Your embedded solution partner.

<< Reference Manual <<

CANopen Stack

CANopen Library

2.7.3

Generated by Doxygen 1.8.11

Contents

Chapter 1

CANopen Stack Reference Manual

1.1 Introduction

The CANopen Slave Stack of emtas is a software library that provides all communication services of the "CANopen Application Layer and Communication Profile" CiA 301 V4.2 and other profiles of CiA e.V. and EN50325-4.

The main features are:

- well-defined interface between driver and CANopen stack
- ANSI-C conform
- MISRA checked
- easy-to-handle Application Programming Interface
- static and dynamic object dictionary are possible
- LED CiA-303
- Layer Setting Services (CiA 305),
- configurable and scalable
- extensions for additional communication profiles such as
  - redundant communication (CiA 302),
  - safety relevant communication (CiA 304) as well as device profile implementations like
  - Generic I/O Modules (CiA 401)
  - EnergyBus Protokoll (CiA 454) are available.

This reference manual describes the functions for the API to evaluate the received data and to use the CANopen services in the network.

Configuration and features settings are supported by the graphical configuration tool CANopen DeviceDesigner.

1.2 General

The CANopen stack use strict data hiding, so access to internal data are only possible by functions. The same is valid for access to the communication segment of the object dictionary.

1.3 Using CANopen stack in an application

At startup, some initialization functions are necessary:

- codrvHardwareInit() - generic, CAN related hardware initialization
- codrvCanInit() - initialize CAN driver
- coCanOpenStackInit() - initialize CANopen functionality
- codrvTimerSetup() - initialize hardware timer
- codrvCanEnable() - start CAN communication

For the CANopen functionality, the central function coCommTask() has to be called in case of

- new CAN message was received
- timer period has been ellapsed.

Therefore signal handlers should be used or a cyclic call of the function coCommTask() is necessary. For operating systems (like LINUX) the function codrvWaitForEvent() can be used to wait for events.
All CANopen functionality is handled inside this function.

The start of CANopen services are also possible.

1.4 Indication functions

Indication functions inform application about CAN and CANopen service events.
To receive an indication, the application has to register a function by the apropriate service register function like coEventRegister_PDO().
Every time the event occures, the registered indication function is called.

Chapter 2

Data Structure Index

2.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 3

File Index

3.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 4

Data Structure Documentation

4.1 CO_CAN_COB_T Struct Reference

Data Fields

- UNSIGNED32 canId
- UNSIGNED32 ignore
- UNSIGNED16 canChan
- BOOL_T extended
- BOOL_T rtr
- BOOL_T enabled

4.1.1 Detailed Description

CAN cob structure

4.1.2 Field Documentation

4.1.2.1 UNSIGNED16 canChan

reserved for driver

4.1.2.2 UNSIGNED32 canId

can identifier

4.1.2.3 BOOL_T enabled

cob enabled/disabled

4.1.2.4 BOOL_T extended

extended id

4.1.2.5 UNSIGNED32 ignore

ignore mask for id

4.1.2.6 BOOL_T rtr

rtr

The documentation for this struct was generated from the following file:

- co_drv.h

4.2 CO_CAN_MSG_T Struct Reference

Data Fields

- LIBDRV_HANDLE_T handle
- CO_CAN_COB_T canCob
- UNSIGNED8 len
- UNSIGNED8 data [CO_CAN_MAX_DATA_LEN]

4.2.1 Detailed Description

CAN message structure

4.2.2 Field Documentation

4.2.2.1 CO_CAN_COB_T canCob

cob infos

4.2.2.2 UNSIGNED8 data[CO_CAN_MAX_DATA_LEN]

data

4.2.2.3 LIBDRV_HANDLE_T handle

library internal handle

4.2.2.4 UNSIGNED8 len

msg len

The documentation for this struct was generated from the following file:

- co_drv.h

4.3 CO_SERVICE_INIT_VAL_T Struct Reference

4.3.1 Detailed Description

line parameter definition

defines number of line parameter for services

The documentation for this struct was generated from the following file:

- co_canopen.h

4.4 CO_TIME_T Struct Reference

Data Fields

- UNSIGNED32 msec
- UNSIGNED16 days

4.4.1 Detailed Description

TIME_OF_DAY structure

4.4.2 Field Documentation

4.4.2.1 UNSIGNED16 days

days after 1st january of 1984

4.4.2.2 UNSIGNED32 msec

milliseconds after midnight

The documentation for this struct was generated from the following file:

- co_time.h

4.5 co_timer Struct Reference

Data Fields

- struct co_timer ∗ pNext
- UNSIGNED32 actTicks
- UNSIGNED32 ticks
- CO_TIMER_FCT_T pFct
- void ∗ pData
- CO_TIMER_ATTR_T attr

4.5.1 Detailed Description

timer structure

4.5.2 Field Documentation

4.5.2.1 UNSIGNED32 actTicks

actual timer ticks

4.5.2.2 CO_TIMER_ATTR_T attr

timer attributes

4.5.2.3 void∗ pData

pointer for own data

4.5.2.4 CO_TIMER_FCT_T pFct

pointer to own function

4.5.2.5 struct co_timer∗ pNext

pointer to next timer

4.5.2.6 UNSIGNED32 ticks

calculated timer ticks

The documentation for this struct was generated from the following file:

- co_timer.h

4.6 PDO_REC_MAP_ENTRY_T Struct Reference

Data Fields

- void ∗ pVar
- UNSIGNED8 len
- BOOL_T numeric
- UNSIGNED32 val
- UNSIGNED16 routePdo [1]

4.6.1 Detailed Description

PDO receive mapping entry (one mapping entry)

4.6.2 Field Documentation

4.6.2.1 UNSIGNED8 len

number of bytes for variable

4.6.2.2 BOOL_T numeric

numeric flag for byte swapping

4.6.2.3 void∗ pVar

pointer to variable

4.6.2.4 UNSIGNED16 routePdo[1]

route to other network

4.6.2.5 UNSIGNED32 val

OD value

The documentation for this struct was generated from the following file:

- co_pdo.h

4.7 PDO_REC_MAP_TABLE_T Struct Reference

Data Fields

- UNSIGNED8 mapCnt
- PDO_REC_MAP_ENTRY_T mapEntry [CO_MAX_MAP_ENTRIES]

4.7.1 Detailed Description

PDO mapping table (mapping entries for one receive PDO)

4.7.2 Field Documentation

4.7.2.1 UNSIGNED8 mapCnt

number of mapping entries

4.7.2.2 PDO_REC_MAP_ENTRY_T mapEntry[CO_MAX_MAP_ENTRIES]

Mapping entries

The documentation for this struct was generated from the following file:

- co_pdo.h

4.8 PDO_TR_MAP_ENTRY_T Struct Reference

Data Fields

- CO_CONST void ∗ pVar
- UNSIGNED8 len
- BOOL_T numeric
- UNSIGNED32 val

4.8.1 Detailed Description

PDO transmit mapping entry (one mapping entry)

4.8.2 Field Documentation

4.8.2.1 UNSIGNED8 len

number of bytes for variable

4.8.2.2 BOOL_T numeric

numeric flag for byte swapping

4.8.2.3 CO_CONST void∗ pVar

pointer to variable

4.8.2.4 UNSIGNED32 val

OD value

The documentation for this struct was generated from the following file:

- co_pdo.h

4.9 PDO_TR_MAP_TABLE_T Struct Reference

Data Fields

- UNSIGNED8 mapCnt
- PDO_TR_MAP_ENTRY_T mapEntry [CO_MAX_MAP_ENTRIES]

4.9.1 Detailed Description

PDO mapping table (mapping entries for one transmit PDO)

4.9.2 Field Documentation

4.9.2.1 UNSIGNED8 mapCnt

number of mapping entries

4.9.2.2 PDO_TR_MAP_ENTRY_T mapEntry[CO_MAX_MAP_ENTRIES]

Mapping entries

The documentation for this struct was generated from the following file:

- co_pdo.h

Chapter 5

File Documentation

5.1 co_candebug.c File Reference

CAN debug functionality.

5.1.1 Detailed Description

CAN debug functionality.

Contain functions to send any data over CAN

5.2 co_candebug.h File Reference

defines for can debug

5.2.1 Detailed Description

defines for can debug

- contains defines for can debug services

5.3 co_canopen.h File Reference

defines for all services

Data Structures

- struct CO_SERVICE_INIT_VAL_T

Functions

- EXTERN_DECL RET_T coCanOpenStackInit (CO_EVENT_STORE_T pLoadFunction)

  coCanOpenStackInit - init of CANopen stack
- EXTERN_DECL RET_T coCanOpenStackInitPara (CO_EVENT_STORE_T pLoadFunction, CO_INIT_OP↩ TION_T ∗pCoOptions)

  coCanOpenStackInit - init of CANopen stack This function is normally generated by the CANopen Device Designer and responsible for the intialization of the CANopen stack. In addition to coCanOpenStackInit some options for services can be added.
- RET_T coCanOpenStackInit_common (CO_EVENT_STORE_T pLoadFunction)

  coCanOpenStackInit_common - init of common part of CANopen stack This function is generated by the CANopen Device Designer and responsible for the common intialization of the CANopen stack. Normally called from coCan↩ OpenStackInit();
- RET_T coCanOpenStackInit_line (CO_INIT_OPTION_T ∗pCoOptions)

  coCanOpenStackInit_line - init one CAN line of CANopen stack This function is generated by the CANopen Device Designer and responsible for the line depending intialization of the CANopen stack. Normally called from coCan↩ OpenStackInit();
- EXTERN_DECL void coCanOpenStackDeInit (void)

  coCanOpenStackDeInit - deinit of CANopen stack
- EXTERN_DECL void coCanOpenStackVarInit (CO_SERVICE_INIT_VAL_T ∗pServiceInitVals)

  coCanOpenStackVarInit - init of variables of the stack

5.3.1 Detailed Description

defines for all services

- contains defines for all services

This header inludes defines for all services of the CANopen library. It can be included instead of header files of each service.

