all_sleep(canCh);
```

Related topics

References

ds4302_can_channel_all_wakeup.......... 34
ds4302_can_channel_init.......... 25
ds4302_can_channel_init_advanced.......... 27

ds4302_can_channel_all_wakeup

Syntax

```
Int32 ds4302_can_channel_all_wakeup(
      const ds4302_canChannel* canCh);
```

Include file

`Ds4302.h`

Purpose

To reactivate all messages that were deactivated by calling the functions `ds4302_can_channel_all_sleep` and `ds4302_can_msg_sleep`.

Description

Use the returned handle from the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to call this function.

Parameters

canCh	Specifies the pointer to the `ds4302_canChannel` structure.

Return value This function returns the error code; the following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function has been performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	The communication buffer occurred. Repeat the function until it returns DS4302_CAN_NO_ERROR.

Example

```
ds4302_can_channel_all_wakeup(canCh);
```

Related topics

References

ds4302_can_channel_BOff_go

Syntax

```
Int32 ds4302_can_channel_BOff_go(
      const ds4302_canChannel* canCh);
```

Include file `Ds4302.h`

Purpose To set the CAN channel to the bus off state. All bus operations performed by the CAN channel are canceled.

Description Use the returned handle from the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to call this function.

Parameters

canCh	Specifies the pointer to the `ds4302_canChannel` structure.

Return value

This function returns the error code; the following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.

Example

```
ds4302_can_channel_BOff_go(canCh);
```

Related topics

References

ds4302_can_channel_BOff_return

Syntax

```
Int32 ds4302_can_channel_BOff_return(
      const ds4302_canChannel* canCh);
```

Include file

`Ds4302.h`

Purpose

To reset the slave DS4302 CAN channel from the bus off state.

Use the returned handle from the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to call this function.

Parameters

canCh	Specifies the pointer to the `ds4302_canChannel` structure.

Return value

This function returns the error code; the following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	An overflow of the master to slave communication buffer occurred. Repeat the function until it returns DS4302_CAN_NO_ERROR.

Example

```
ds4302_can_channel_BOff_return(canCh);
```

Related topics

References

ds4302_can_channel_set

Syntax

```
Int32 ds4302_can_channel_set(
      const ds4302_canChannel* canCh,
      const UInt32 mask_type,
      const UInt32 mask_value);
```

Include file

`Ds4302.h`

Purpose

To set a mask value or attribute for the specified CAN channel. Use this function to write the value to the specified CAN controller memory area. Use the returned handle from the function `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to call this function.

Parameters

canCh Specifies the pointer to the `ds4302_canChannel` structure.

mask_type Specifies the mask type. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_CHANNEL_SET_MASK15`	Sets the Message 15 Mask Register.
`DS4302_CAN_CHANNEL_SET_ARBMASK15`	Sets the Arbitration Register for mailbox 15.
`DS4302_CAN_CHANNEL_SET_C252_SUBINT`	Connects the PCA82C252 error interrupt to a subinterrupt. The subinterrupt will be generated when one of the following CAN bus events occurs: CAN-H wire interrupted, ▪ CAN-L wire interrupted, ▪ CAN-H shorted to battery, ▪ CAN-L shorted to ground, ▪ CAN-L shorted to battery, ▪ CAN-H shorted to ground, ▪ CAN-L mutually shorted to CAN-H

Predefined Symbol	Meaning
DS4302_CAN_CHANNEL_SET_BUSCFG	*Only for piggy-back modules.* Sets the Bus Configuration Register of the CAN controller. Call this function before `ds4302_can_channel_start`.
DS4302_CAN_CHANNEL_SET_P2CONF	*Only for piggy-back modules.* Sets the PORT 2 Register of the CAN controller. Call this function before `ds4302_can_channel_start`.
DS4302_CAN_CHANNEL_SET_P2OUT	*Only for piggy-back modules.* Sets the PORT 2 Register of the CAN controller.
DS4302_CAN_CHANNEL_SET_TRANSCEIVER	Set the transceiver to be used.
DS4302_CAN_CHANNEL_SET_TERMINATION	Set the termination resistor for the channel.
DS4302_CAN_CHANNEL_SET_BAUDRATE	Sets the baud rate of the selected channel during run time.

mask_value Specifies the value of the mask to be written: 0 = "don't care", 1 = "must match".

mask_type	mask_value
DS4302_CAN_CHANNEL_SET_ARBMASK15	Arbitration field for mailbox 15. Bit0 (on the right in mask_value) corresponds to bit ID0 in the arbitration field, Bit1 = ID1, …, Bit28 = ID28.
DS4302_CAN_CHANNEL_SET_MASK15	For mailbox 15 only: Message 15 Mask Register. Bit0 (on the right in mask_value) corresponds to bit ID0 in the arbitration field, Bit1 = ID1, …, Bit28 = ID28.
DS4302_CAN_CHANNEL_SET_C252_SUBINT	Number of the subinterrupt within the range 0 … 30.
DS4302_CAN_CHANNEL_SET_BUSCFG	Bit0 (on the right in mask_value) corresponds to the LSB of the Bus Configuration Register.
DS4302_CAN_CHANNEL_SET_P2CONF	Bit0 (on the right in mask_value) corresponds to the LSB of the P2CONF Register.
DS4302_CAN_CHANNEL_SET_P2OUT	Bit0 (on the right in mask_value) corresponds to the LSB of the P2OUT Register.
DS4302_CAN_CHANNEL_SET_TRANSCEIVER	Use one of the following symbols to set the transceiver: `DS4302_CHANNEL_RS485_TRANSCEIVER`, `DS4302_CHANNEL_ISO11898_TRANSCEIVER`, `DS4302_CHANNEL_C252_TRANSCEIVER`, or `DS4302_CHANNEL_NO_TRANSCEIVER`
DS4302_CAN_CHANNEL_SET_TERMINATION	Use one of the following symbols to set the termination resistor: `DS4302_CHANNEL_TERMINATION_ON`, `DS4302_CHANNEL_TERMINATION_OFF`, `DS4302_CHANNEL_C252_LOW_RESISTOR`, or `DS4302_CHANNEL_C252_HIGH_RESISTOR`
DS4302_CAN_CHANNEL_SET_BAUDRATE	Sets the baud rate (in baud). Valid range: 10,000 … 1,000,000. Some baud rates in the allowed range cannot be met. If the actual baud rate differs from the one you specify by more than 1%, the function outputs a warning with the actual baud rate settings. Using CAN service functions, you can check the current bus status and whether the new baud rate parameters were changed correctly. Refer to CAN Service Functions on page 85.

For further information on the registers, refer to the manual of the CAN controller.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	An overflow of the master to slave communication buffer occurred. Repeat the function until it returns DS4302_CAN_NO_ERROR.
`DS4302_CAN_BAUDRATE_L_ERROR`	The baud rate is too low. The operation is aborted.
`DS4302_CAN_BAUDRATE_H_ERROR`	The baud rate is too high. The operation is aborted.
`DS4302_CAN_BAUDRATE_SET_BAUDR_ERROR`	Error during the calculation of the new bit timing parameters. The operation is aborted.

Messages

The following messages are defined:

Type	Message	Description
Warning	DS4302 (0x y,...): baudrate on channel ... doesn't match the desired baudrate. New baudrate = ... bit/s (y: board index)	The actual baud rate differs from the one you specified by more than 1%.

Example

```
ds4302_can_channel_set(
        canCh,
        DS4302_CAN_CHANNEL_SET_MASK15,
        0xFFFFFFFE);
/* Set the Lowest bit of the Message 15 Mask Register */
/* to “don't care" */
```

Related topics

References

ds4302_can_channel_txqueue_clear

Syntax

```
Int32 ds4302_can_channel_txqueue_clear(
      const ds4302_canChannel* canCh);
```

Include file	`Ds4302.h`
Purpose	To clear the content of the transmit queues of the selected CAN channel.

Description

The function clears the content of the transmit queues of the selected CAN channel.

Note

When you use this function, all the TX messages in the transmit queues are deleted.

Parameters

canCh	Specifies the pointer to the `ds4302_canChannel` structure.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	An overflow of the master to slave communication buffer occurred. Repeat the function until it returns DS4302_CAN_NO_ERROR.

CAN Message Handling

Introduction

To handle different kinds of CAN messages.

Where to go from here

Information in this section

ds4302_can_msg_tx_register

Syntax

```
ds4302_canMsg* ds4302_can_msg_tx_register(
      const ds4302_canChannel* canCh,
      const Int32 queue,
      const UInt32 identifier,
      const UInt32 format,
      const UInt32 inform,
      const Int32 subinterrupt,
      const Float32 start_time,
      const Float32 repetition_time,
      const Float32 timeout);
```

Include file `Ds4302.h`

Purpose

To register a transmit message on the slave DS4302.

Description

If no error occurs, the function returns a pointer to the `ds4302_canMsg` structure.

Use the returned handle when calling one of the following functions:

- `ds4302_can_msg_write` to write new data to the message
- `ds4302_can_msg_read` to read the returned timestamps
- `ds4302_can_msg_send` to send the message with new data
- `ds4302_can_msg_trigger` to send the message
- `ds4302_can_msg_sleep` to deactivate the message
- `ds4302_can_msg_wakeup` to reactivate the message
- `ds4302_can_msg_clear` to clear the message object data
- `ds4302_can_msg_processed_register` to register the processed function
- `ds4302_can_msg_processed_request` to request the processed function
- `ds4302_can_msg_processed_read` to read the returned data

Note

You must call `ds4302_can_msg_write` to make the message valid for the CAN channel.

Parameters

canCh Specifies the pointer to the `ds4302_canChannel` structure.

queue Specifies the communication channel within the range 0 … 5.

identifier Specifies the identifier of the message.

format Specifies the message format. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_STD`	11-bit standard format, CAN 2.0A
`DS4302_CAN_EXT`	29-bit extended format, CAN 2.0B

inform Specifies the information values to be updated. You can combine the predefined symbols with the logical OR operator. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_INFO`	Returns no information.
`DS4302_CAN_MSG_INFO`	Updates the message identifier and the message format.

Predefined Symbol	Meaning
`DS4302_CAN_TIMECOUNT_INFO`	Updates the timestamp and the `deltatime` parameters.
`DS4302_CAN_DELAYCOUNT_INFO`	Updates the `delaytime` parameter.

subinterrupt Specifies the subinterrupt number for a received message. The valid range is 0 … 30.

Note

The interrupt number 31 is occupied by the buffer overflow warning interrupt (DS4302_CAN_SUBINT_BUFFERWARN). Do not use this number for any other interrupt.

Use the following predefined symbol to select no interrupt for the TX message:

Predefined Symbol	Meaning
`DS4302_CAN_NO_SUBINT`	No interrupt for the TX message

start_time Specifies the point in time of the first sending after timer start. Enter the value in seconds within the range 0 … 420.

Note

For board revision DS4302-05, the maximum time value is 40 seconds. You can inspect the revision of your DS4302 using dSPACE experiment software.

repetition_time Specifies the time interval for repeating the message automatically. Enter the value in seconds within the range 0 ... 100.

Note

For board revision DS4302-05, the maximum time value is 40 seconds. You can inspect the revision of your DS4302 using dSPACE experiment software.

Use the following predefined symbol to define a message sent only once with `ds4302_can_msg_trigger`:

Predefined Symbol	Meaning
`DS4302_CAN_TRIGGER_MSG`	Calls `ds4302_can_msg_trigger` to send the message.

timeout The message will occupy the mailbox only up to this point in time. When the threshold is exceeded, the message is released from the mailbox. Enter the value in seconds within the range 0 … 100.

Note

For board revision DS4302-05, the maximum time value is 40 seconds. You can inspect the revision of your DS4302 using dSPACE experiment software.

Use the following predefined symbol to calculate the timeout value internally:

Predefined Symbol	Meaning
`DS4302_CAN_TIMEOUT_NORMAL`	The timeout value is calculated internally when registering the message. This timeout value works in most cases.

Return value

canMsg Specifies the pointer to the `ds4302_canMsg` structure.

Messages

The following messages are defined:

ID	Type	Message	Description
101	Error	ds4302_can_msg_tx_register(x,..) memory allocation error on master	Memory allocation error. No free memory on the master.
102	Error	ds4302_can_msg_tx_register(x,..) queue: Illegal communication queue.	There is no communication channel with this queue number.
103	Error	ds4302_can_msg_tx_register(x,..) index: illegal function index	The index does not exist in the command table and is not equal to DS4302_CAN_AUTO_INDEX.
104	Error	ds4302_can_msg_tx_register(x,..) queue: master to slave overflow	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted.
106	Error	ds4302_can_msg_tx_register(x,..) slave: not responding	The slave did not finish the initialization within one second. This error may be caused by an active I/O error line (/IOERR) of the PHS-bus due to a wrong initialization order of the I/O boards connected to the PHS-bus. You have to initialize the I/O boards with the appropriate board_init functions and the DS4302 has to be the last board to be initialized.
107	Error	ds4302_can_msg_tx_register(x,..) slave: memory allocation error	Memory allocation error on the slave. There are too many functions registered.
108	Error	ds4302_can_msg_tx_register(x,..) queue: slave to master overflow	Not enough memory space between the slave write pointer and the master read pointer. The slave tries to write data to a filled queue. To prevent this error, deactivate all messages with `ds4302_can_msg_sleep` or `ds4302_can_channel_all_sleep` when registering messages or services.
140	Error	ds4302_can_msg_tx_register(x,..) format: wrong format	Only the symbols `DS4302_CAN_STD` and `DS4302_CAN_EXT` are allowed for the parameter `format`.
141	Error	(x,..) subint: use range 0..14! ds4302_can_msg_tx_register(x,..) subint: use range 0..30!	The subinterrupt number must be within the range 0 … 30.

ID	Type	Message	Description
142	Error	ds4302_can_msg_tx_register(x,..) subint: used for busoff!	The specified subinterrupt number is used for the CAN channel bus off subinterrupt.
143	Error	ds4302_can_msg_tx_register(x,..) id: Illegal id or id conflict	The CAN controller does not install the identifier given in the program. For further information, refer to `ds4302_canService` on page 13.
144	Error	ds4302_can_msg_tx_register(x,..) Too much messages (max. 200)!	The total number of registered messages is limited to 200. The program is aborted.
145	Error	ds4302_can_msg_tx_register(x,..) starttime: too high (max. 420s)!	The `start_time` value must not be higher than 420 seconds. Exceeding this value causes an error and the program is aborted.
146	Error	ds4302_can_msg_tx_register(x,..) rep. time: too high (max. 100s)!	The `repetition_time` value must not be higher than 100 seconds. Exceeding this value causes an error and the program is aborted.
147	Error	ds4302_can_msg_tx_register(x,..) rep. time: too low !	Must be at least CAN_FRAME_TIME. A lower value causes an error and the program is aborted. Note that CAN_FRAME_TIME = (136 / Baud rate).
148	Error	ds4302_can_msg_tx_register(x,..) timeout: too high (max. 100s)!	The `timeout` value must not be higher than 100 seconds. Exceeding this value causes an error and the program is aborted.
149	Error	ds4302_can_msg_tx_register(x,..) timeout: too low !	The `timeout` value has to be at least 3 · CAN_FRAME_TIME. A lower value causes an error and the program is aborted. Note that CAN_FRAME_TIME = (136 / Baud rate).
152	Error	ds4302_can_msg_tx_register(x,..) canCh: the CAN channel wasn't initialized	This message is displayed if: ▪ You try to register a CAN message on an uninitialized CAN channel. ▪ You try to register a CAN service on an uninitialized CAN channel. Use `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to initialize the CAN channel.

Example

For examples of how to use this function, refer to Example of Handling Transmit and Receive Messages on page 94 and Example of Using Subinterrupts on page 98.

Related topics

References

ds4302_can_channel_all_sleep	33
ds4302_can_channel_init	25
ds4302_can_channel_init_advanced	27
ds4302_can_msg_clear	80
ds4302_can_msg_processed_read	83
ds4302_can_msg_processed_register	81
ds4302_can_msg_processed_request	82
ds4302_can_msg_read	77
ds4302_can_msg_send	67
ds4302_can_msg_sleep	75
ds4302_can_msg_trigger	79
ds4302_can_msg_wakeup	76
ds4302_can_msg_write	65
ds4302_canMsg	16

ds4302_can_msg_rx_register

Syntax

```
ds4302_canMsg* ds4302_can_msg_rx_register(
      const ds4302_canChannel* canCh,
      const Int32 queue,
      const UInt32 identifier,
      const UInt32 format,
      const UInt32 inform,
      const Int32 subinterrupt);
```

Include file

`Ds4302.h`

Purpose

To register a receive message on the slave DS4302.

Description

If no error occurs, `ds4302_can_msg_rx_register` returns a pointer to the `ds4302_canMsg` structure.

Use the returned handle when calling one of the following functions:

- `ds4302_can_msg_read` to read the returned data and timestamps
- `ds4302_can_msg_sleep` to deactivate the message
- `ds4302_can_msg_wakeup` to reactivate the message
- `ds4302_can_msg_clear` to clear the message data
- `ds4302_can_msg_processed_register` to register the processed function
- `ds4302_can_msg_processed_request` to request the processed function
- `ds4302_can_msg_processed_read` to read the returned data

Parameters

canCh Specifies the pointer to the `ds4302_canChannel` structure.

queue Specifies the communication channel within the range 0 … 5.

identifier Specifies the identifier of the message.

format Specifies the message format. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_STD`	11-bit standard format, CAN 2.0A
`DS4302_CAN_EXT`	29-bit extended format, CAN 2.0B

inform Specifies the information values to be updated. You can combine the predefined symbols with the logical OR operator. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_INFO`	Returns no information.
`DS4302_CAN_DATA_INFO`	Updates the data and datalen parameters (needed for receive and request (RQRX) messages).
`DS4302_CAN_MSG_INFO`	Updates the message identifier and the message format.
`DS4302_CAN_TIMECOUNT_INFO`	Updates the timestamp and deltatime parameters.

subinterrupt Specifies the subinterrupt number for a received message. The valid range is 0 … 30.

Note

The interrupt number 31 is occupied by the buffer overflow warning interrupt (DS4302_CAN_SUBINT_BUFFERWARN). Do not use this number for any other interrupt.

Use the following predefined symbol to select no interrupt for the receive message:

Predefined Symbol	Meaning
`DS4302_CAN_NO_SUBINT`	No interrupt for the receive message

Return value

canMsg This function returns the pointer to the `ds4302_canMsg` structure.

Messages

The following messages are defined:

ID	Type	Message	Description
101	Error	ds4302_can_msg_rx_register(x,..) memory allocation error on master	Memory allocation error. No free memory on the master.
102	Error	ds4302_can_msg_rx_register(x,..) queue: Illegal communication queue.	There is no communication channel with this queue number.

ID	Type	Message	Description
103	Error	ds4302_can_msg_rx_register(x,..) index: illegal function index	The index does not exist in the command table and is not equal to DS4302_CAN_AUTO_INDEX.
104	Error	ds4302_can_msg_rx_register(x,..) queue: master to slave overflow	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted.
106	Error	ds4302_can_msg_rx_register(x,..) slave: not responding	The slave did not finish the initialization of the communication within one second. This error may be caused by an active I/O error line (/IOERR) of the PHS-bus due to a wrong initialization order of the I/O boards connected to the PHS-bus. You have to initialize the I/O boards with the appropriate board_init functions and the DS4302 has to be the last board to be initialized.
107	Error	ds4302_can_msg_rx_register(x,..) slave: memory allocation error	Memory allocation error on the slave. There are too many functions registered.
108	Error	ds4302_can_msg_rx_register(x,..) queue: slave to master overflow	Not enough memory space between the slave write pointer and the master read pointer. The slave tries to write data to a filled queue. To prevent this error, deactivate all messages with `ds4302_can_msg_sleep` or `ds4302_can_channel_all_sleep` when registering messages or services.
140	Error	ds4302_can_msg_rx_register(x,..) format: wrong format	Only the symbols `DS4302_CAN_STD` and `DS4302_CAN_EXT` are allowed for the parameter `format`.
141	Error	ds4302_can_msg_rx_register(x,..) subint: use range 0..30 !	The subinterrupt number must be within the range 0 … 30.
142	Error	ds4302_can_msg_rx_register(x,..) subint: used for busoff!	The specified subinterrupt number is used for the CAN channel bus off subinterrupt.
143	Error	ds4302_can_msg_rx_register(x,..) id: Illegal id or id conflict	The CAN controller does not install the identifier given in the program. For further information, see `ds4302_canService` on page 13.
144	Error	ds4302_can_msg_rx_register(x,..): Too much messages (max. 200)!	The total number of registered messages is limited to 200. The program is aborted.
152	Error	ds4302_can_msg_rx_register(x,..) canCh: the CAN channel wasn't initialized	This message is displayed if: ▪ You try to register a CAN message on an uninitialized CAN channel. ▪ You try to register a CAN service on an uninitialized CAN channel. Use `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to initialize the CAN channel.

Example

For examples of how to use this function, refer to Example of Handling Transmit and Receive Messages on page 94 and Example of Using Subinterrupts on page 98.