5.3.2 Function Documentation

5.3.2.1 EXTERN_DECL void coCanOpenStackDeInit (void)

coCanOpenStackDeInit - deinit of CANopen stack

This function is normally generated by the CANopen Device Designer and responsible for the de-intialization of the CANopen stack.

Returns

void

5.3.2.2 EXTERN_DECL RET_T coCanOpenStackInit (CO_EVENT_STORE_T *pLoadFunction*)

coCanOpenStackInit - init of CANopen stack

This function is normally generated by the CANopen Device Designer and responsible for the whole intialization of the CANopen stack.

Parameters

pLoadFunction	pointer to loadFunction

Returns

RET_T

5.3.2.3 RET_T coCanOpenStackInit_common (CO_EVENT_STORE_T *pLoadFunction*)

coCanOpenStackInit_common - init of common part of CANopen stack This function is generated by the CA↩ Nopen Device Designer and responsible for the common intialization of the CANopen stack. Normally called from coCanOpenStackInit();

Parameters

pLoadFunction	pointer to loadFunction

Returns

RET_T

5.3.2.4 RET_T coCanOpenStackInit_line (CO_INIT_OPTION_T ∗ *pCoOptions*)

coCanOpenStackInit_line - init one CAN line of CANopen stack This function is generated by the CANopen Device Designer and responsible for the line depending intialization of the CANopen stack. Normally called from coCan↩ OpenStackInit();

Parameters

pCoOptions	pointer to coOptions

Returns

RET_T

5.3.2.5 EXTERN_DECL RET_T coCanOpenStackInitPara (CO_EVENT_STORE_T *pLoadFunction,* CO_INIT_OPTION_T ∗ *pCoOptions*)

coCanOpenStackInit - init of CANopen stack This function is normally generated by the CANopen Device Designer and responsible for the intialization of the CANopen stack. In addition to coCanOpenStackInit some options for services can be added.

Parameters

pLoadFunction	pointer to loadFunction
pCoOptions	pointer to coOptions

Returns

RET_T

5.3.2.6 EXTERN_DECL void coCanOpenStackVarInit (CO_SERVICE_INIT_VAL_T ∗ *pServiceInitVals*)

coCanOpenStackVarInit - init of variables of the stack

This function initializes all global and local variables of the stack.

It can also be used to reinitialize the stack.

Returns

nothing

Parameters

pServiceInitVals	pointer to init vals

5.4 co_cfgman.c File Reference

config manager handling

Functions

- RET_T coCfgStart (UNSIGNED8 sdoNr, UNSIGNED8 srvNodeId, UNSIGNED8 ∗pBuf, UNSIGNED32 buf←Len, UNSIGNED32 sdoTimeOut)

  co_cfgStart - start configuration
- RET_T coCfgConvToConsive (CHAR ∗pDcfData, UNSIGNED8 ∗pConsBuf, UNSIGNED32 ∗pConsBufLen)

  co_convertToConsiceDcf - convert to consice DCF
- RET_T coEventRegister_CFG_MANAGER (CO_EVENT_CFG_MANAGER_T pFunction)

  coEventRegister_CFG_MAN - register CFG_MAN event

5.4.1 Detailed Description

config manager handling

contains configuration manager handling

5.4.2 Function Documentation

5.4.2.1 RET_T coCfgConvToConsive (CHAR ∗ *pDcfData*, UNSIGNED8 ∗ *pConsBuf*, UNSIGNED32 ∗ *pConsBufLen*)

co_convertToConsiceDcf - convert to consice DCF

This function convert the given data to the consive DCF. At function call the parameter pConsBufLen contains the maximal buffer length, and is updated with the real len of written buffer.

Returns

RET_T

Parameters

pDcfData	pointer to DCF data
pConsBuf	pointer to consive DCF buffer
pConsBufLen	max len of consive DCF buffer

5.4.2.2 RET_T coCfgStart (UNSIGNED8 *sdoNr*, UNSIGNED8 *srvNodeId*, UNSIGNED8 ∗ *pBuf*, UNSIGNED32 *bufLen*, UNSIGNED32 *sdoTimeOut*)

co_cfgStart - start configuration

This function starts the SDO transfer to setup a node with a new configuration. Parameter are given as consive DCF buffer. For the SDO transfer, the client with sdoNr is used. If parameter srvNodeId != 0, then the sdo channel is automatically configured with the default server sdo cobs for the given nodeId.

If transfer is started successful, the function returns RET_OK. Finish of the whole transfer is indicated by the function configured by coEventRegister_CFG_MANAGER().

Returns

RET_T

Parameters

sdoNr	use sdo number
srvNodeId	write to node n
pBuf	pointer to consive dcf buffer
bufLen	len of consive dcf buffer
sdoTimeOut	SDO timeout in msec

5.4.2.3 RET_T coEventRegister_CFG_MANAGER (CO_EVENT_CFG_MANAGER_T *pFunction*)

coEventRegister_CFG_MAN - register CFG_MAN event

This function registers an indication function for CFG_MAN events. The indication function is called after transfer to slave has been finished

Returns

RET_T

Parameters

pFunction	pointer to function

5.5 co_cfgman.h File Reference

defines for config manager services

Typedefs

- typedef void(∗ CO_EVENT_CFG_MANAGER_T) (CO_CFG_TRANSFER_T, UNSIGNED8, UNSIGNED16, UNSIGNED8, UNSIGNED32)

    function pointer to SDO server event

Enumerations

Functions

- EXTERN_DECL RET_T coCfgStart (UNSIGNED8 sdoNr, UNSIGNED8 srvNodeId, UNSIGNED8 ∗pBuf, U←NSIGNED32 bufLen, UNSIGNED32 sdoTimeOut)

    co_cfgStart - start configuration
- EXTERN_DECL RET_T coCfgConvToConsive (char ∗pDcfData, UNSIGNED8 ∗pConsBuf, UNSIGNED32 ∗pConsBufLen)

    co_convertToConsiceDcf - convert to consice DCF
- EXTERN_DECL RET_T coEventRegister_CFG_MANAGER (CO_EVENT_CFG_MANAGER_T pFct)

    coEventRegister_CFG_MAN - register CFG_MAN event

5.5.1 Detailed Description

defines for config manager services

- contains defines for cfgman services

5.5.2 Typedef Documentation

5.5.2.1 typedef void(∗ CO_EVENT_CFG_MANAGER_T) (CO_CFG_TRANSFER_T, UNSIGNED8, UNSIGNED16, UNSIGNED8, UNSIGNED32)

function pointer to SDO server event

Parameters

type	- result type
sdoNr	- sdo number
index	- object index
subindex	- object subindex
reason	- error reason

Returns

void

5.5.3 Enumeration Type Documentation

5.5.3.1 enum CO_CFG_TRANSFER_T

CO_CFG_TRANSFER_T state

Enumerator

CO_CFG_TRANSFER_FINISHED	transfer finished ok
CO_CFG_TRANSFER_ABORT	transfer abort by SDO server
CO_CFG_TRANSFER_ERROR	transfer error by start SDO client

5.5.4 Function Documentation

5.5.4.1 EXTERN_DECL RET_T coCfgConvToConsive (CHAR ∗ *pDcfData,* UNSIGNED8 ∗ *pConsBuf,* UNSIGNED32 ∗ *pConsBufLen*)

co_convertToConsiceDcf - convert to consice DCF

This function convert the given data to the consive DCF. At function call the parameter pConsBufLen contains the maximal buffer length, and is updated with the real len of written buffer.

Returns

RET_T

Parameters

pDcfData	pointer to DCF data
pConsBuf	pointer to consive DCF buffer
pConsBufLen	max len of consive DCF buffer

5.5.4.2 EXTERN_DECL RET_T coCfgStart (UNSIGNED8 *sdoNr,* UNSIGNED8 *srvNodeId,* UNSIGNED8 ∗ *pBuf,* UNSIGNED32 *bufLen,* UNSIGNED32 *sdoTimeOut*)

co_cfgStart - start configuration

This function starts the SDO transfer to setup a node with a new configuration. Parameter are given as consive DCF buffer. For the SDO transfer, the client with sdoNr is used. If parameter srvNodeId != 0, then the sdo channel is automatically configured with the default server sdo cobs for the given nodeId.

If transfer is started successful, the function returns RET_OK. Finish of the whole transfer is indicated by the function configured by coEventRegister_CFG_MANAGER().

Returns

RET_T

Parameters

sdoNr	use sdo number
srvNodeId	write to node n
pBuf	pointer to consive dcf buffer
bufLen	len of consive dcf buffer
sdoTimeOut	SDO timeout in msec

5.5.4.3 EXTERN_DECL RET_T coEventRegister_CFG_MANAGER (CO_EVENT_CFG_MANAGER_T *pFunction*)

coEventRegister_CFG_MAN - register CFG_MAN event

This function registers an indication function for CFG_MAN events. The indication function is called after transfer to slave has been finished

Returns

RET_T

Parameters

pFunction	pointer to function

5.6 co_cob.h File Reference

cob defines

Macros

- #define CO_COB_INVALID 0x80000000UL
- #define CO_COB_29BIT 0x20000000UL
- #define CO_COB_VALID_MASK 0x80000000UL
- #define CO_COB_29BIT_MASK 0x20000000UL
- #define CO_COB_ID_MASK 0x1FFFFFFFUL

5.6.1 Detailed Description

cob defines

- contains defines for cobs

5.6.2 Macro Definition Documentation

5.6.2.1 #define CO_COB_29BIT 0x20000000UL

COB 29bit flag if this bit is set, the COB-ID is a 29-bit identifier

5.6.2.2 #define CO_COB_29BIT_MASK 0x20000000UL

COB 29bit mask With this mask, cobs can be checked for 29bit identifier

5.6.2.3 #define CO_COB_ID_MASK 0x1FFFFFFFUL

COB ID mask With this mask, only identifier bits are masked

5.6.2.4 #define CO_COB_INVALID 0x80000000UL

COB invalid if this bit is set, the COB-ID (and the service) is invalid

5.6.2.5 #define CO_COB_VALID_MASK 0x80000000UL

cob valid mask With this mask, cobs can be checked for valid

5.7 co_cobhandler.c File Reference

Functions for COB handling.