```
ds4302_canMsg* rxMsg = ds4302_can_msg_rx_register(
         canCh,
         0,
         0x123,
         DS4302_CAN_STD,
         DS4302_CAN_DATA_INFO,
         DS4302_CAN_NO_SUBINT);
```

Related topics

References

ds4302_can_msg_rqtx_register

Syntax

```
ds4302_canMsg* ds4302_can_msg_rqtx_register(
      const ds4302_canChannel* canCh,
      const Int32 queue,
      const UInt32 identifier,
      const UInt32 format,
      const UInt32 inform,
      const Int32 subinterrupt,
      const Float32 start_time,
      const Float32 repetition_time,
      const Float32 timeout);
```

Include file

`Ds4302.h`

Purpose

To register a request transmission (RQTX) message on the slave DS4302.

Description

Use this function to register a request message. Use the function `ds4302_can_msg_rqtx_register` to register a function that receives the requested data. If no error occurs, `ds4302_can_msg_rqtx_register` returns a pointer to the `ds4302_canMsg` structure.

Use the returned handle when calling one of the following functions:

Function	Description
`ds4302_can_msg_rqtx_activate`	to activate the message
`ds4302_can_msg_read`	To read the returned time stamps.
`ds4302_can_msg_sleep`	To deactivate the message.
`ds4302_can_msg_wakeup`	To reactivate the message.
`ds4302_can_msg_trigger`	To send the request message.
`ds4302_can_msg_clear`	To clear the message object data.
`ds4302_can_msg_processed_register`	To register the processed function.
`ds4302_can_msg_processed_request`	To request the processed function.
`ds4302_can_msg_processed_read`	To read the returned data.

Note

You must call `ds4302_can_msg_rqtx_activate` to make the message valid for the CAN channel.

Parameters

canCh Specifies the pointer to the `ds4302_canChannel` structure.

queue Specifies the communication channel within the range 0 … 5.

identifier Specifies the identifier of the message.

format Specifies the message format. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_STD`	11-bit standard format, CAN 2.0A
`DS4302_CAN_EXT`	29-bit extended format, CAN 2.0B

inform Specifies the information values to be updated. You can combine the predefined symbols with the logical OR operator; the following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_INFO`	Returns no information.
`DS4302_CAN_MSG_INFO`	Updates the message identifier and the message format.

Predefined Symbol	Meaning
`DS4302_CAN_TIMECOUNT_INFO`	Updates the timestamp and deltatime parameters.
`DS4302_CAN_DELAYCOUNT_INFO`	Updates the delaytime parameter.

subinterrupt Specifies the subinterrupt number for a received message. The valid range is 0 … 30.

Note

The interrupt number 31 is occupied by the buffer overflow warning interrupt (DS4302_CAN_SUBINT_BUFFERWARN). Do not use this number for any other interrupt.

Use the following predefined symbol to select no interrupt for the RQTX message:

Predefined Symbol	Meaning
`DS4302_CAN_NO_SUBINT`	No interrupt for the RQTX messages.

start_time Specifies the point in time of the first sending after timer start. Enter the value in seconds within the range 0 … 420.

Note

For board revision DS4302-05, the maximum time value is 40 seconds. You can inspect the revision of your DS4302 using dSPACE experiment software.

repetition_time Specifies the time interval for repeating the message automatically. Enter the value in seconds within the range 0 ... 100.

Note

For board revision DS4302-05, the maximum time value is 40 seconds. You can inspect the revision of your DS4302 using dSPACE experiment software.

Use the following predefined symbol to define a message sent only once with `ds4302_can_msg_trigger`:

Predefined Symbol	Meaning
`DS4302_CAN_TRIGGER_MSG`	Calls `ds4302_can_msg_trigger` to send the message.

timeout The message will occupy the mailbox only up to this point in time. When the threshold is exceeded, the message is released from the mailbox. Enter the value in seconds within the range 0 … 100.

Note

For board revision DS4302-05, the maximum time value is 40 seconds. You can inspect the revision of your DS4302 using dSPACE experiment software.

Use the following predefined symbol to calculate the timeout value internally:

Predefined Symbol	Meaning
`DS4302_CAN_TIMEOUT_NORMAL`	The timeout value is calculated internally when registering the message. This timeout value works in most cases.

Return value

canMsg This function returns the pointer to the ds4302_canMsg structure.

Messages

The following messages are defined:

ID	Type	Message	Description
101	Error	ds4302_can_msg_rqtx_register(x,..) memory allocation error on master	Memory allocation error. No free memory on the master.
102	Error	ds4302_can_msg_rqtx_register(x,..) queue: Illegal communication queue	There is no communication channel with this queue number.
103	Error	ds4302_can_msg_rqtx_register(x,..) index: illegal function index	The index does not exist in the command table and is not equal to DS4302_CAN_AUTO_INDEX.
104	Error	ds4302_can_msg_rqtx_register(x,..) queue: master to slave overflow	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted.
106	Error	ds4302_can_msg_rqtx_register(x,..) slave: not responding	The slave did not finish the initialization of the communication within one second. This error may be caused by an active I/O error line (/IOERR) of the PHS-bus due to a wrong initialization order of the I/O boards connected to the PHS-bus. You have to initialize the I/O boards with the appropriate board_init functions and the DS4302 has to be the last board to be initialized.
107	Error	ds4302_can_msg_rqtx_register(x,..) slave: memory allocation error	Memory allocation error on the slave. There are too many functions registered.
108	Error	ds4302_can_msg_rqtx_register(x,..) queue: slave to master overflow	Not enough memory space between the slave write pointer and the master read pointer. The slave tries to write data to a filled queue. To prevent this error, deactivate all messages with `ds4302_can_msg_sleep` or

ID	Type	Message	Description
			`ds4302_can_channel_all_sleep` when registering messages or services.
140	Error	ds4302_can_msg_rqtx_register(x,..) format: wrong format	Only the symbols DS4302_CAN_STD and DS4302_CAN_EXT are allowed for the parameter format.
141	Error	(x,..) subint: use range 0..14 ! ds4302_can_msg_rqtx_register(x,..) subint: use range 0..30 !	The subinterrupt number must be within the range 0 … 30.
142	Error	ds4302_can_msg_rqtx_register(x,..) subint: used for busoff!	The specified subinterrupt number is used for the CAN channel bus off subinterrupt.
143	Error	can_tp1_msg_rqtx_registerds4302_can_msg_rqtx_register(x,..) id: Illegal id or id conflict	The CAN controller does not install the identifier specified in the program. For further information, refer to DS4302_CAN_SERVICE_MAILBOX_ERR.
144	Error	(x,..): Too much messages (max.100)! ds4302_can_msg_rqtx_register(x,..): Too much messages (max.200)!	The total number of registered messages is limited to 200. The program is aborted.
152	Error	ds4302_can_msg_rqtx_register(x,..) canCh: the CAN channel wasn't initialized	This message is displayed if: ▪ You try to register a CAN message on an uninitialized CAN channel. ▪ You try to register a CAN service on an uninitialized CAN channel. Use `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to initialize the CAN channel.

Example

For examples of how to use this function, refer to Example of Handling Request and Remote Messages on page 96.

```
ds4302_canMsg* rqtxMsg = ds4302_can_msg_rqtx_register(
         canCh,
         0,
         0x123,
         DS4302_CAN_STD,
         DS4302_CAN_TIMECOUNT_INFO,
         DS4302_CAN_NO_SUBINT,
         1.5,
         0.3,
         DS4302_CAN_TIMEOUT_NORMAL);
```

Related topics

References

ds4302_can_channel_all_sleep	33
ds4302_can_msg_clear	80
ds4302_can_msg_processed_read	83
ds4302_can_msg_processed_register	81
ds4302_can_msg_processed_request	82
ds4302_can_msg_read	77
ds4302_can_msg_rqtx_activate	64
ds4302_can_msg_sleep	75
ds4302_can_msg_trigger	79
ds4302_can_msg_wakeup	76

ds4302_can_msg_rqrx_register

Syntax

```
ds4302_canMsg* ds4302_can_msg_rqrx_register(
      const ds4302_canMsg* rqtxMsg,
      const UInt32 inform,
      const Int32 subinterrupt);
```

Include file

`Ds4302.h`

Purpose

To register an RQRX message on the slave DS4302.

Description

Use this message to receive the data requested with an RQTX message. If no error occurs, `ds4302_can_msg_rqrx_register` returns a pointer to the `ds4302_canMsg` structure.

Use the returned handle when calling one of the following functions:

- `ds4302_can_msg_read` to read the returned data and time stamps
- `ds4302_can_msg_sleep` to deactivate the message
- `ds4302_can_msg_wakeup` to reactivate the message
- `ds4302_can_msg_clear` to clear the message object data
- `ds4302_can_msg_processed_register` to register the processed function
- `ds4302_can_msg_processed_request` to request the processed function
- `ds4302_can_msg_processed_read` to read the returned data

Parameters

rqtxMsg Specifies the pointer to the related RQTX message.

inform Specifies the information values to be updated. You can combine the predefined symbols with the logical OR operator. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_INFO`	Returns no information.
`DS4302_CAN_DATA_INFO`	Updates the data and datalen parameters (needed for receive and request (RQRX) messages).
`DS4302_CAN_MSG_INFO`	Updates the message identifier and the message format.
`DS4302_CAN_TIMECOUNT_INFO`	Updates the timestamp and deltatime parameters.

subinterrupt Specifies the subinterrupt number for a received message. The valid range is 0 … 30.

Note

The interrupt number 31 is occupied by the buffer overflow warning interrupt (DS4302_CAN_SUBINT_BUFFERWARN). Do not use this number for any other interrupt.

Use the following predefined symbol to select no interrupt for the RQRX message:

Predefined Symbol	Meaning
`DS4302_CAN_NO_SUBINT`	No interrupt for the RQRX message.

Return value

canMsg Specifies the pointer to the `DSxyz_canMsg` structure.

Messages

The following messages are defined:

ID	Type	Message	Description
101	Error	ds4302_can_msg_rqrx_register(x,..) memory allocation error on master	Memory allocation error. No free memory on the master.
102	Error	ds4302_can_msg_rqrx_register(x,..) queue: Illegal communication queue	There is no communication channel with this queue number.
103	Error	ds4302_can_msg_rqrx_register(x,..) index: illegal function index	The index does not exist in the command table and is not equal to DS4302_CAN_AUTO_INDEX.
104	Error	ds4302_can_msg_rqrx_register(x,..) queue: master to slave overflow	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted.
106	Error	ds4302_can_msg_rqrx_register(x,..) slave: not responding	The slave did not finish the initialization of the communication within one second. This error may be caused by an active I/O error line (/IOERR) of the PHS-bus due to a wrong initialization order of the I/O boards connected to the PHS-bus. You have to initialize the I/O boards with the

ID	Type	Message	Description
			appropriate board_init functions and the DS4302 has to be the last board to be initialized.
107	Error	ds4302_can_msg_rqrx_register(x,..) slave: memory allocation error	Memory allocation error on the slave. There are too many functions registered.
108	Error	ds4302_can_msg_rqrx_register(x,..) queue: slave to master overflow	Not enough memory space between the slave write pointer and the master read pointer. The slave tries to write data to a filled queue. To prevent this error, deactivate all messages with `ds4302_can_msg_sleep` or `ds4302_can_channel_all_sleep` when registering messages or services.
140	Error	ds4302_can_msg_rqrx_register(x,..) format: wrong format	Only the symbols `DS4302_CAN_STD` and `DS4302_CAN_EXT` are allowed for the parameter format.
141	Error	(x,..) subint: use range 0..14 ! ds4302_can_msg_rqrx_register(x,..) subint: use range 0..30 !	The subinterrupt number must be within the range 0 … 30.
142	Error	ds4302_can_msg_rqrx_register(x,..) subint: used for busoff!	The specified subinterrupt number is used for the CAN channel bus off subinterrupt.
143	Error	ds4302_can_msg_rqrx_register(x,..) id: Illegal id or id conflict	The CAN controller does not install the identifier specified in the program. For further information, see `ds4302_canService` on page 13.
144	Error	ds4302_can_msg_rqrx_register(x,..): Too much messages (max.200)!	The total number of registered messages is limited to 200. The program is aborted.
152	Error	ds4302_can_msg_rqrx_register(x,..) canCh: the CAN channel wasn't initialized	This message is displayed if: ▪ You try to register a CAN message on an uninitialized CAN channel. ▪ You try to register a CAN service on an uninitialized CAN channel. Use `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to initialize the CAN channel.

Example

For examples of how to use this function, refer to Example of Handling Request and Remote Messages on page 96.