5.7.1 Detailed Description

Functions for COB handling.

contains functions for cob handling

5.8 co_commtask.c File Reference

communication task routines

Functions

- void coCommTask (void)

  coCommTask - main communication task
- BOOL_T coCommTaskCheck (void)

  coCommTaskCheck - check communication tasks
- void coCommStateEvent (CO_COMM_STATE_EVENT_T newEvent)

  coCommStateEvent - set a new communication state
- RET_T coEventRegister_CAN_STATE (CO_EVENT_CAN_STATE_T pFunction)

  coEventRegister_CAN_STATE - register can state changes
- RET_T coEventRegister_COMM_EVENT (CO_EVENT_COMM_T pFunction)

  coEventRegister_COMM_EVENT - register communication event changes

5.8.1 Detailed Description

communication task routines

contains communication task functions of canopen library

5.8.2 Function Documentation

5.8.2.1 void coCommStateEvent (CO_COMM_STATE_EVENT_T *newEvent*)

coCommStateEvent - set a new communication state

This function should be called, if a new communication state has been reached.
It sets the LEDs and informs the application about the event.

Returns

void

Parameters

newEvent	new communication event

5.8.2.2 void coCommTask (void)

coCommTask - main communication task

This is the main communication task for the CANopen stack. It has to be called cyclically by the application or signal driven after each received CAN message or timer event.

Returns

void

5.8.2.3 BOOL_T coCommTaskCheck (void)

coCommTaskCheck - check communication tasks

This function checks if all communication task are done. If not, it returns true. In this case, coCommTask() should be called again.

Returns

BOOL_T

Return values

CO_TRUE	further actions necessary - call coCommTask again
CO_FALSE	all actions passed

5.8.2.4 RET_T coEventRegister_CAN_STATE (CO_EVENT_CAN_STATE_T *pFunction*)

coEventRegister_CAN_STATE - register can state changes

With this function the application can register a function which is called, when the CAN state was changed.
CAN states are:

- BUS_OFF
- BUS_ON
- PASSIV
- UNCHANGED

Returns

RET_T

Parameters

pFunction	pointer to function

5.8.2.5 RET_T coEventRegister_COMM_EVENT (CO_EVENT_COMM_T *pFunction*)

coEventRegister_COMM_EVENT - register communication event changes

With this function the application can register a function which is called, when the communication state has been changed.

- BUS OFF - no communication possible
- CAN OVERRUN - messages was lost
- Receice queue full - receive messages is full
- Receice queue overrun - receive messages was lost
- Transmit queue full - no more messages can be send
- Transmit queue overflow - transmit messages was lost
- Transmit queue empty - new transmit messages can be send

Returns

RET_T

Parameters

pFunction	pointer to function

5.9 co_commtask.h File Reference

defines for communication services

Typedefs

- typedef void(∗ CO_EVENT_CAN_STATE_T) (CO_CAN_STATE_T)

  function pointer to CAN state indication function
- typedef void(∗ CO_EVENT_COMM_T) (CO_COMM_STATE_EVENT_T)

  function pointer to Communication state event indication function

Enumerations

Functions

- EXTERN_DECL void coCommTask (void)

  coCommTask - main communication task
- EXTERN_DECL BOOL_T coCommTaskCheck (void)

  coCommTaskCheck - check communication tasks
- EXTERN_DECL void coCommStateEvent (CO_COMM_STATE_EVENT_T newEvent)

  coCommStateEvent - set a new communication state
- EXTERN_DECL RET_T coEventRegister_COMM_EVENT (CO_EVENT_COMM_T pFunction)

  coEventRegister_COMM_EVENT - register communication event changes
- EXTERN_DECL RET_T coEventRegister_CAN_STATE (CO_EVENT_CAN_STATE_T pFunction)

  coEventRegister_CAN_STATE - register can state changes
- EXTERN_DECL void coQueueInit (void)

  coQueueInit - (re)init queues

5.9.1 Detailed Description

defines for communication services

- contains defines for communication services of the CANopen library

5.9.2 Typedef Documentation

5.9.2.1 typedef void(∗ CO_EVENT_CAN_STATE_T) (CO_CAN_STATE_T)

function pointer to CAN state indication function

Parameters

canState	- new CAN state

Provides a new CAN controller state like Bus on, Bus off, error passive

Returns

void

5.9.2.2 typedef void(∗ CO_EVENT_COMM_T) (CO_COMM_STATE_EVENT_T)

function pointer to Communication state event indication function

Parameters

commState	- new communication state

Provides new communication states like buffer state, CAN working state CO_COMM_STATE_EVENT_REC←↩_QUEUE_FULL CO_COMM_STATE_EVENT_REC_QUEUE_OVERFLOW CO_COMM_STATE_EVENT_REC_←↩QUEUE_EMPTY CO_COMM_STATE_EVENT_TR_QUEUE_FULL CO_COMM_STATE_EVENT_TR_QUEUE_←↩OVERFLOW CO_COMM_STATE_EVENT_TR_QUEUE_EMPTY CO_COMM_STATE_EVENT_CAN_OVERRUN CAN controller states are only signaled by CO_EVENT_CAN_STATE_T

Returns

void

5.9.3 Enumeration Type Documentation

5.9.3.1 enum CO_CAN_STATE_T

CAN states

Enumerator

CO_CAN_STATE_BUS_OFF	CAN bus state is bus off
CO_CAN_STATE_BUS_ON	CAN bus state is bus on
CO_CAN_STATE_PASSIVE	CAN bus state is passive
CO_CAN_STATE_UNCHANGED	CAN bus state is unchanged

5.9.3.2 enum CO_COMM_STATE_EVENT_T

Communication state events

Enumerator

CO_COMM_STATE_EVENT_NONE	no event
CO_COMM_STATE_EVENT_BUS_OFF	bus off
CO_COMM_STATE_EVENT_BUS_OFF_RECOVERY	recvovery from bus off
CO_COMM_STATE_EVENT_BUS_ON	bus on
CO_COMM_STATE_EVENT_PASSIVE	can passive
CO_COMM_STATE_EVENT_ACTIVE	can active
CO_COMM_STATE_EVENT_CAN_OVERRUN	can overrun
CO_COMM_STATE_EVENT_REC_QUEUE_FULL	receice queue full
CO_COMM_STATE_EVENT_REC_QUEUE_OVERFLOW	receive queue overflow
CO_COMM_STATE_EVENT_REC_QUEUE_EMPTY	receice queue empty
CO_COMM_STATE_EVENT_TR_QUEUE_FULL	transmit queue full
CO_COMM_STATE_EVENT_TR_QUEUE_OVERFLOW	transmit queue overflow
CO_COMM_STATE_EVENT_TR_QUEUE_EMPTY	transmit queue emty

5.9.3.3 enum CO_COMMTASK_EVENT_T

Communication task events

5.9.4 Function Documentation

5.9.4.1 EXTERN_DECL void coCommStateEvent (CO_COMM_STATE_EVENT_T *newEvent*)

coCommStateEvent - set a new communication state

This function should be called, if a new communication state has been reached.
It sets the LEDs and informs the application about the event.

Returns

void

Parameters

newEvent	new communication event

5.9.4.2 EXTERN_DECL void coCommTask (void)

coCommTask - main communication task

This is the main communication task for the CANopen stack. It has to be called cyclically by the application or signal driven after each received CAN message or timer event.

Returns

void

5.9.4.3 EXTERN_DECL BOOL_T coCommTaskCheck (void)

coCommTaskCheck - check communication tasks

This function checks if all communication task are done. If not, it returns true. In this case, coCommTask() should be called again.

Returns

BOOL_T

Return values

CO_TRUE	further actions necessary - call coCommTask again
CO_FALSE	all actions passed

5.9.4.4 EXTERN_DECL RET_T coEventRegister_CAN_STATE (CO_EVENT_CAN_STATE_T *pFunction*)

coEventRegister_CAN_STATE - register can state changes

With this function the application can register a function which is called, when the CAN state was changed.
CAN states are:

- BUS_OFF
- BUS_ON
- PASSIV
- UNCHANGED

Returns

RET_T

Parameters

pFunction	pointer to function

5.9.4.5 EXTERN_DECL RET_T coEventRegister_COMM_EVENT (CO_EVENT_COMM_T *pFunction*)

coEventRegister_COMM_EVENT - register communication event changes

With this function the application can register a function which is called, when the communication state has been changed.

- BUS OFF - no communication possible
- CAN OVERRUN - messages was lost
- Receice queue full - receive messages is full
- Receice queue overrun - receive messages was lost
- Transmit queue full - no more messages can be send
- Transmit queue overflow - transmit messages was lost
- Transmit queue empty - new transmit messages can be send

Returns

RET_T

Parameters

pFunction	pointer to function

5.9.4.6 EXTERN_DECL void coQueueInit (void)

coQueueInit - (re)init queues

This function clears the transmit and the receive queue

Returns

none

5.10 co_datatype.h File Reference

data types

Macros

- #define MSG_OVERWRITE 1u
- #define MSG_RET_INHIBIT 2u

Enumerations

5.10.1 Detailed Description

data types

- contains defines for data types

5.10.2 Macro Definition Documentation

5.10.2.1 #define MSG_OVERWRITE 1u

transmit message flags: if the last message is not transmitted yet, overwrite the last data with the new data

5.10.2.2 #define MSG_RET_INHIBIT 2u

return, if the inhibit time is not ellapsed yet

5.10.3 Enumeration Type Documentation

5.10.3.1 enum BOOL_T

define for bool values

Enumerator

CO_FALSE	false
CO_TRUE	true

5.10.3.2 enum RET_T

Defines for RET_T

Enumerator

RET_OK	all ok
RET_INVALID_PARAMETER	error invalid parameter
RET_PARAMETER_INCOMPATIBLE	error incompatible parameter
RET_NOT_INITIALIZED	error function not initialized
RET_EVENT_NO_RESSOURCE	error no ressource available
RET_INVALID_NMT_STATE	error invalid NMT state
RET_INVALID_NODEID	invalid node id
RET_ALREADY_INITIALIZED	error already initialized
RET_IDX_NOT_FOUND	error index not found
RET_SUBIDX_NOT_FOUND	error subindex not found

RET_OD_ACCESS_ERROR	error access at object dictionary
RET_NO_READ_PERM	error no read permission
RET_NO_WRITE_PERM	error no write permission
RET_SDO_UNKNOWN_CCS	error unknown command specifier
RET_SDO_DATA_TYPE_NOT_MATCH	error wrong data type
RET_SDO_INVALID_VALUE	error invalid value
RET_SDO_TRANSFER_NOT_SUPPORTED	error transfer not supported
RET_OUT_OF_MEMORY	error out of memory
RET_DATA_TYPE_MISMATCH	error data type mismatch
RET_TOGGLE_MISMATCH	error toogle bit not alternate
RET_SDO_CRC_ERROR	error CRC mismatch
RET_SDO_WRONG_BLOCKSIZE	error wrong blocksize
RET_SDO_WRONG_SEQ_NR	error wrong sequence number
RET_SDO_TIMEOUT	error sdo timeout
RET_SDO_SPLIT_INDICATION	SDO split indikation
RET_NO_COB_AVAILABLE	error no cob available
RET_COB_DISABLED	error cob-id is disabled
RET_DRV_WRONG_BITRATE	error unknown bitrate
RET_DRV_ERROR	error driver
RET_DRV_TRANS_BUFFER_FULL	error transmit buffer full
RET_DRV_BUSY	error driver is busy
RET_MAP_ERROR	error map entry incorrect
RET_MAP_LEN_ERROR	error mapping len incorrect
RET_INHIBIT_ACTIVE	error inhibit is active
RET_INTERNAL_ERROR	error internal
RET_HARDWARE_ERROR	error hardware access
RET_ERROR_PRESENT_DEVICE_STATE	error wrong device state
RET_VALUE_NOT_AVAILABLE	error value not available
RET_ERROR_STORE	error store data
RET_SERVICE_ALREADY_INITIALIZED	service already initialized
RET_SERVICE_NOT_INITIALIZED	service not initialized
RET_SERVICE_BUSY	error service is busy
RET_CFG_CONVERT_ERROR	cfg manager convert error
RET_NETWORK_ID_UNKNOWN	network id unknown
RET_NW_NODE_ID_UNKNOWN	networking node id unknown
RET_NW_SDO_CHANNEL_IN_USE	networking sdo channel already in use

5.11 co_drv.h File Reference

defines for driver

Data Structures

- struct CO_CAN_COB_T
- struct CO_CAN_MSG_T

Functions

- EXTERN_DECL void codrvHardwareInit (void)

    codrvHardwareInit - hardware initialization
- EXTERN_DECL RET_T codrvCanInit (UNSIGNED16 bitRate)

    codrvCanInit - init CAN controller
- EXTERN_DECL RET_T codrvCanReInit (UNSIGNED16 bitRate)

    codrvCanReInit - reinit CAN controller
- EXTERN_DECL RET_T codrvCanSetBitRate (UNSIGNED16 bitRate)

    codrvCanSetBitRate - set CAN Bitrate
- EXTERN_DECL RET_T codrvCanStartTransmission (void)

    codrvCanStartTransmission - start can transmission if not active
- EXTERN_DECL void codrvCanDriverHandler (void)

    codrvCanDriverHandler - can driver handler
- EXTERN_DECL RET_T codrvCanEnable (void)

    codrvCanEnable - enable CAN controller
- EXTERN_DECL RET_T codrvCanDisable (void)

    codrvCanDisable - disable CAN controller
- EXTERN_DECL void coQueueMsgTransmitted (const CO_CAN_MSG_T ∗pBuf)

    coQueueMsgTransmitted - message was transmitted
- EXTERN_DECL CO_CAN_MSG_T ∗ coQueueGetNextTransmitMessage (void)

    coQueueGetNextTransmitMessage - get next message to transmit
- EXTERN_DECL BOOL_T coQueueReceiveMessageAvailable (void)

    coQueueReceiveMessageAvailable - receive messages available
- EXTERN_DECL void codrvCanEnableInterrupt (void)

    codrvCanEnableInterrupt - enable the CAN interrupt
- EXTERN_DECL void codrvCanDisableInterrupt (void)

    codrvCanDisableInterrupt - disable the CAN interrupt
- EXTERN_DECL RET_T codrvTimerSetup (UNSIGNED32 timerInterval)

    codrvTimerSetup - init and configure the hardware Timer

5.11.1 Detailed Description

defines for driver

- contains defines for driver

5.11.2 Function Documentation

5.11.2.1 EXTERN_DECL RET_T codrvCanDisable (void)

codrvCanDisable - disable CAN controller

This function disables the CAN controller. The function waits for the CAN controller being disabled. Code calling this function typically expects that after returning the CAN controller is in Init mode.

But note, the time the CAN controller needs to enter the Init mode can be as long as the duration of one CAN frame.

Returns

RET_T

Return values

RET_OK	CAN controller is set to be disabled

5.11.2.2 EXTERN_DECL void codrvCanDriverHandler (void)

codrvCanDriverHandler - can driver handler

This function is cyclically called from the CANopen stack to get the current CAN state (BUS_OFF, PASSIVE, AC↩ TIVE).

If a bus off event has occurred, this function should try to get to bus on again (activate the CAN controller).

Returns

void

5.11.2.3 EXTERN_DECL RET_T codrvCanEnable (void)

codrvCanEnable - enable CAN controller

This function enables the CAN controller. At this point the enable bit is set. Typically the CAN controller requests 11 recessive bits to go in active mode. This will be checked later outside of this function.

Returns

RET_T

Return values

RET_OK	CAN controller, enabled was set

5.11.2.4 EXTERN_DECL RET_T codrvCanInit (UNSIGNED16 *bitRate*)

codrvCanInit - init CAN controller

This function initializes the CAN controller and configures the bitrate. At the end of the function, the CAN controller should be in state disabled.

Returns

RET_T

Return values

RET_OK	initialization was OK

Parameters

bitRate	CAN bitrate

5.11.2.5 EXTERN_DECL RET_T codrvCanReInit (UNSIGNED16 *bitRate*)

codrvCanReInit - reinit CAN controller

This Function reinits the CAN controller after deactivation.

In Filter mode: After this function call all Filter are reset and must be reconfigured!

At the end of the function, the CAN controller should be in state disabled.

Parameters

bitRate	- CANopen bitrate

Returns

RET_T

Parameters

bitRate	CAN bitrate

5.11.2.6 EXTERN_DECL RET_T codrvCanSetBitRate (UNSIGNED16 *bitRate*)

codrvCanSetBitRate - set CAN Bitrate

This function sets the CAN Bitrate to the given value. Changing the Bitrate is only allowed, if the CAN controller is in reset. The state at the start of the function is unknown, so the CAN controller should be switch to state reset.

At the end of the function the CAN controller should be stay in state reset.

Returns

RET_T

Return values

RET_OK	setting of Bitrate was OK

Parameters

bitRate	CAN Bitrate in kbit/s

5.11.2.7 EXTERN_DECL RET_T codrvCanStartTransmission (void)

codrvCanStartTransmission - start can transmission if not active

Transmission of CAN messages should be interrupt driven. If a message was sent, the Transmit Interrupt is called and the next message can be transmitted. To start the transmission of the first message, this function is called from the CANopen stack.

The easiest way to implement this function is to trigger the transmit interrupt, but only of the transmission is not already active.

Returns

RET_T

Return values

RET_OK	start transmission was successful

5.11.2.8 EXTERN_DECL void codrvHardwareInit (void)

codrvHardwareInit - hardware initialization

This function initializes the hardware, incl. clock and CAN hardware.

5.11.2.9 EXTERN_DECL RET_T codrvTimerSetup (UNSIGNED32 *timerInterval*)

codrvTimerSetup - init and configure the hardware Timer

This function starts a cyclic hardware timer to provide a timing interval for the CANopen library. Alternativly it can be derived from an other system timer with the timer interval given by the function parameter.

Returns

RET_T

Return values

RET_OK	intialization of the timer was ok

Parameters

timerInterval	timer interval in usec

5.11.2.10 EXTERN_DECL CO_CAN_MSG_T∗ coQueueGetNextTransmitMessage (void)

coQueueGetNextTransmitMessage - get next message to transmit

This function returns the next available transmit message from the transmit queue. It increments also trBufferRdCnt.

Returns

CO_CAN_MSG_T∗ pointer to next tx message

Return values

!NULL	pointer to transmit queue entry
NULL	no message available

5.11.2.11 EXTERN_DECL void coQueueMsgTransmitted (const CO_CAN_MSG_T ∗ *pBuf*)

coQueueMsgTransmitted - message was transmitted

This function is called after a message was succesfull transmitted.

Returns

none

Parameters

pBuf	pointer to transmitted message

5.11.2.12 EXTERN_DECL BOOL_T coQueueReceiveMessageAvailable (void)

coQueueReceiveMessageAvailable - receive messages available

This functions checks the receive queue for new messages. Are new messages available, return CO_TRUE. Otherwise CO_FALSE

Return values

CO_FALSE	no data available
CO_FALSE	data available

5.12 co_dynod.c File Reference

This file implements a dynamic object dictionary for objects => 0x2000.