```
ds4302_canMsg* rqrxMsg = ds4302_can_msg_rqrx_register(
        rqtxMsg,
        DS4302_CAN_DATA_INFO,
        DS4302_CAN_NO_SUBINT);
```

Related topics

References

ds4302_can_msg_rm_register

Syntax

```
ds4302_canMsg* ds4302_can_msg_rm_register(
      const ds4302_canChannel* canCh,
      const Int32 queue,
      const UInt32 identifier,
      const UInt32 format,
      const UInt32 inform,
      const Int32 subinterrupt);
```

Include file

`Ds4302.h`

Purpose

To register a remote message on the slave DS4302.

Description

If no error occurs, the function returns a pointer to the `ds4302_canMsg` structure.

Use the returned handle when calling one of the following functions:

- `ds4302_can_msg_write` to support the remote message with data
- `ds4302_can_msg_read` to read the returned time stamps
- `ds4302_can_msg_sleep` to deactivate the message
- `ds4302_can_msg_wakeup` to reactivate the message
- `ds4302_can_msg_clear` to clear the message object data
- `ds4302_can_msg_processed_register` to register the processed function
- `ds4302_can_msg_processed_request` to request the processed function
- `ds4302_can_msg_processed_read` to read the returned data

A remote message is a special kind of a transmit message. It is sent only if the CAN controller has received an associated request message and carries the requested data.

Note

A remote message permanently occupies a mailbox on the slave DS4302 CAN channel. Therefore, only 10 remote messages are allowed within the same model for each CAN channel to ensure secure CAN operation. If this is not done, the function outputs an error and aborts the program.

Parameters

canCh Specifies the pointer to the `ds4302_canChannel` structure.

queue Specifies the communication channel within the range 0 … 5.

identifier Specifies the identifier of the message.

format Specifies the message format. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_STD`	11-bit standard format, CAN 2.0A
`DS4302_CAN_EXT`	29-bit extended format, CAN 2.0B

inform Specifies the information values to be updated. You can combine the predefined symbols with the logical OR operator. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_INFO`	Returns no information.
`DS4302_CAN_MSG_INFO`	Updates the message identifier and the message format.
`DS4302_CAN_TIMECOUNT_INFO`	Updates the timestamp and the **`deltatime`** parameters.
`DS4302_CAN_DELAYCOUNT_INFO`	Updates the **`delaytime`** parameter.

subinterrupt Specifies the subinterrupt number for a received message. The valid range is 0 … 30.

Note

The interrupt number 31 is occupied by the buffer overflow warning interrupt (DS4302_CAN_SUBINT_BUFFERWARN). You must not use this number for any other interrupt.

Use the following predefined symbol to select no interrupt for the RM message:

Predefined Symbol	Meaning
`DS4302_CAN_NO_SUBINT`	No interrupt for the RM message

Return value	**canMsg**	This function returns the pointer to the `ds4302_canMsg` structure.

Messages

The following error and warning messages are defined:

ID	Type	Message	Description
101	Error	ds4302_can_msg_rm_register(x,..) memory allocation error on master	Memory allocation error. No free memory on the master.
102	Error	ds4302_can_msg_rm_register(x,..) queue: Illegal communication queue.	There is no communication channel with this queue number.
103	Error	ds4302_can_msg_rm_register(x,..) index: illegal function index	The index does not exist in the command table and is not equal to DS4302_CAN_AUTO_INDEX.
104	Error	ds4302_can_msg_rm_register(x,..) queue: master to slave overflow	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted.
106	Error	ds4302_can_msg_rm_register(x,..) slave: not responding	The slave did not finish the initialization of the communication within one second. This error may be caused by an active I/O error line (/IOERR) of the PHS-bus due to a wrong initialization order of the I/O boards connected to the PHS-bus. You have to initialize the I/O boards with the appropriate board_init functions and the DS4302 has to be the last board to be initialized.
107	Error	ds4302_can_msg_rm_register(x,..) slave: memory allocation error	Memory allocation error on the slave. There are too many functions registered.
108	Error	ds4302_can_msg_rm_register(x,..) queue: slave to master overflow	Not enough memory space between the slave write pointer and the master read pointer. The slave tries to write data to a filled queue. To prevent this error, deactivate all messages with `ds4302_can_msg_sleep` or `ds4302_can_channel_all_sleep` when registering messages or services.
140	Error	ds4302_can_msg_rm_register(x,..) format: wrong format	Only the symbols DS4302_CAN_STD and DS4302_CAN_EXT are allowed for the parameter format.
141	Error	ds4302_can_msg_rm_register(x,..) subint: use range 0..30 !	The subinterrupt number must be within the range 0 … 30.
142	Error	ds4302_can_msg_rm_register(x,...) subint: used for busoff.	The specified subinterrupt number is used for the CAN channel bus off subinterrupt.
143	Error	ds4302_can_msg_rm_register(x,..) id: illegal id or id conflict	The CAN controller does not install the identifier specified in the program. For further information, see `ds4302_canService` on page 13.
144	Error	ds4302_can_msg_rm_register(x,...) Too much messages (max. 200)!	The total number of registered messages is limited to 200. The program is aborted.
150	Error	ds4302_can_msg_rm_register(x,...) no rm mailbox free (max. 10).	For each channel, only 10 remote messages are allowed within the same model to ensure secure CAN operation. If this is not done, the function outputs an error and the program is aborted.

ID	Type	Message	Description
152	Error	ds4302_can_msg_rm_register(x,...) canCh: the CAN channel wasn't initialized	This message is displayed if: ▪ You try to register a CAN message on an uninitialized CAN channel. ▪ You try to register a CAN service on an uninitialized CAN channel. Use `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to initialize the CAN channel.

Example

For examples of how to use this function, refer to Example of Handling Request and Remote Messages on page 96.

```
ds4302_canMsg* rmMsg = ds4302_can_msg_rm_register(
         canCh,
         0,
         0x123,
         DS4302_CAN_STD,
         DS4302_CAN_TIMECOUNT_INFO,
         DS4302_CAN_NO_SUBINT);
```

Related topics

References

ds4302_can_msg_set

Syntax

```
Int32 ds4302_can_msg_set(
      ds4302Msg* msg,
      const UInt32 type,
      const void* value );
```

Include file

`Ds4302.h`

Purpose

To set the properties of a CAN message.

Description

This function allows you to

- Receive different message IDs with one message via a bitmask (type = DS4302_CAN_MSG_MASK),
- Set the send period for a TX or RQ message (type = DS4302_CAN_MSG_PERIOD),
- Set the identifier for a TX or RQ message (type = DS4302_CAN_MSG_ID) or
- Set the queue depth for a message (type = DS4302_CAN_MSG_QUEUE).
- Set the length for a message (type = DS4302_CAN_MSG_LEN).

Note

For DS4302_CAN_MSG_MASK the following rules apply:

- For each CAN channel, only one mask for STD and one mask for EXT messages is allowed.
- If you call `ds4302_can_msg_set` for another message, the bitmask is removed from the first message.
- Using the bitmask might cause conflicts with messages installed for one message ID. In this case, message data is received via the message installed for this ID.
- You can skip the bitmask by setting all bits to `"must match"` (`0xFFFFFFFF`) again.

Parameters

msg Specifies the pointer to the message structure.

type Defines the property to be specified. Use one of the predefined symbols:

Predefined Symbol	Meaning
`DS4302_CAN_MSG_MASK`	To set the arbitrary mask for an RX message
`DS4302_CAN_MSG_PERIOD`	To set the send period for a TX or RQ message
`DS4302_CAN_MSG_ID`	To set the identifier for a TX or RQ message
`DS4302_CAN_MSG_QUEUE`	To set the queue depth for a message
`DS4302_CAN_MSG_LEN`	To set the data length code (DLC) for a TX, RQTX, or RM message

value Specifies the value to be set for the defined `type`.

For the DS4302_CAN_MSG_LEN type, you can specify the data length code (DLC) value (UInt32) in the range 0 ... 8 bytes.

Note

If the specified length exceeds 8 bytes, the function sets the length to 8 bytes.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
`DS4302_CAN_TYPE_ERROR`	The operation is not allowed for the specified message object.
`DS4302_CAN_MSG_TYPE_ERROR`	The function is not available for the specified message type. It is available only for TX, RQTX, and RM messages.

Example

This example shows how to receive different message IDs with one message:

Install one message with a bitmask that allows you to set some bits of the mask to "don’t care" via ds4302_can_msg_set.

```
UInt32 mask = 0xFFFFFFF0; // Sets the last four bits to
                          // "Don’t Care".
ds4302_can_msg_set( msg, DS4302_CAN_MSG_MASK, &mask );
```

Example

This example shows how to receive different message IDs with one message via a bitmask:

- A message with ID 0x120 was registered. Now, you set the bitmask via `ds4302_can_msg_set(msg, DS4302_CAN_MSG_MASK,&mask);` with `mask = 0xFFFFFFF0`.

  This lets you receive the message IDs 0x120, 0x121, ..., 0x12F.
- A message with ID 0x120 was registered. Now, you set the bitmask to 0x1FFFFFEF. This lets you receive the message IDs 0x120 and 0x130.

Example

This example shows how to apply the DS4302_CAN_MSG_QUEUE option.

You can define a buffer for each message to receive several messages. Otherwise, only the most recently received message will be available.

- Register the message as usual

  ```
  myMsg = ds4302_can_msg_xx_register(...)
  ```

  By default, `myMsg` stores only one message.
- Define a message queue of length *n* for `myMsg`

  ```
  ds4302_can_msg_set(myMsg, DS4302_CAN_MSG_QUEUE, &n)
  ```

- Call `ds4302_can_msg_read(myMsg)` repeatedly until the function returns DS4302_CAN_NO_DATA.