Functions

- RET_T coDynOdInit (UNSIGNED16 objCnt, UNSIGNED16 u8Cnt, UNSIGNED16 u16Cnt, UNSIGNED16 u32Cnt, UNSIGNED16 i8Cnt, UNSIGNED16 i16Cnt, UNSIGNED16 i32Cnt, UNSIGNED16 u64Cnt) *coDynOdInit - init dynamic object dictionary*

- RET_T coDynOdRelease (void)

  coDynOdRelease - release dynamic object dictionary
- RET_T coDynOdAddIndex (UNSIGNED16 index, UNSIGNED8 nrOfSubs, CO_ODTYPE_T odType)

  coDynOdAddIndex - add a new object index
- RET_T coDynOdAddSubIndex (UNSIGNED16 index, UNSIGNED8 subIndex, CO_DATA_TYPE_T dataType, UNSIGNED16 attr, void ∗pVar)

  coDynOdAddSubIndex - add new subindex
- RET_T coDynOdSetSubIndexAddr (UNSIGNED16 index, UNSIGNED8 subIndex, CO_DATA_TYPE_↩ T dataType, void ∗pVar)

  coDynOdSetSubIndexAddr - set new pointer for subindex

5.12.1 Detailed Description

This file implements a dynamic object dictionary for objects => 0x2000.

5.12.2 Function Documentation

5.12.2.1 RET_T coDynOdAddIndex (UNSIGNED16 *index*, UNSIGNED8 *nrOfSubs*, CO_ODTYPE_T *odType*)

coDynOdAddIndex - add a new object index

Return values

RET_IDX_NOT_FOUND	index < 0x2000 are not allowed
RET_INVALID_PARAMETER	index already exist
RET_EVENT_NO_RESSOURCE	no resource available

Parameters

index	index
nrOfSubs	number of subindex
odType	variable, array, struct

5.12.2.2 RET_T coDynOdAddSubIndex (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, CO_DATA_TYPE_T *dataType*, UNSIGNED16 *attr*, void ∗ *pVar*)

coDynOdAddSubIndex - add new subindex

no check for to many data or duplicate subindex

Return values

RET_DATA_TYPE_MISMATCH	data type not supported (only U8, U16, U32, I8, I16, I32 allowed)
RET_IDX_NOT_FOUND	index not found

Parameters

index	index
subIndex	number of subindex
dataType	data type
attr	attribute
pVar	pointer to variable

5.12.2.3 RET_T coDynOdInit (UNSIGNED16 *objCnt*, UNSIGNED16 *u8Cnt*, UNSIGNED16 *u16Cnt*, UNSIGNED16 *u32Cnt*, UNSIGNED16 *i8Cnt*, UNSIGNED16 *i16Cnt*, UNSIGNED16 *i32Cnt*, UNSIGNED16 *u64Cnt*)

coDynOdInit - init dynamic object dictionary

Return values

RET_OK	initialisation OK
RET_EVENT_NO_RESSOURCE	error at malloc()

Parameters

objCnt	number of new objects for can line
u8Cnt	number of U8 vars for can line
u16Cnt	number of U16 vars for can line
u32Cnt	number of U32 vars for can line
i8Cnt	number of i8 vars for can line
i16Cnt	number of i16 vars for can line
i32Cnt	number of i32 vars for can line
u64Cnt	number of U64 vars for can line

5.12.2.4 RET_T coDynOdRelease (void)

coDynOdRelease - release dynamic object dictionary

Deinit dynamic object dictionary and release all requested memory

Return values

RET_OK	deinitialisation OK

5.12.2.5 RET_T coDynOdSetSubIndexAddr (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, CO_DATA_TYPE_T *dataType*, void * *pVar*)

coDynOdSetSubIndexAddr - set new pointer for subindex

set a new data pointer for a given sub index

Return values

RET_DATA_TYPE_MISMATCH	data type not supported (only U8, U16, U32, I8, I16, I32 allowed)
RET_IDX_NOT_FOUND	index not found

Parameters

index	index
subIndex	number of subindex
dataType	data type
pVar	pointer to variable

5.13 co_edsparse.c File Reference

EDS parser module.

Functions

- RET_T coEdsparseAddEdsToRepository (char ∗edsFilePath)
  coEdsparseAddEdsToRepository - add file to eds repository
- RET_T coEdsparseDetectSlaveEds (UNSIGNED8 nodeId, UNSIGNED8 sdoClientNr, CO_DETECT_SLA↩ VE_FCT_T finishFct)
  detectSlaveEds - detect slave EDS file
- RET_T coEdsparseReadEdsMapping (UNSIGNED8 nodeId, char ∗edsFileName)
  coEdsparseReadEdsMapping - read mapping from EDS file
- CO_EDS_MAP_TABLE_T ∗ coEdsparseGetRPdoMapEntry (UNSIGNED16 mapIdx)
  coEdsparseGetRPdoMapEntry - get RPDO map entry from EDS table
- CO_EDS_MAP_TABLE_T ∗ coEdsparseGetTPdoMapEntry (UNSIGNED16 mapIdx)
  coEdsparseGetTPdoMapEntry - get TPDO map entry from EDS table
- UNSIGNED16 coEdsparseGetSupportedObjCnt (char ∗edsFileName, char ∗section)
  coEdsparseGetSupportedIndexCnt - return number of supported index
- RET_T coEdsparseGetIndexDesc (char ∗edsFileName, char ∗pSection, UNSIGNED16 edsIdx, UNSIGNE↩ D16 ∗pIndex, UNSIGNED8 ∗pNrOfSubs)
  coEdsparseGetIndexDesc - return index description
- RET_T coEdsparseGetObjectDesc (char ∗edsFileName, UNSIGNED16 index, UNSIGNED8 subIndex, U↩ NSIGNED16 ∗pDataType, UNSIGNED16 ∗pAttr, char ∗pDefaultVal)
  coEdsparseGetObjectDesc - get object description

5.13.1 Detailed Description

EDS parser module.

contains EDS parse routines

5.13.2 Function Documentation

5.13.2.1 RET_T coEdsparseAddEdsToRepository (char ∗ *edsFilePath*)

coEdsparseAddEdsToRepository - add file to eds repository

This function add an EDS file to the internal repository and parse it for identity data

Returns

RET_T

Parameters

edsFilePath	eds file name

5.13.2.2 RET_T coEdsparseDetectSlaveEds (UNSIGNED8 *nodeId,* UNSIGNED8 *sdoClientNr,* CO_DETECT_SLAVE_FCT_T *finishFct*)

detectSlaveEds - detect slave EDS file

This function read the identity from the given slave and checks it by available identity parameter from EDS repository. If it fit the identity from the device and the EDS given finishFct returns the fitting EDS file name.

If an error occurs, the finishFct returns without EDS file name but with the appropriate error.

Returns

RET_T

Parameters

nodeId	node id
sdoClientNr	SDO client number
finishFct	function for finish action

5.13.2.3 RET_T coEdsparseGetIndexDesc (char ∗ *edsFileName,* char ∗ *pSection,* UNSIGNED16 *edsIdx,* UNSIGNED16 ∗ *pIndex,* UNSIGNED8 ∗ *pNrOfSubs*)

coEdsparseGetIndexDesc - return index description

This function returns some information about the object index given by eds index. The maximum number of eds index can get by function coEdsparseGetSupportedObjCnt()

section should be one of MandatoryObjects OptionalObjects ManufacturerObjects

Returns

RET_T

Parameters

edsFileName	eds file name
pSection	section name
edsIdx	index at eds file list
pIndex	object index
pNrOfSubs	number of subindex

5.13.2.4 RET_T coEdsparseGetObjectDesc (char ∗ *edsFileName*, UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED16 ∗ *pDataType*, UNSIGNED16 ∗ *pAttr*, char ∗ *pDefaultVal*)

coEdsparseGetObjectDesc - get object description

This function returns object description from EDS for the given object index.

Returns

RET_T

Parameters

edsFileName	eds file name
index	object index
subIndex	object subindex
pDataType	pointer for data type
pAttr	pointer for object attributes
pDefaultVal	pointer for default val

5.13.2.5 CO_EDS_MAP_TABLE_T∗ coEdsparseGetRPdoMapEntry (UNSIGNED16 *mapIdx*)

coEdsparseGetRPdoMapEntry - get RPDO map entry from EDS table

This function returns a RPDO map entry from the EDS table

Returns

RET_T

Parameters

mapIdx	map index at table

5.13.2.6 UNSIGNED16 coEdsparseGetSupportedObjCnt (char ∗ *edsFileName*, char ∗ *section*)

coEdsparseGetSupportedIndexCnt - return number of supported index

This function counts the supported index for the given section and return the number of supported index

section should be one of MandatoryObjects OptionalObjects ManufacturerObjects

Returns

number of supported index

Parameters

edsFileName	eds file name
section	section name

5.13.2.7 CO_EDS_MAP_TABLE_T∗ coEdsparseGetTPdoMapEntry (UNSIGNED16 *mapIdx*)

coEdsparseGetTPdoMapEntry - get TPDO map entry from EDS table

This function returns a TPDO map entry from the EDS table

Returns

RET_T

Parameters

mapIdx	map index at table

5.13.2.8 RET_T coEdsparseReadEdsMapping (UNSIGNED8 *nodeId,* char ∗ *edsFileName*)

coEdsparseReadEdsMapping - read mapping from EDS file

This function read the EDS file and save the values at the internal mapping tables.