```
UInt32 n;
canMsg = ds4302_can_msg_rx_register( canCh,…
n = 5000;
ds4302_can_msg_set(canMsg, DS4302_CAN_MSG_QUEUE, &n);
...
while(DS4302_CAN_NO_DATA != (error =
             ds4302_can_msg_read(canMsg)))
{
   if(DS4302_CAN_DATA_LOST == error)
   {
      /* error handling */
   }
   else if(DS4302_CAN_NO_ERROR == error)
   {
      /* process the message */
   }
   else /* DS4302_CAN_NO_DATA == error */
   {
      /* no further CAN-messages */
   }
}
```

Related topics

References

ds4302_can_msg_read.. 77

ds4302_can_msg_rqtx_activate

Syntax

```
Int32 ds4302_can_msg_rqtx_activate(
     const ds4302_canMsg* canMsg);
```

Include file

`Ds4302.h`

Purpose

To activate the request transmission message on the slave DS4302 registered by `ds4302_can_msg_rqtx_register`.

Description

This function does not send the message. Sending the message is done by the timer for cyclic sending or by calling `ds4302_can_msg_trigger` for acyclic sending. Use the returned handle from the function `ds4302_can_msg_rqtx_register` to call this function.

Parameters

canMsg	Specifies the pointer to the `ds4302_canMsg` structure.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
`DS4302_CAN_TYPE_ERROR`	The operation is not allowed for the given message object.

Example

For examples of how to use this function, refer to Example of Handling Request and Remote Messages on page 96.

Related topics

References

ds4302_can_msg_rqtx_register 50
ds4302_can_msg_trigger 79

ds4302_can_msg_write

Syntax

```
Int32 ds4302_can_msg_write(
      const ds4302_canMsg* canMsg,
      const UInt32 datalen,
      const UInt32* data);
```

Include file

`Ds4302.h`

Purpose

To write CAN message data.

Description

There are differences for the following message types:

- TX message

  Calling this function for the first time prepares the message to be sent with the specified parameters in the message register function. A TX message with a repetition time is sent automatically with the specified value. A TX message registered by DS4302_CAN_TRIGGER_MSG is sent only when calling `ds4302_can_msg_trigger` or `ds4302_can_msg_send`.

Calling this function again updates CAN message data and data length.

- RM message

  Calling this function for the first time prepares and activates the remote message to be sent with the specified data and data length. The remote message is sent when a corresponding request message is received.

  Calling this function again updates CAN message data and data length.

Use the returned handle from the function `ds4302_can_msg_tx_register` or `ds4302_can_msg_rm_register` to call this function.

Parameters

canMsg Specifies the pointer to the `ds4302_canMsg` structure.

datalen Specifies the length of the CAN message data. The valid range is 0 … 8 bytes.

data Specifies the buffer for CAN message data.

Return value

This function returns the error code; the following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function has been performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
`DS4302_CAN_TYPE_ERROR`	The operation is not allowed for the specified message object.

Example

For examples, refer to:

- Example of Handling Transmit and Receive Messages on page 94
- Example of Handling Request and Remote Messages on page 96
- Example of Using Subinterrupts on page 98

Related topics

References

ds4302_can_msg_rm_register 58
ds4302_can_msg_send 67
ds4302_can_msg_trigger 79
ds4302_can_msg_tx_register 42

ds4302_can_msg_send

Syntax

```
Int32 ds4302_can_msg_send(
      const ds4302_canMsg* canMsg,
      const UInt32 datalen,
      const UInt32* data,
      const Float32 delay);
```

Include file

`Ds4302.h`

Purpose

To write CAN message data and send the data immediately after the delay time. To send the transmit message with new data.

Description

The transmit message must have been registered by calling `ds4302_can_msg_tx_register`. Then `ds4302_can_msg_send` writes the CAN message data to the dual-port memory. After this, the message is set up on the CAN controller and the sending of the message is started. The message is sent according to the specified parameters in the register function.

Use the returned handle from the function `ds4302_can_msg_tx_register` to call this function.

Note

Suppose the `ds4302_can_msg_send` function is called twice. If the interval between the function calls is short, the second function call might occur *before* the TX message was sent by the first function call. In this case, the TX message is sent only once, with the data of the second function call.

Parameters

canMsg Specifies the pointer to the `ds4302_canMsg` structure.

datalen Specifies the length of the CAN message data. The valid range is 0 … 8 bytes.

data Specifies the buffer for CAN message data.

delay Sends the message after the delay time. The valid range is 0.0 … 100.0 seconds.

Note

For board revision DS4302-05, the maximum time value is 40 seconds. You can inspect the revision of your DS4302 using dSPACE experiment software.

Return value

This function returns the error code; the following symbols are predefined:

Predefined Symbol	Meaning
DS4302_CAN_NO_ERROR	The function was performed without error.
DS4302_CAN_BUFFER_OVERFLOW	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
DS4302_CAN_TYPE_ERROR	The operation is not allowed for the specified message object.

Example

```
UInt32 txData[8] = {1,2,3,4,5,6,7,8};
ds4302_can_msg_send (txMsg, 8, txData, 0.005);
```

Related topics

References

ds4302_can_msg_send_id

Syntax

```
Int32 ds4302_can_msg_send_id (
      ds4302_canMsg* canMsg,
      const UInt32 id,
      const UInt32 datalen,
      const UInt8* data,
      const Float32 delay);
```

Include file

`Ds4302.h`

Purpose

To send a message with a modified identifier. This lets you send any message ID with one registered message.

Parameters

canMsg Specifies the pointer to the `ds4302_canMsg` structure.

id Specifies the ID of the message to be modified.

datalen Specifies the length of the CAN message data. The valid range is 0 … 8 bytes.

data Specifies the buffer for CAN message data.

delay Sends the message after the delay time. The valid range is 0.0 … 100.0 seconds.

Note

For board revision DS4302-05, the maximum time value is 40 seconds.
You can inspect the revision of your DS4302 using dSPACE experiment software.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
`DS4302_CAN_TYPE_ERROR`	The operation is not allowed for the specified message object.

Note

- The message format is determined by the format in which the message was installed when it was used for the first time.
- You have to use a handshake mechanism to send a message via `ds4302_can_msg_send_id` to make sure that a message installed for the message object has been sent already.
  Each message object is buffered only once on the slave. This might cause conflicts when you try to send several message objects with different IDs.

Example

The `ds4302_can_msg_send_id` function lets you send any message ID with one registered message.

```
ds4302_can_msg_send_id(msg, 0x123, data, 8, 0.001)
```

Related topics

References

ds4302_can_msg_queue_level 70
ds4302_can_msg_send 67
ds4302_can_msg_tx_register 42

ds4302_can_msg_queue_level

Syntax

```
Int32 ds4302_can_msg_queue_level (
         ds4302_canMsg* canMsg);
```

Include file

`Ds4302.h`

Purpose

To return the number of messages stored in the message queue allocated on the master with `ds4302_can_msg_set(msg, DS4302_CAN_MSG_QUEUE, &size)`.

Description

Use `ds4302_can_msg_read` to copy the messages from the communication channel to the message buffer.

Note

This is not the number of messages in the DPMEM.

Parameters

canMsg	Specifies the pointer to the `ds4302_canMsg` structure.

Return value

This function returns the number of messages in the message queue.

Related topics

References

ds4302_can_msg_read 77
ds4302_can_msg_set 61

ds4302_can_msg_txqueue_init

Syntax

```
Int32 ds4302_can_msg_txqueue_init(
      ds4302_canMsg* canMsg,
      const UInt32 overrun_policy,
      Float32 delay);
```

Include file

`Ds4302.h`

Purpose

To initialize the transmit queue that is used to queue messages sent by the `ds4302_can_msg_send_id_queued` function.

Description

The function allocates a circular buffer on the slave with the specified overrun policy, where the transmit orders from the `ds4302_can_msg_send_id_queued` function are stored. The queue stores up to 64 message entries.

Parameter

canMsg Specifies the pointer to the `ds4302_canMsg` structure.

overrun_policy Selects the overrun policy of the transmit queue. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_TXQUEUE_OVERRUN_OVERWRITE`	The oldest message is overwritten.
`DS4302_CAN_TXQUEUE_OVERRUN_IGNORE`	The oldest message is kept. The new message is lost.

delay Specifies the delay between the messages of the transmit queue within the range 0.0 … 10 s.

Note

- Even if a delay of 0 seconds is specified, the distance between two message frames is greater than 0. The length of this gap depends on the load of the slave. If the delay is smaller than 0, the function sets the delay to 0. The real delay between two message frames might not be constant due to jitter. The jitter of the delay also depends on the load of the slave.
- Only two message objects (one STD and one EXT format message) can be used for queuing for every channel. Nevertheless `ds4302_can_msg_send_id_queued` allows the identifier of the message object to be changed.
- The function can be called again to change the delay or to assign the transmit queue to another message. The old messages in the transmit queue are lost (not transmitted) if the transmit queue is initialized again.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
DS4302_CAN_NO_ERROR	The transmit queue was initialized successfully.
DS4302_CAN_BUFFER_OVERFLOW	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
DS4302_CAN_TXQUEUE_INIT_NOT_REG_ERROR	The message (canMsg) was not registered. The operation is aborted.
DS4302_CAN_TXQUEUE_INIT_MSG_TYPE_ERROR	The message (canMsg) is not a TX message. The operation is aborted.

Messages

The following messages are defined:

ID	Type	Message	Description
154	Error	ds4302_can_msg_txqueue_init(): TX message is not registered	The message was not registered successfully.
155	Error	ds4302_can_msg_txqueue_init(): not a TX message	The specified message is not a TX message.
301	Warning	ds4302_can_msg_txqueue_init(): delay time: too high (max. 10 s). Set to maximum.	The delay time must be within the range 0 … 10 s.

Example

The following example shows you how to initialize a TX queue.

```
void main()
{
   ds4302_canMsg* txMsg;
...
   txMsg = ds4302_can_msg_tx_register(txCh,
            2, 0x1, DS4302_CAN STD,
            DS4302_CAN_TIMECOUNT_INFO | DS4302_CAN_MSG_INFO,
            1, 0.0,
            DS4302_CAN_TRIGGER_MSG, 0);
   ds4302_can_msg_txqueue_init (
         txMsg, DS4302_CAN_TXQUEUE_OVERRUN_OVERWRITE, 0.01);
...
}
```

Related topics

References

ds4302_can_msg_send_id_queued.. 73

ds4302_can_msg_send_id_queued

Syntax

```
Int32 ds4302_can_msg_send_id_queued(
      ds4302_canMsg* canMsg,
      const UInt32 id,
      const UInt32 data_len,
      const UInt32* data);
```

Include file

`Ds4302.h`

Purpose

To build a transmit order and transmit it in the same order as the function is called.

Description

If no queue overflow occurs, each message is transmitted. In the case of queue overflow (number of messages is greater than 64), the newest message overwrites the oldest one or the oldest messages are kept while new messages are lost. See `ds4302_can_msg_txqueue_init` on page 71.

The `DS4302_CAN_SERVICE_TXQUEUE_OVERFLOW_COUNT` service allows the overflow counter of the transmit queue to be requested to check whether an overflow occurred.

Parameter

canMsg	Specifies the pointer to the `ds4302_canMsg` structure.
id	Specifies the CAN message identifier type (STD/EXT). The identifier type must correspond to the type (STD/EXT) of the registered message object. This allows the identifier of the message object to be changed during run time.
data_len	Specifies the length of data within the range 0 … 8.
data	Specifies the message data.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The transmit queue was initialized successfully.
`DS4302_CAN_BUFFER_OVERFLOW`	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
`DS4302_CAN_SEND_ID_QUEUED_INIT_ERROR`	The transmit queue for TX messages was not initialized.

Messages

The following messages are defined:

ID	Type	Message	Description
153	Error	ds4302_can_msg_send_id_queued(): TX queue: Not initialized!	The transmit queue was not initialized.

Example

The following example shows how to build a transmit sequence for a TX queue.

```
void main()
   ds4302_canMsg* txMsg;
   UInt32 txMsgData[8];
   ...
   txMsg = ds4302_can_msg_tx_register( txCh,
            2, 0x1, DS4302_CAN_STD, DS4302_CAN_TIMEOUTINFO|
            DS4302_CAN_MSG_INFO,
            1, 0.0,
            DS4302_CAN_TRIGGER_MSG, 0);
/* Initialize a transmit queue with delay = 0.01s */
   ds4302_can_msg_txqueue_init(
         txMsg, DS4302_CAN_TXQUEUE_OVERRUN_OVERWRITE, 0.01);
/* Write three messages to the transmit queue. The first*/
/* message is transmitted immediately. The following */
/* messages are transmitted with a delay of 0.01s. */
   txMsgData[0] = 0x01;
   ds4302_can_msg_send_id_queued(txMsg, 0x12, 1, txMsgData);
   txMsgData[0] = 0x02;
   ds4302_can_msg_send_id_queued(txMsg, 0x13, 1, txMsgData);
   txMsgData[0] = 0x03;
   ds4302_can_msg_send_id_queued(txMsg, 0x14, 1, txMsgData);
}
```

Related topics

References

ds4302_can_msg_txqueue_init.. 71

ds4302_can_msg_txqueue_level_read

Syntax

```
UInt32 ds4302_can_msg_txqueue_level_read(
      const ds4302_canMsg* canMsg);
```

Include file

`Ds4302.h`

Purpose	To read the fill level of the transmit queue for the specified TX message on the CAN slave.
Description	The function reads the fill level of the transmit queue for the specified TX message on the CAN slave. **Note** The TX messages pending in the command queue between the CAN master and the CAN slave are not taken into account.
Parameter	**canMsg** Specifies the pointer to the `ds4302_canMsg` structure.
Return value	**Level of TX-queue** The number of TX messages in the transmit queue on the CAN slave (0 … 64).

ds4302_can_msg_sleep

Syntax

```
Int32 ds4302_can_msg_sleep(
    const ds4302_canMsg* canMsg);
```

Include file

`Ds4302.h`

Purpose

The purpose depends on the message type:

- TX, RQTX, and RM messages
  To stop the transmission of the message to the CAN bus.
- RX and RQRX messages
  To stop the transmission of the message data from the slave to the master.

Description

The message is deactivated and remains in sleep mode until it is reactivated by calling `ds4302_can_msg_wakeup` or `ds4302_can_channel_all_wakeup`.

Parameters

canMsg	Specifies the pointer to the `ds4302_canMsg` structure.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
`DS4302_CAN_TYPE_ERROR`	The operation is not allowed for the given message object.

Example

```
ds4302_can_msg_sleep(txMsg);
```

Related topics

References

ds4302_can_msg_wakeup

Syntax

```
Int32 ds4302_can_msg_wakeup(
      const ds4302_canMsg* canMsg);
```

Include file

`Ds4302.h`

Purpose

To reactivate a message that has been deactivated by calling the `ds4302_can_msg_sleep` or `ds4302_can_channel_all_sleep` function.

Parameters

canMsg	Specifies the pointer to the `ds4302_canMsg` structure.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
`DS4302_CAN_TYPE_ERROR`	The operation is not allowed for the given message object.

Example

```
ds4302_can_msg_wakeup(txMsg);
```

Related topics

References

ds4302_can_msg_read

Syntax

```
Int32 ds4302_can_msg_read(
      ds4302_canMsg* canMsg);
```

Include file

`Ds4302.h`

Purpose

To read the data length, the data, and the status information from the dual-port memory.

Description

The return value provides information on whether or not the data is new. If not, the existing parameter values remain unchanged.

You can call this function several times for one message object to read all the messages available in the message buffer (see also `ds4302_can_msg_set` on page 61). By default, only one message can be received.

Use the function `ds4302_can_msg_clear` to clear the message data and time stamps. This is useful for simulation start/stop transitions.

Note

The status information that is returned depends on the previously specified inform parameter in the register function that corresponds to the message.

Parameters

canMsg Specifies the pointer to the `ds4302_canMsg` structure.

Parameter	Meaning
data	Buffer with the updated data
datalen	Data length of the message
deltatime	Delta time of the message
timestamp	Time stamp of the message
delaytime	Delaytime of the message
processed	Processed flag of the message
identifier	Identifier of the message
format	Format of the identifier

Return value

This function returns the error code. The following symbols are predefined:

Symbols	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_NO_DATA`	No data was updated.
`DS4302_CAN_DATA_LOST`	The input data of a previous request for the specified function was overwritten.

Example

For examples, refer to:
- Example of Handling Transmit and Receive Messages on page 94
- Example of Handling Request and Remote Messages on page 96
- Example of Using Subinterrupts on page 98

Related topics

References

ds4302_can_msg_clear........ 80
ds4302_can_msg_set........ 61

ds4302_can_msg_trigger

Syntax

```
Int32 ds4302_can_msg_trigger(
      const ds4302_canMsg* canMsg,
      const Float32 delay);
```

Include file

`Ds4302.h`

Purpose

To send a transmit or request message immediately after the specified delay time.

Description

This function can be used for acyclic message sending. Use the returned handle from the `ds4302_can_msg_tx_register` or `ds4302_can_msg_rqtx_register` function to call this function.

Parameters

canMsg Specifies the pointer to the `ds4302_canMsg` structure.

delay Sends the message after the delay time. The valid range is 0.0 … 100.0 seconds.

Note

For board revision DS4302-05, the maximum time value is 40 seconds. You can inspect the revision of your DS4302 using dSPACE experiment software.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	**Meaning**
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
`DS4302_CAN_TYPE_ERROR`	The operation is not allowed for the specified message object.

Example

```
ds4302_can_msg_trigger(txMsg, 0.005); /* 5 ms delay */
```

Related topics

References

ds4302_can_msg_clear

Syntax

```
void ds4302_can_msg_clear(
         ds4302_canMsg* canMsg);
```

Include file

`Ds4302.h`

Purpose

To clear the following message data: data[8], datalen, timestamp, deltatime, timecount, delaytime and processed.

Description

This is useful for simulation start/stop transitions.

Use the returned handle from the message register functions to call this function.

Note

The structure members identifier, format, module, queue, index, msg_no, type, inform, canChannel, and msgService are untouched, because any manipulation of these structure members would corrupt the message object.

Parameters

canMsg	Specifies the pointer to the `ds4302_canMsg` structure.

Return value

None

Example

```
ds4302_can_msg_clear(rxMsg);
```

Related topics

References

ds4302_can_msg_processed_register

Syntax

```
void ds4302_can_msg_processed_register(
         ds4302_canMsg* canMsg);
```

Include file

`Ds4302.h`

Purpose

To register the processed function in the command table.

Use `ds4302_can_msg_processed_read` to read the processed flag and time stamp without registering the message with the inform parameter `DS4302_CAN_TIMECOUNT_INFO`.

Parameters

canMsg Specifies the pointer to the `ds4302_canMsg` structure.

Return value

None

Messages

The following error and warning messages are defined:

ID	Type	Description	Message
101	Error	ds4302_can_msg_processed_register(x,..) memory allocation error on master	Memory allocation error. No free memory on the master.
102	Error	ds4302_can_msg_processed_register(x,..) queue: Illegal communication queue.	There is no communication channel with this queue number.
103	Error	ds4302_can_msg_processed_register(x,..) index: illegal function index	The index does not exist in the command table and is not equal to DS4302_CAN_AUTO_INDEX.
104	Error	ds4302_can_msg_processed_register(x,..) queue: master to slave overflow	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted.

ID	Type	Description	Message
106	Error	ds4302_can_msg_processed_register(x,..) slave: not responding	The slave did not finish the initialization of the communication within one second. This error may be caused by an active I/O error line (/IOERR) of the PHS-bus due to a wrong initialization order of the I/O boards connected to the PHS-bus. You have to initialize the I/O boards with the appropriate board_init functions and the DS4302 has to be the last board to be initialized.
107	Error	ds4302_can_msg_processed_register(x,..) slave: memory allocation error	Memory allocation error on the slave. There are too many functions registered.
108	Error	ds4302_can_msg_processed_register(x,..) queue: slave to master overflow	Not enough memory space between the slave write pointer and the master read pointer. The slave tries to write data to a filled queue. To prevent this error, deactivate all messages with `ds4302_can_msg_sleep` or `ds4302_can_channel_all_sleep` when registering messages or services.

Example

```
ds4302_can_msg_processed_register(rxMsg);
```

Related topics

References

ds4302_can_msg_processed_request

Syntax

```
Int32 ds4302_can_msg_processed_request(
     const ds4302_canMsg* canMsg);
```

Include file

`Ds4302.h`

Purpose

To request the message processed information from the slave DS4302.

Parameters

canMsg Specifies the pointer to the `ds4302_canMsg` structure.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
DS4302_CAN_NO_ERROR	The function was performed without error.
DS4302_CAN_BUFFER_OVERFLOW	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
DS4302_CAN_NO_DATA	`ds4302_can_msg_processed_request` was called without registering the function with `ds4302_can_msg_processed_register` or an empty canMsg structure was handled.

Example

```
ds4302_can_msg_processed_request(rxMsg);
```

Related topics

References

ds4302_can_msg_processed_read...83
ds4302_can_msg_processed_register...81

ds4302_can_msg_processed_read

Syntax

```
Int32 ds4302can_msg_processed_read(
      ds4302_canMsg* canMsg,
      double* timestamp,
      UInt32* processed);
```

Include file

`Ds4302.h`

Purpose

To read the message processed information from the slave DS4302.

Description

Prior to this, this information must have been requested by the master calling the function `ds4302_can_msg_processed_request` that demands the processed flag and the time stamp from the slave DS4302.

Parameters

canMsg Specifies the pointer to the `ds4302_canMsg` structure.

timestamp Specifies the time stamp when the message was last sent or received.

processed Specifies the processed flag of the message. The following symbols are predefined:

Symbols	Meaning
`DS4302_CAN_PROCESSED`	Message has been sent/received since the last execution call.
`DS4302_CAN_NOT_PROCESSED`	Message has not been sent/received since the last execution call.

Return value

This function returns the error code. The following symbols are predefined:

Symbols	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_NO_DATA`	No data was updated.
`DS4302_CAN_DATA_LOST`	The input data of a previous request for the specified function was overwritten.

Related topics

References

ds4302_can_msg_processed_register.. 81
ds4302_can_msg_processed_request.. 82

CAN Service Functions

Introduction	To get information on errors and status information.

Where to go from here

Information in this section

ds4302_can_service_register

Syntax

```
ds4302_canService* ds4302_can_service_register(
     const ds4302_canChannel* canCh,
     const UInt32 service_type);
```

Include file

`Ds4302.h`

Purpose

To register the service function.

Description

Use `ds4302_can_service_read` to read a registered service specified by the `service_type` parameter.

Parameters

canCh Specifies the pointer to the `ds4302_canChannel` structure.

service_type Specifies the service to be installed. For additional information, see the `type` parameter of `ds4302_canService` structure. You can use the bitwise OR operator to combine several services.

Return value

canService This function returns the pointer to the `ds4302_canService` structure.

Messages

The following messages are defined:

ID	Type	Message	Description
101	Error	ds4302_can_service_register(x,..) memory allocation error on master	Memory allocation error. No free memory on the master.
102	Error	ds4302_can_service_register(x,..) queue: Illegal communication queue.	There is no communication channel with this queue number.
103	Error	ds4302_can_service_register(x,..) index: illegal function index	The index does not exist in the command table and is not equal to DS4302_CAN_AUTO_INDEX.
104	Error	ds4302_can_service_register(x,..) queue: master to slave overflow	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted.
106	Error	ds4302_can_service_register(x,..) slave: not responding	The slave did not finish the initialization of the communication within one second. This error may be caused by an active I/O error line (/IOERR) of the PHS-bus due to a wrong initialization order of the I/O boards connected to the PHS-bus. You have to initialize the I/O boards with the appropriate board_init functions and the DS4302 has to be the last board to be initialized.
107	Error	ds4302_can_service_register(x,..) slave: memory allocation error	Memory allocation error on the slave. There are too many functions registered.
108	Error	ds4302_can_service_register(x,..) queue: slave to master overflow	Not enough memory space between the slave write pointer and the master read pointer. The slave tries to write data to a filled queue. To prevent this error deactivate all messages with `ds4302_can_msg_sleep` or `ds4302_can_channel_all_sleep` when registering messages or services.
152	Error	ds4302_can_service_register(x,..) canCh: the CAN channel wasn't initialized	This message is displayed if: ▪ You try to register a CAN message on an uninitialized CAN channel. You try to register a CAN service on an uninitialized CAN channel. ▪ You try to start an uninitialized CAN channel with `ds4302_can_channel_start`. Use `ds4302_can_channel_init` or `ds4302_can_channel_init_advanced` to initialize the CAN channel.

Example

For a detailed example of how to use this function, refer to Example of Using Service Functions on page 100.