Returns

RET_T

Parameters

nodeId	node id
edsFileName	eds file name

5.14 co_edsparse.h File Reference

defines for eds parser services

Typedefs

- typedef void(∗ CO_DETECT_SLAVE_FCT_T) (UNSIGNED8 nodeId, char ∗pEdsFileName)

  function pointer to detect slave finish indication

Functions

- RET_T coEdsparseAddEdsToRepository (char ∗edsFilePath)

  coEdsparseAddEdsToRepository - add file to eds repository
- RET_T coEdsparseDetectSlaveEds (UNSIGNED8 nodeId, UNSIGNED8 sdoClientNr, CO_DETECT_SLA↩ VE_FCT_T finishFct)

  detectSlaveEds - detect slave EDS file
- RET_T coEdsparseReadEdsMapping (UNSIGNED8 nodeId, char ∗edsFileName)

  coEdsparseReadEdsMapping - read mapping from EDS file
- CO_EDS_MAP_TABLE_T ∗ coEdsparseGetRPdoMapEntry (UNSIGNED16 mapIdx)

  coEdsparseGetRPdoMapEntry - get RPDO map entry from EDS table
- CO_EDS_MAP_TABLE_T ∗ coEdsparseGetTPdoMapEntry (UNSIGNED16 mapIdx)

  coEdsparseGetTPdoMapEntry - get TPDO map entry from EDS table
- UNSIGNED16 coEdsparseGetSupportedObjCnt (char ∗edsFileName, char ∗section)

  coEdsparseGetSupportedIndexCnt - return number of supported index
- RET_T coEdsparseGetObjectDesc (char ∗edsFileName, UNSIGNED16 index, UNSIGNED8 subIndex, U↩ NSIGNED16 ∗pDataType, UNSIGNED16 ∗pAttr, char ∗pDefaultVal)

  coEdsparseGetObjectDesc - get object description
- RET_T coEdsparseGetIndexDesc (char ∗edsFileName, char ∗section, UNSIGNED16 edsIdx, UNSIGNED16 ∗pIndex, UNSIGNED8 ∗pNrOfSubs)

  coEdsparseGetIndexDesc - return index description

5.14.1 Detailed Description

defines for eds parser services

- contains defines for eds parser services

5.14.2 Typedef Documentation

5.14.2.1 typedef void(∗ CO_DETECT_SLAVE_FCT_T) (UNSIGNED8 nodeId, char ∗pEdsFileName)

function pointer to detect slave finish indication

Parameters

nodeId	- node id
pEdsFileName	- EDS file name fitting the node

Returns

void

5.14.3 Function Documentation

5.14.3.1 RET_T coEdsparseAddEdsToRepository (char ∗ *edsFilePath*)

coEdsparseAddEdsToRepository - add file to eds repository

This function add an EDS file to the internal repository and parse it for identity data

Returns

RET_T

Parameters

edsFilePath	eds file name

5.14.3.2 RET_T coEdsparseDetectSlaveEds (UNSIGNED8 *nodeId,* UNSIGNED8 *sdoClientNr,* CO_DETECT_SLAVE_FCT_T *finishFct*)

detectSlaveEds - detect slave EDS file

This function read the identity from the given slave and checks it by available identity parameter from EDS repository. If it fit the identity from the device and the EDS given finishFct returns the fitting EDS file name.

If an error occurs, the finishFct returns without EDS file name but with the appropriate error.

Returns

RET_T

Parameters

nodeId	node id
sdoClientNr	SDO client number
finishFct	function for finish action

5.14.3.3 RET_T coEdsparseGetIndexDesc (char ∗ *edsFileName,* char ∗ *pSection,* UNSIGNED16 *edsIdx,* UNSIGNED16 ∗ *pIndex,* UNSIGNED8 ∗ *pNrOfSubs*)

coEdsparseGetIndexDesc - return index description

This function returns some information about the object index given by eds index. The maximum number of eds index can get by function coEdsparseGetSupportedObjCnt()

section should be one of MandatoryObjects OptionalObjects ManufacturerObjects

Returns

RET_T

Parameters

edsFileName	eds file name
pSection	section name
edsIdx	index at eds file list
pIndex	object index
pNrOfSubs	number of subindex

5.14.3.4 RET_T coEdsparseGetObjectDesc (char ∗ *edsFileName,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED16 ∗ *pDataType,* UNSIGNED16 ∗ *pAttr,* char ∗ *pDefaultVal*)

coEdsparseGetObjectDesc - get object description

This function returns object description from EDS for the given object index.

Returns

RET_T

Parameters

edsFileName	eds file name
index	object index
subIndex	object subindex
pDataType	pointer for data type
pAttr	pointer for object attributes
pDefaultVal	pointer for default val

5.14.3.5 CO_EDS_MAP_TABLE_T∗ coEdsparseGetRPdoMapEntry (UNSIGNED16 *mapIdx*)

coEdsparseGetRPdoMapEntry - get RPDO map entry from EDS table

This function returns a RPDO map entry from the EDS table

Returns

RET_T

Parameters

mapIdx	map index at table

5.14.3.6 UNSIGNED16 coEdsparseGetSupportedObjCnt (char ∗ *edsFileName,* char ∗ *section*)

coEdsparseGetSupportedIndexCnt - return number of supported index

This function counts the supported index for the given section and return the number of supported index

section should be one of MandatoryObjects OptionalObjects ManufacturerObjects

Returns

number of supported index

Parameters

edsFileName	eds file name
section	section name

5.14.3.7 CO_EDS_MAP_TABLE_T∗ coEdsparseGetTPdoMapEntry (UNSIGNED16 *mapIdx*)

coEdsparseGetTPdoMapEntry - get TPDO map entry from EDS table

This function returns a TPDO map entry from the EDS table

Returns

RET_T

Parameters

mapIdx	map index at table

5.14.3.8 RET_T coEdsparseReadEdsMapping (UNSIGNED8 *nodeId,* char ∗ *edsFileName*)

coEdsparseReadEdsMapping - read mapping from EDS file

This function read the EDS file and save the values at the internal mapping tables.

Returns

RET_T

Parameters

nodeId	node id
edsFileName	eds file name

5.15 co_emcy.c File Reference

Emergency handling.

Functions

- RET_T coEmcyWriteReq (UNSIGNED16 emcyErrCode, CO_CONST UNSIGNED8 pData[ ])

  coEmcyWriteReq - write an emergency message
- RET_T coEventRegister_EMCY (CO_EVENT_EMCY_T pFunction)

  coEventRegister_EMCY - register emergency event function
- RET_T coEventRegister_EMCY_CONSUMER (CO_EVENT_EMCY_CONS_T pFunction)

  coEventRegister_EMCY_CONSUMER - register emergency consumer event function
- RET_T coEmcyProducerInit (void)

  coEmcyProducerInit - initialization for emergency producer
- RET_T coEmcyConsumerInit (UNSIGNED8 emcyCnt)

  coEmcyConsumerInit - initialization for emergency consumer

5.15.1 Detailed Description

Emergency handling.

contains emcy routines

5.15.2 Function Documentation

5.15.2.1 RET_T coEmcyConsumerInit (UNSIGNED8 *emcyCnt*)

coEmcyConsumerInit - initialization for emergency consumer

This function initializes the emergency consumers.

Returns

RET_T

Parameters

emcyCnt	number of emergency consumers

5.15.2.2 RET_T coEmcyProducerInit (void)

coEmcyProducerInit - initialization for emergency producer

This function initializes the emergency producer functionality.

Returns

RET_T

5.15.2.3 RET_T coEmcyWriteReq (UNSIGNED16 *emcyErrCode,* CO_CONST UNSIGNED8 *pData[ ]*)

coEmcyWriteReq - write an emergency message

With this function, an emergency message can be send.
The message is automatically composed and transmitted by the given parameter and the error register value (index 0x1001:0). After that, the error history (index 0x1003:n) is updated with the new data.
The parameter pData can be NULL, if no application specific data should be sent.

The error register (index 0x1001:0) has to be updated by the application.

Returns

RET_T

Parameters

emcyErrCode	emergency error code
pData	pointer to additional 5 data bytes

5.15.2.4 RET_T coEventRegister_EMCY (CO_EVENT_EMCY_T *pFunction*)

coEventRegister_EMCY - register emergency event function

This function registers an emergency indication function.

Returns

RET_T

5.15.2.5 RET_T coEventRegister_EMCY_CONSUMER (CO_EVENT_EMCY_CONS_T *pFunction*)

coEventRegister_EMCY_CONSUMER - register emergency consumer event function

This function registers an emergency consumer indication function.

Returns

RET_T

5.16 co_emcy.h File Reference

defines for emcy services

Macros

- #define CO_EMCY_ERRCODE_PDO_LEN 0x8210u
- #define CO_EMCY_ERRCODE_COMM_ERROR 0x8130u

Typedefs

- typedef RET_T(∗ CO_EVENT_EMCY_T) (UNSIGNED16 errCode, const UNSIGNED8 ∗addErrorCode)

  function pointer to emergency function
- typedef void(∗ CO_EVENT_EMCY_CONS_T) (UNSIGNED8 node, UNSIGNED16 errCode, UNSIGNED8 errorRegister, UNSIGNED8 const ∗addErrorCode)

  function pointer to emergency consumer function

Functions

- EXTERN_DECL RET_T coEmcyProducerInit (void)

  coEmcyProducerInit - initialization for emergency producer
- EXTERN_DECL RET_T coEmcyConsumerInit (UNSIGNED8 emcyCnt)

  coEmcyConsumerInit - initialization for emergency consumer
- EXTERN_DECL RET_T coEventRegister_EMCY (CO_EVENT_EMCY_T pFunction)

  coEventRegister_EMCY - register emergency event function
- EXTERN_DECL RET_T coEventRegister_EMCY_CONSUMER (CO_EVENT_EMCY_CONS_T pFunction)

  coEventRegister_EMCY_CONSUMER - register emergency consumer event function

5.16.1 Detailed Description

defines for emcy services

- contains defines for emcy services

5.16.2 Macro Definition Documentation

5.16.2.1 #define CO_EMCY_ERRCODE_COMM_ERROR 0x8130u

define for Emergency Error Code communication error

5.16.2.2 #define CO_EMCY_ERRCODE_PDO_LEN 0x8210u

define for Emergency Error Code wrong PDO length

5.16.3 Typedef Documentation

5.16.3.1 typedef void(∗ CO_EVENT_EMCY_CONS_T) (UNSIGNED8 node, UNSIGNED16 errCode, UNSIGNED8 errorRegister, UNSIGNED8 const ∗addErrorCode)

function pointer to emergency consumer function

Parameters

node	- node id of received emergency
errCode	- emergency error code
errorRegister	- emergency error register
addErrorCode	- pointer to 5 bytes error code

Returns

none

5.16.3.2 typedef RET_T(∗ CO_EVENT_EMCY_T) (UNSIGNED16 errCode, const UNSIGNED8 ∗addErrorCode)

function pointer to emergency function

Parameters

errCode	- emergency error code
addErrorCode	- pointer to 5 bytes error code

Returns

RET_T

Return values

RET_OK	send emergency
RET_xx	don't send emergency

5.16.4 Function Documentation

5.16.4.1 EXTERN_DECL RET_T coEmcyConsumerInit (UNSIGNED8 *emcyCnt*)

coEmcyConsumerInit - initialization for emergency consumer

This function initializes the emergency consumers.