```
ds4302_canService* service;
...
service = ds4302_can_service_register(txCh,
    DS4302_CAN_SERVICE_TX_OK |
    DS4302_CAN_SERVICE_TXQUEUE_OVERFLOW_COUNT );
```

Related topics

References

ds4302_can_service_request

Syntax

```
Int32 ds4302_can_service_request(
      const ds4302_canService* service);
```

Include file

`Ds4302.h`

Purpose

To request the service information from the slave DS4302. Use `ds4302_can_service_read` to read the registered service.

Description

Use the returned handle from the function `ds4302_can_service_register` on page 85 to call this function.

Parameters

service Specifies the pointer to the `ds4302_canService` structure.

Return value

This function returns the error code. The following symbols are predefined:

Predefined Symbol	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_BUFFER_OVERFLOW`	Not enough memory space between the master write pointer and the slave read pointer. The operation is aborted. Repeat the function until it returns DS4302_CAN_NO_ERROR.
`DS4302_CAN_NO_DATA`	`ds4302_can_service_request` was called without registering the function with `ds4302_can_service_register` or an empty service structure was handled.

Example

For an example of how to use this function, refer to Example of Using Service Functions on page 100.

Related topics

References

ds4302_can_service_read

Syntax

```
Int32 ds4302_can_service_read(
         ds4302_canService* service);
```

Include file

`Ds4302.h`

Purpose

To read the service information from the slave DS4302.

Description

Prior to this, this information must have been requested by the master calling the `ds4302_can_service_request` function that asks for the service information from the slave DS4302.

Use the returned handle from the `ds4302_can_service_register` function.

Parameters

service Specifies the pointer to the updated `ds4302_canService` structure. The following data will be updated if available: busstatus, stdmask, extmask, msg_mask15, tx_ok, rx_ok, crc_err, ack_err, form_err, stuffbit_err, bit1_err, bit0_err, rx_lost, data_lost, version, mailbox_err, c252s_err, p2in, txqueue_overflowcnt_std, txqueue_overflowcnt_ext.

Return value

This function returns the error code. The following symbols are predefined:

Symbols	Meaning
`DS4302_CAN_NO_ERROR`	The function was performed without error.
`DS4302_CAN_NO_DATA`	No data was updated.
`DS4302_CAN1_DATA_LOST`	The input data of a previous request for the specified function was overwritten.

Example

For an example of how to use this function, refer to Example of Using Service Functions on page 100.

```
ds4302_canService* service;
....
service = (txCh,
          DS4302_CAN_SERVICE_TX_OK |
          DS4302_CAN_SERVICE_TXQUEUE_OVERFLOW_COUNT);
(service);
ds4302_service_read(service);
```

Related topics

References

CAN Subinterrupt Handling

Where to go from here

Information in this section

Defining a Callback Function

Callback function

The callback function is a function that performs the action(s) that you define for a given subinterrupt. The callback function must be installed with the `ds4302_can_subint_handler_install` function.

Each time a CAN subinterrupt occurs, the subinterrupt handling then passes the information to the callback function.

Defining a callback function

Define your callback function as follows:

```
void can_callback_fcn(void* subint_data, Int32 subint);
```

with the parameters

subint_data Pointer to the board index of the related board within the range 0 … 15

subint Subinterrupt number within the range 0 … 30

Note

The last subinterrupt number to be generated is always "-1". This value indicates that there are no more pending subinterrupts.

Related topics

References

ds4302_can_subint_handler_install....91

ds4302_can_subint_handler_install

Syntax

```
ds4302_can_subint_handler_t  ds4302_can_subint_handler_install(
      const UInt32 base,
      const ds4302_can_subint_handler_t handler);
```

Include file

`Ds4302.h`

Purpose

To install a subinterrupt handler for all CAN interrupts.

Parameters

base	Specifies the PHS-bus base address of the DS4302 board.
handler	Specifies the pointer to your callback function.

For information on defining a callback function, refer to Defining a Callback Function on page 90.

Return value

This function returns the following value:

Symbol	Meaning
`ds4302_can_subint_handler_t`	Pointer to the previously installed callback function

Example

For an example of how to use this function, refer to Example of Using Subinterrupts on page 98.

Related topics

Basics

Defining a Callback Function.. 90

Utilities

Introduction

Information on setting the time base to a defined value, clearing CAN data on the master, and reading the current error code.

Where to go from here

Information in this section

ds4302_can_all_data_clear

Syntax

```
void ds4302_can_all_data_clear(const UInt32 base);
```

Include file

`Ds4302.h`

Purpose

To clear the data buffer of the master. This is required by the RTI environment to clear all data when restarting the simulation.

Parameters

base	Specifies the PHS-bus base address of the DS4302 board.

Return value

None

Example

```
ds4302_can_all_data_clear(DS4302_1_BASE)
```

Related topics

References

ds4302_can_msg_clear.....80

ds4302_can_error_read

Syntax

```
Int32 ds4302_can_error_read(
      const UInt32 base,
      const Int32 queue);
```

Include file Ds4302.h

Purpose To read the current error of the slave DS4302 from the dual-port memory.

Parameters

base	Specifies the PHS-bus base address of the DS4302 board.
queue	Specifies the communication channel within the range 0 … 6.

Return value This function returns the error code; the following symbols are predefined:

Predefined Symbol	Meaning
DS4302_CAN_NO_ERROR	No error on the slave DS4302.
DS4302_CAN_SLAVE_ALLOC_ERROR	Memory allocation error on the slave DS4302. There are too many functions registered.
DS4302_CAN_LAVE_BUFFER_OVERFLOW	Not enough memory space between the slave write pointer and the master read pointer.
DS4302_CAN_INIT_ACK	Acknowledge code. This is no error.
DS4302_CAN_SLAVE_UNDEF_ERROR	Undefined error. An error that cannot be assigned to one of the previous errors.

Example

```
#define QUEUE0   0
Int32 slave_error;
slave_error = ds4302_can_error_read(DS4302_1_BASE, QUEUE0);
/*     */
/* error handling */
/*     */
```

Examples of Using CAN

Introduction

Examples of how to use the CAN functions.

Where to go from here

Information in this section

Example of Handling Transmit and Receive Messages

Example

This example shows how to register a transmit and a receive message.

After a delay of 4.0 seconds, the transmit message is sent periodically every 1.0 seconds. If you connect the two CAN channels with each other, you can receive the transmitted CAN message on the other CAN channel. After the CAN message is received successfully, an info message is sent to the message module.

```
#include <Brtenv.h>
#include <Ds4302.h>
ds4302_canChannel* txCh;
ds4302_canChannel* rxCh;
ds4302_canChannel* txCh;
ds4302_canChannel* rxCh;
ds4302_canMsg* txMsg;
ds4302_canMsg* rxMsg;
UInt32 txMsgData[8] = {1,2,3,4,5,6,7,8};
main()
{
  init(); /* initialize hardware system */
  msg_info_set(MSG_SM_RTLIB, 0, "System started.");
  ds4302_init(DS4302_1_BASE);
  ds4302_can_communication_init(DS4302_1_BASE,
                              DS4302_CAN_INT_DISABLE);
  txCh = ds4302_can_channel_init(DS4302_1_BASE, 0,
                    500000,
                    DS4302_CAN_STD,
                    DS4302_CAN_NO_SUBINT);
  ds4302_can_channel_transceiver(txCh,
                            DS4302_CAN_ISO11898_TRANSCEIVER,
                            DS4302_CAN_TERMINATION_ON);
  rxCh = ds4302_can_channel_init(DS4302_1_BASE, 1,
                    500000,
                    DS4302_CAN_STD,
                    DS4302_CAN_NO_SUBINT);
  ds4302_can_channel_transceiver(rxCh,
                            DS4302_CAN_ISO11898_TRANSCEIVER,
                            DS4302_CAN_TERMINATION_ON);
  txMsg = ds4302_can_msg_tx_register(txCh,
                    2,
                    0x123,
                    DS4302_CAN_STD,
                    DS4302_CAN_TIMECOUNT_INFO,
                    DS4302_CAN_NO_SUBINT,
                    4.0,
                    1.0,
                    DS4302_CAN_TIMEOUT_NORMAL);
  rxMsg = ds4302_can_msg_rx_register(rxCh,
                    3,
                    0x123,
                    DS4302_CAN_STD,
                    DS4302_CAN_DATA_INFO |
                    DS4302_CAN_TIMECOUNT_INFO,
                    DS4302_CAN_NO_SUBINT);
  ds4302_can_msg_write(txMsg, 8, txMsgData);
  ds4302_can_channel_start(rxCh, DS4302_CAN_INT_DISABLE);
  ds4302_can_channel_start(txCh, DS4302_CAN_INT_DISABLE);
  for(;;)
  {
    ds4302_can_msg_read(txMsg);
    if (txMsg->processed == DS4302_CAN_PROCESSED)
    {
      msg_info_printf(MSG_SM_RTLIB, 0,
      "TX CAN message, time: %f, deltatime: %f ",
      txMsg->timestamp, txMsg->deltatime);
    }
    ds4302_can_msg_read(rxMsg);
    if (rxMsg->processed == DS4302_CAN_PROCESSED)
    {
      msg_info_printf(MSG_SM_RTLIB, 0,
      "RX CAN message, time: %f,deltatime: %f ",
      rxMsg->timestamp, rxMsg->deltatime);
    }
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

Examples

Example of Handling Request and Remote Messages

Example

This example shows how to register a request and a remote message.