Returns

RET_T

Parameters

emcyCnt	number of emergency consumers

5.16.4.2 EXTERN_DECL RET_T coEmcyProducerInit (void)

coEmcyProducerInit - initialization for emergency producer

This function initializes the emergency producer functionality.

Returns

RET_T

5.16.4.3 EXTERN_DECL RET_T coEventRegister_EMCY (CO_EVENT_EMCY_T *pFunction*)

coEventRegister_EMCY - register emergency event function

This function registers an emergency indication function.

Returns

RET_T

5.16.4.4 EXTERN_DECL RET_T coEventRegister_EMCY_CONSUMER (CO_EVENT_EMCY_CONS_T *pFunction*)

coEventRegister_EMCY_CONSUMER - register emergency consumer event function

This function registers an emergency consumer indication function.

Returns

RET_T

5.17 co_errctrl.c File Reference

Error control handling (Heartbeat, Guarding)

Functions

- RET_T coHbConsumerSet (UNSIGNED8 node, UNSIGNED16 hbTime)
  coHbConsumerSet - setup heartbeat consumer
- RET_T coHbConsumerStart (UNSIGNED8 node)
  coHbConsumerStart - start heartbeat consumer monitoring
- CO_NMT_STATE_T coNmtGetRemoteNodeState (UNSIGNED8 nodeId)
  coNmtGetRemoteNodeState - get remote node state
- RET_T coEventRegister_ERRCTRL (CO_EVENT_ERRCTRL_T pFunction)
  coEventRegister_ERRCTRL - register error control event
- RET_T coErrorCtrlInit (UNSIGNED16 hbTime, UNSIGNED8 hbConsCnt)
  coInitNmt - init error control

5.17.1 Detailed Description

Error control handling (Heartbeat, Guarding)

Contains error control routines to handle Heartbeat or Guarding.

5.17.2 Function Documentation

5.17.2.1 RET_T coErrorCtrlInit (UNSIGNED16 *hbTime,* UNSIGNED8 *hbConsCnt*)

coInitNmt - init error control

Setup error control handling for local node (transmit heartbeat) and remote node (heartbeat monitoring)

Returns

RET_T

Parameters

hbTime	heartbeat producer time
hbConsCnt	heartbeat consumer count

5.17.2.2 RET_T coEventRegister_ERRCTRL (CO_EVENT_ERRCTRL_T *pFunction*)

coEventRegister_ERRCTRL - register error control event

Returns

RET_T

Parameters

pFunction	pointer to function

5.17.2.3 RET_T coHbConsumerSet (UNSIGNED8 *node,* UNSIGNED16 *hbTime*)

coHbConsumerSet - setup heartbeat consumer

This function configures a hearbeat consumer for the given node-id and the monitoring time. The data are automatically saved at the object dictionary. If an entry at the object dictionary already exist, then it will be overwritten.
The parameter node have to be valid, otherwise the function returns an error.

Returns

RET_T

Return values

RET_PARAMETER_INCOMPATIBLE	invalid node id

Parameters

node	node id
hbTime	heartbeat monitoring time

5.17.2.4 RET_T coHbConsumerStart (UNSIGNED8 *node*)

coHbConsumerStart - start heartbeat consumer monitoring

This function starts a hearbeat consumer monitoring for the given node-id and the configured monitoring time from object dictionary.

Please note: The NMT state is set to unknown until next HB was received

Returns

RET_T

Return values

RET_PARAMETER_INCOMPATIBLE	invalid node id

Parameters

node	node id

5.17.2.5 CO_NMT_STATE_T coNmtGetRemoteNodeState (UNSIGNED8 *nodeId*)

coNmtGetRemoteNodeState - get remote node state

This function returns the NMT state of a remote node. If heartbeat monitoring of this node is disabled or has been failed, CO_NMT_STATE_UNKNOWN is returned.

Returns

CO_NMT_STATE_T

Parameters

node↩ Id	remote node id

5.18 co_event.c File Reference

event routines

Functions

- RET_T icoEventStart (CO_EVENT_T ∗pEvent, CO_EVENT_FCT_T ptrToFct, void ∗pData)

    coEventStart - start a event
- BOOL_T icoEventIsActive (CO_CONST CO_EVENT_T ∗pEvent)

    coEventIsActive - check if event is active
- void icoEventInit (void)

    icoEventInit - init event interval

5.18.1 Detailed Description

event routines

contains event routines

5.18.2 Function Documentation

5.18.2.1 void icoEventInit (void)

icoEventInit - init event interval

This function initializes the internal event handling.

Returns

none

5.18.2.2 BOOL_T icoEventIsActive (CO_CONST CO_EVENT_T ∗ *pEvent*)

coEventIsActive - check if event is active

With this function can be ckecked, if a event is currently in the event list.

Returns

BOOL_T

Return values

CO_TRUE	event is active
CO_FALSE	event is not active

Parameters

pEvent	pointer to event struct

5.18.2.3 RET_T icoEventStart (CO_EVENT_T ∗ *pEvent,* CO_EVENT_FCT_T *ptrToFct,* void ∗ *pData*)

coEventStart - start a event

This function add an event at end of the event list

Returns

RET_T

Parameters

pEvent	pointer to eventstruct
ptrToFct	function for event
pData	pointer for own data

5.19 co_flyingmaster.c File Reference

flying master handling

5.19.1 Detailed Description

flying master handling

contains flying master services

5.20 co_flyingmaster.h File Reference

defines for nmt flying master services

Typedefs

- typedef void(∗ CO_EVENT_FLYMA_T) (CO_FLYMA_STATE_T, UNSIGNED8, UNSIGNED8)
  function pointer to NMT flying master event function

Enumerations

5.20.1 Detailed Description

defines for nmt flying master services

- contains defines for nmt flying master services

5.20.2 Typedef Documentation

5.20.2.1 typedef void(∗ CO_EVENT_FLYMA_T) (CO_FLYMA_STATE_T, UNSIGNED8, UNSIGNED8)

function pointer to NMT flying master event function

Parameters

nmtFlymaState	- flying master event
node	- node id of actual master
prior	- priority of actual master

Returns

void

5.20.3 Enumeration Type Documentation

5.20.3.1 enum CO_FLYMA_STATE_T

NMT states

Enumerator

CO_FLYMA_STATE_DETECT_NO_MASTERS	no master detected
CO_FLYMA_STATE_MASTERS_AVAILABLE	master capable available
CO_FLYMA_STATE_NO_ACTIVE_MASTER	no active master found
CO_FLYMA_STATE_NEGOTIATION_STARTED	negotiation started
CO_FLYMA_STATE_MASTER	we are master
CO_FLYMA_STATE_SLAVE	we are slave

5.21 co_gfc.c File Reference

global failsafe command handling

5.21.1 Detailed Description

global failsafe command handling

Contains functions for the global failsafe services. The global failsafe service is not safety relevant, so there dynamic events possible for this service.

5.22 co_gfc.h File Reference

defines and the public API for the GFC modul.

Typedefs

- typedef void(∗ CO_EVENT_GFC_T) (void)

  function pointer to gfc function

5.22.1 Detailed Description

defines and the public API for the GFC modul.

- contains defines for gfc services

5.22.2 Typedef Documentation

5.22.2.1 typedef void(∗ CO_EVENT_GFC_T) (void)

function pointer to gfc function

Returns

void

5.23 co_guarding.c File Reference

Gaurding Master services.

Functions

- RET_T coGuardingMasterStart (UNSIGNED8 node)

  coGuardingMasterStart - start master node guarding
- RET_T coGuardingMasterStop (UNSIGNED8 node)

  coGuardingMasterStop - stop master node guarding
- CO_NMT_STATE_T icoGuardGetRemoteNodeState (UNSIGNED8 nodeId)

  coNmtGetRemoteNodeState - get remote node state

5.23.1 Detailed Description

Gaurding Master services.

Contains gurading master routines.

5.23.2 Function Documentation

5.23.2.1 RET_T coGuardingMasterStart (UNSIGNED8 *node*)

coGuardingMasterStart - start master node guarding

This function starts the master node guarding monitoring for the given node-id and the configured monitoring time from object dictionary.

Please note: The NMT state is set to unknown until next guarding was received

Returns

RET_T

Return values

RET_PARAMETER_INCOMPATIBLE	invalid node id

Parameters

node	node id

5.23.2.2 RET_T coGuardingMasterStop (UNSIGNED8 *node*)

coGuardingMasterStop - stop master node guarding

This function stops the master node guarding monitoring for the given node-id

Returns

RET_T

Return values

RET_PARAMETER_INCOMPATIBLE	invalid node id

Parameters

node	node id

5.23.2.3 CO_NMT_STATE_T icoGuardGetRemoteNodeState (UNSIGNED8 *nodeId*)

coNmtGetRemoteNodeState - get remote node state

This function returns the NMT state of a remote node. If guarding monitoring of this node is disabled or has been failed, CO_NMT_STATE_UNKNOWN is returned.

Returns

CO_NMT_STATE_T

Parameters

node↩ Id	remote node id

5.24 co_led.c File Reference

LED handling according CiA 303-3.