After a delay of 4.0 seconds, the request message is sent periodically every 2.0 seconds. If you connect the two CAN channels with each other you can receive the request message on the other CAN channel. After the requested data is received successfully, an info message is sent to the message module.

```c
#include <Brtenv.h>
#include <Ds4302.h>

ds4302_canChannel* rqCh;
ds4302_canChannel* rmCh;
ds4302_canMsg* rqtxMsg;
ds4302_canMsg* rqrxMsg;
ds4302_canMsg* rmMsg;
UInt32 rmMsgData[8] = {1,2,3,4,5,6,7,8};

main()
{
   init(); /* initialize hardware system */

   ds4302_init(DS4302_1_BASE);

   ds4302_can_communication_init(DS4302_1_BASE,
                  DS4302_CAN_INT_DISABLE);

   rqCh = ds4302_can_channel_init(DS4302_1_BASE, 0,
                  500000,
                  DS4302_CAN_STD,
                  DS4302_CAN_NO_SUBINT);

   rmCh = ds4302_can_channel_init(DS4302_1_BASE, 1,
                  500000,
                  DS4302_CAN_STD,
                  DS4302_CAN_NO_SUBINT);

   ds4302_can_channel_transceiver(rqCh,
                  DS4302_CAN_ISO11898_TRANSCEIVER,
                  DS4302_CAN_TERMINATION_ON);

   ds4302_can_channel_transceiver(rmCh,
                  DS4302_CAN_ISO11898_TRANSCEIVER,
                  DS4302_CAN_TERMINATION_ON);

   rqtxMsg = ds4302_can_msg_rqtx_register(rqCh,
                  2,
                  0x123,
                  DS4302_CAN_STD,
                  DS4302_CAN_TIMECOUNT_INFO,
                  DS4302_CAN_NO_SUBINT,
                  4.0,
                  2.0,
                  DS4302_CAN_TIMEOUT_NORMAL);

   rqrxMsg = ds4302_can_msg_rqrx_register(rqtxMsg,
                  DS4302_CAN_DATA_INFO | DS4302_CAN_TIMECOUNT_INFO,
                  DS4302_CAN_NO_SUBINT);

   rmMsg = ds4302_can_msg_rm_register(rmCh,
                  3,
                  0x123,
                  DS4302_CAN_STD,
                  DS4302_CAN_TIMECOUNT_INFO,
                  DS4302_CAN_NO_SUBINT);

   ds4302_can_msg_write(rmMsg, 8, rmMsgData);

   ds4302_can_msg_rqtx_activate(rqtxMsg);

   ds4302_can_channel_start(rqCh, DS4302_CAN_INT_DISABLE);

   ds4302_can_channel_start(rmCh, DS4302_CAN_INT_DISABLE);

   for(;;)
   {
      host_service(0,0);
      master_cmd_server();
      ds4302_can_msg_read(rqrxMsg);
      ds4302_can_msg_read(rqtxMsg);
      ds4302_can_msg_read(rmMsg);

      if (rqrxMsg->processed == DS4302_CAN_PROCESSED)
      {
         msg_info_printf(MSG_SM_RTLIB, 0,
                  "RQRX CAN message, time: %f,deltatime: %f ",
         rqrxMsg->timestamp, rqrxMsg->deltatime);
      }

      if (rqtxMsg->processed == DS4302_CAN_PROCESSED)
      {
         msg_info_printf(MSG_SM_RTLIB, 0,
                  "RQTX CAN message, time: %f,deltatime: %f ",
         rqtxMsg->timestamp, rqtxMsg->deltatime);
      }

      if (rmMsg->processed == DS4302_CAN_PROCESSED)
      {
         msg_info_printf(MSG_SM_RTLIB, 0,
               "RM   CAN message, time: %f,deltatime: %f ",
               rmMsg->timestamp, rmMsg->deltatime);
      }

   }
}
```

Related topics

Examples

Example of Using Subinterrupts

Example

This example shows how to register messages that can generate a subinterrupt.

The CAN controller is started and a CAN message is sent immediately. If the CAN message was sent successfully, a subinterrupt is generated to call the installed callback function.

The callback function in this example evaluates the specified subinterrupt and sends the CAN message again with a time delay of 0.1 s.

After the CAN message is received, another subinterrupt is generated to read the CAN message and pass an info message to the message module.

Note

The CAN channels 0 and 1 have to be connected.

```c
#include <Brtenv.h>
#include <Ds4302.h>
#define tx_subint 2
#define rx_subint 3
ds4302_canChannel* txCh;
ds4302_canChannel* rxCh;
ds4302_canMsg* txMsg;
ds4302_canMsg* rxMsg;
UInt32 txMsgData[8] = { 1,2,3,4,5,6,7,8 };
void can_user_callback(void* subint_data, Int32 subint)
{
  switch(subint)
  {
    case tx_subint:
      txMsgData[0] = (txMsgData[0]+1) & 0xFF;
      /* send the message delayed */
      ds4302_can_msg_send( txMsg, 8, txMsgData, 0.1);
      msg_info_printf(MSG_SM_RTLIB, 0, "TX Subint:%d",
                      subint);
    break;
    case rx_subint:
      /* read the message from the communication buffer */
      ds4302_can_msg_read(rxMsg);
      msg_info_printf(MSG_SM_RTLIB, 0,
       "RX Subint:%d, time: %fs, deltatime: %fs data[0]: %x",
                       subint,
                       rxMsg->timestamp,
                       rxMsg->deltatime,
                       rxMsg->data[0]);
    break;
    default:
    break;
  }
}
main()
{
   init(); /* initialize hardware system */
   ds4302_init(DS4302_1_BASE);
   ds4302_can_communication_init(DS4302_1_BASE,
                                 DS4302_CAN_INT_ENABLE);
   ds4302_can_subint_handler_install(DS4302_1_BASE,
                                 can_user_callback);
   txCh = ds4302_can_channel_init (DS4302_1_BASE, 1,
                                  500000, DS4302_CAN_STD,
                                   DS4302_CAN_NO_SUBINT);
   ds4302_can_channel_transceiver(txCh,
                            DS4302_CAN_ISO11898_TRANSCEIVER,
                             DS4302_CAN_TERMINATION_ON);
   txMsg = ds4302_can_msg_tx_register(txCh,
                                  0,
                                  0x123,
                                  DS4302_CAN_STD,
                                  DS4302_CAN_NO_INFO,
                                  tx_subint,
                                  0.0,
                                  0.0,
                                  DS4302_CAN_TIMEOUT_NORMAL);
   rxCh = ds4302_can_channel_init(DS4302_1_BASE,
                                 0,
                                 500000,
                                 DS4302_CAN_STD,
                                 DS4302_CAN_NO_SUBINT);
   ds4302_can_channel_transceiver(rxCh,
                            DS4302_CAN_ISO11898_TRANSCEIVER,
                             DS4302_CAN_TERMINATION_ON);
   rxMsg = ds4302_can_msg_rx_register(rxCh,
                                 0,
                                 0x123,
                                 DS4302_CAN_STD,
                                 DS4302_CAN_DATA_INFO |
                                 DS4302_CAN_TIMECOUNT_INFO,
                                 rx_subint);
   ds4302_can_channel_start(rxCh, DS4302_CAN_INT_DISABLE);
   ds4302_can_channel_start(txCh, DS4302_CAN_INT_DISABLE);
   ds4302_can_msg_send( txMsg, 8, txMsgData, 0.0);
   RTLIB_INT_ENABLE();
   for(;;)
   {
     RTLIB_BACKGROUND_SERVICE();
   }
}
```

Related topics

Examples

Example of Using Service Functions

Example

This example shows how to use the service functions `DS4302_CAN_SERVICE_TX_OK` and `DS4302_CAN_SERVICE_RX_OK`.

Note

No message is installed on the DS4302 in this example.

```
#include <Brtenv.h>
#include <Ds4302.h>
ds4302_canChannel* canCh0;
ds4302_canChannel* canCh1;
ds4302_canChannel* txokServ;
ds4302_canService* rxokServ;
main()
{
  init();
  ds4302_init(DS4302_1_BASE);
  ds4302_can_communication_init(DS4302_1_BASE,
              DS4302_CAN_INT_DISABLE);
  canCh0 = ds4302_can_channel_init(DS4302_1_BASE,0,
              500000,    DS4302_CAN_STD,
              DS4302_CAN_NO_SUBINT);
  ds4302_can_channel_transceiver(canCh0,
                 DS4302_CAN_ISO11898_TRANSCEIVER,
                 DS4302_CAN_TERMINATION_ON);
  canCh1 = ds4302_can_channel_init(DS4302_1_BASE, 1,
           500000, DS4302_CAN_STD, DS4302_CAN_NO_SUBINT);
  ds4302_can_channel_transceiver(canCh1,
                DS4302_CAN_ISO11898_TRANSCEIVER,
                 DS4302_CAN_TERMINATION_ON);
  ds4302_can_channel_start(canCh0, DS4302_CAN_INT_ENABLE);
  ds4302_can_channel_start(canCh1, DS4302_CAN_INT_ENABLE);
  /* register the tx_ok function which delivers the */
  /* count of the tx-ok counter for CAN channel 0 */
  txokServ = ds4302_can_service_register(canCh0,
              DS4302_CAN_SERVICE_TX_OK);
  /* register the rx_ok function which delivers the */
  /* count of the rx-ok counter for CAN channel 1 */
  rxokServ = ds4302_can_service_register(canCh1,
             DS4302_CAN_SERVICE_RX_OK);
  for(;;)
  {
     /* request the tx-ok counter from the slave DS4302 */
```

```
    ds4302_can_service_request(txokServ);
    /* request the rx-ok counter from the slave DS4302 */
    ds4302_can_service_request(rxokServ);
    /* read the tx-ok counter from the slave DS4302 */
    /* the data will be available in txokServ->data0 */
    ds4302_can_service_read(txokServ);
    /* read the rx-ok counter from the slave DS4302 */
    /* the data will be available in rxokServ->data0 */
    ds4302_can_service_read(rxokServ);
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

Examples

Example of Receiving Different Message IDs

Example

This example shows you how to set up a CAN controller to receive the message IDs 0x100 … 0x1FF via one message queue.

```
#include <Brtenv.h>
#include <Ds4302.h>
ds4302_canChannel* rxCh;
ds4302_canMsg* canMonitor;
UInt32 data[8];
UInt32 mask = 0x1FFFFF00;
UInt32 queue_size = 64;
main()
{
  init();
  ds4302_init(DS4302_1_BASE);
  ds4302_can_communication_init(DS4302_1_BASE,
           DS4302_CAN_INT_DISABLE);
  rxCh = ds4302_can_channel_init(DS4302_1_BASE, 0,
              500000, DS4302_CAN_STD,
              DS4302_CAN_NO_SUBINT);
  ds4302_can_channel_transceiver(rxCh,
              DS4302_CAN_ISO11898_TRANSCEIVER,
              DS4302_CAN_TERMINATION_ON);
  canMonitor = ds4302_can_msg_rx_register (rxCh, 1,
                        0x100, DS4302_CAN_STD,
                        DS4302_CAN_TIMECOUNT_INFO | DS4302_CAN_MSG_INFO,
                        DS4302_CAN_NO_SUBINT);
  ds4302_can_msg_set(canMonitor, DS4302_CAN_MSG_MASK,
                   &mask);
  ds4302_can_msg_set(canMonitor, DS4302_CAN_MSG_QUEUE,
                   &queue_size);
  ds4302_can_channel_start(rxCh, DS4302_CAN_INT_DISABLE);
  for(;;)
  {
    ds4302_can_msg_read(canMonitor);
    if (canMonitor->processed == DS4302_CAN_PROCESSED)
    {
      msg_info_printf(0,0, "id: %d time: %f",
      canMonitor->identifier, canMonitor->timestamp);
    }
    RTLIB_BACKGROUND_SERVICE();
  }
}
```

Related topics

Examples