Functions

- void coLedSetGreen (CO_LED_STATE_T newLedState)

  coLedSetGreen - set green led to new state
- void coLedSetRed (CO_LED_STATE_T newLedState)

  coLedSetRed - set red led to new state
- void coLedSetState (CO_LED_STATE_T newState, BOOL_T on)

  coLedSetState - set led state
- RET_T coEventRegister_LED_GREEN (CO_EVENT_LED_T pFunction)

  coEventRegister_LED_GREEN - register for green LED
- RET_T coEventRegister_LED_RED (CO_EVENT_LED_T pFunction)

  coEventRegister_LED_RED - register for red LED

5.24.1 Detailed Description

LED handling according CiA 303-3.

contains LED handling according CiA 303-3

5.24.2 Function Documentation

5.24.2.1 RET_T coEventRegister_LED_GREEN (CO_EVENT_LED_T *pFunction*)

coEventRegister_LED_GREEN - register for green LED

Returns

RET_T

Parameters

pFunction	pointer to function

5.24.2.2 RET_T coEventRegister_LED_RED (CO_EVENT_LED_T *pFunction*)

coEventRegister_LED_RED - register for red LED

Register application function for controlling of LED state

Returns

RET_T

Parameters

pFunction	pointer to function

5.24.2.3 void coLedSetGreen (CO_LED_STATE_T *newLedState*)

coLedSetGreen - set green led to new state

Set green LED to one of the following state:

- OFF,
- FLICKERING,
- FLASH_1, FLASH_2, FLASH_3,
- BLINKING,
- ON

Returns

none

Parameters

newLedState	new led state

5.24.2.4 void coLedSetRed (CO_LED_STATE_T *newLedState*)

coLedSetRed - set red led to new state

Set led LED to one of the following state: OFF, FLICKERING, FLASH_1, FLASH_2, FLASH_3, BLINKING, ON

Returns

none

Parameters

newLedState	new led state

5.24.2.5 void coLedSetState (CO_LED_STATE_T *newState*, BOOL_T *on*)

coLedSetState - set led state

Set the error led to special state OFF, FLICKERING, FLASH_1, FLASH_2, FLASH_3, BLINKING, ON

All states are saved. Only the highest prior state is displayed. If the highest state is reset, the next state is displayed.

Returns

none

Parameters

newState	new state
on	set state to on/off

5.25 co_led.h File Reference

defines for usage of LED CiA 303

Typedefs

- typedef void(∗ CO_EVENT_LED_T) (BOOL_T)

  function pointer to LED indication function

Enumerations

Functions

- EXTERN_DECL void coLedSetGreen (CO_LED_STATE_T newLedState)

  coLedSetGreen - set green led to new state
- EXTERN_DECL void coLedSetRed (CO_LED_STATE_T newLedState)

  coLedSetRed - set red led to new state
- EXTERN_DECL void coLedSetState (CO_LED_STATE_T newState, BOOL_T on)

  coLedSetState - set led state
- EXTERN_DECL RET_T coEventRegister_LED_GREEN (CO_EVENT_LED_T pFunction)

  coEventRegister_LED_GREEN - register for green LED
- EXTERN_DECL RET_T coEventRegister_LED_RED (CO_EVENT_LED_T pFunction)

  coEventRegister_LED_RED - register for red LED

5.25.1 Detailed Description

defines for usage of LED CiA 303

- contains defines for usage of LED CiA 303

5.25.2 Typedef Documentation

5.25.2.1 typedef void(∗ CO_EVENT_LED_T) (BOOL_T)

function pointer to LED indication function

Parameters

led_state	- set led on/off

Returns

void

5.25.3 Enumeration Type Documentation

5.25.3.1 enum CO_LED_STATE_T

LED states

Enumerator

CO_LED_STATE_OFF	led is off
CO_LED_STATE_FLICKERING	led is flickering
CO_LED_STATE_FLASH_1	led is flashing mode 1
CO_LED_STATE_FLASH_2	led is flashing mode 2
CO_LED_STATE_FLASH_3	led is flashing mode 3
CO_LED_STATE_BLINKING	led is blinking
CO_LED_STATE_ON	led is on

5.25.4 Function Documentation

5.25.4.1 EXTERN_DECL RET_T coEventRegister_LED_GREEN (CO_EVENT_LED_T *pFunction*)

coEventRegister_LED_GREEN - register for green LED

Returns

RET_T

Parameters

pFunction	pointer to function

5.25.4.2 EXTERN_DECL RET_T coEventRegister_LED_RED (CO_EVENT_LED_T *pFunction*)

coEventRegister_LED_RED - register for red LED

Register application function for controlling of LED state

Returns

RET_T

Parameters

pFunction	pointer to function

5.25.4.3 EXTERN_DECL void coLedSetGreen (CO_LED_STATE_T *newLedState*)

coLedSetGreen - set green led to new state

Set green LED to one of the following state:

- OFF,
- FLICKERING,
- FLASH_1, FLASH_2, FLASH_3,
- BLINKING,
- ON

Returns

none

Parameters

newLedState	new led state

5.25.4.4 EXTERN_DECL void coLedSetRed (CO_LED_STATE_T *newLedState*)

coLedSetRed - set red led to new state

Set led LED to one of the following state: OFF, FLICKERING, FLASH_1, FLASH_2, FLASH_3, BLINKING, ON

Returns

none

Parameters

newLedState	new led state

5.25.4.5 EXTERN_DECL void coLedSetState (CO_LED_STATE_T *newState*, BOOL_T *on*)

coLedSetState - set led state

Set the error led to special state OFF, FLICKERING, FLASH_1, FLASH_2, FLASH_3, BLINKING, ON

All states are saved. Only the highest prior state is displayed. If the highest state is reset, the next state is displayed.

Returns

none

Parameters

newState	new state
on	set state to on/off

5.26 co_lss.c File Reference

LSS slave handling.

Functions

- void coLssNonConfigSlave (void)

  coLssNonConfigSlave - request for unconfigured slaves
- RET_T coEventRegister_LSS (CO_EVENT_LSS_T pFunction)

  coEventRegister_LSS - register LSS event
- RET_T coLssInit (void)

  coLssInit - init LSS functionality

5.26.1 Detailed Description

LSS slave handling.

contains LSS slave services

5.26.2 Function Documentation

5.26.2.1 RET_T coEventRegister_LSS (CO_EVENT_LSS_T *pFunction*)

coEventRegister_LSS - register LSS event

This function registers an indication function for LSS events.

Returns

RET_T

Parameters

pFunction	pointer to function

5.26.2.2 RET_T coLssInit (void)

coLssInit - init LSS functionality

This function initializes the LSS functionality, depending on the define CO_LSS_SLAVE_SUPPORTED or CO_L↩ SS_MASTER_SUPPORTED as slave or master.

Returns

RET_T

5.26.2.3 void coLssNonConfigSlave (void)

coLssNonConfigSlave - request for unconfigured slaves

get answer, if node-id == 255

Returns

none

5.27 co_lss.h File Reference

defines for lss services

Typedefs

- typedef void(∗ CO_EVENT_LSS_T) (CO_LSS_SERVICE_T service, UNSIGNED16 bitrate, UNSIGNED8 ∗pErrCode, UNSIGNED8 ∗pErrSpec)

  function pointer to LSS indication
- typedef void(∗ CO_EVENT_LSS_MASTER_T) (CO_LSS_MASTER_SERVICE_T, UNSIGNED16 errorCode, UNSIGNED8 errorSpec, UNSIGNED32 ∗pIdentity)

  function pointer to LSS master indication

Enumerations

Functions

- EXTERN_DECL RET_T coLssInit (void)

  coLssInit - init LSS functionality
- EXTERN_DECL RET_T coLssMasterInit (void)

  coLssMasterInit - init LSS functionality
- EXTERN_DECL RET_T coEventRegister_LSS (CO_EVENT_LSS_T pFunction)

  coEventRegister_LSS - register LSS event
- EXTERN_DECL RET_T coEventRegister_LSS_MASTER (CO_EVENT_LSS_MASTER_T pFunction)

  coEventRegister_LSS_MASTER - register LSS master event
- EXTERN_DECL RET_T coLssIdentifyNonConfiguredSlaves (UNSIGNED16 timeOutVal, UNSIGNED16 interval)

  coLssIdentifyNonConfiguredSlaves - identify unconfigured remote slaves
- EXTERN_DECL void coLssNonConfigSlave (void)

  coLssNonConfigSlave - request for unconfigured slaves
- EXTERN_DECL RET_T coLssFastScan (UNSIGNED16 timeOutVal)

  coLssFastScan - start fastscan
- EXTERN_DECL RET_T coLssFastScanKnownDevice (UNSIGNED32 vendorId, UNSIGNED32 product↩ Code, UNSIGNED32 versionNr, UNSIGNED32 serNr, UNSIGNED16 timeOutVal)

  coLssFastScanKnownDevice - start fastscan for known device
- EXTERN_DECL RET_T coLssSetNodeId (UNSIGNED8 nodeId, UNSIGNED16 timeOutVal)

  coLssNodeId - set node id for remote node
- EXTERN_DECL RET_T coLssSetBitrate (UNSIGNED16 bitRate, UNSIGNED16 timeOutVal)

  coLssSetBitrate - set bitrate for remote nodes
- EXTERN_DECL RET_T coLssSetBitrateTable (UNSIGNED8 tableSelector, UNSIGNED8 tableIndex, UNS↩ IGNED16 timeOutVal)

  coLssSetBitrate - set bitrate for remote nodes
- EXTERN_DECL RET_T coLssActivateBitrate (UNSIGNED16 switchDelay)

  coLssActivateBitrate - activate bitratenodes
- EXTERN_DECL RET_T coLssSwitchGlobal (CO_LSS_STATE_T mode)

  coLssSwitchGlobal - send global switch command
- EXTERN_DECL RET_T coLssSwitchSelective (UNSIGNED32 vendorId, UNSIGNED32 productCode, UN↩ SIGNED32 versionNr, UNSIGNED32 serNr, UNSIGNED16 timeOutVal)

  coLssSwitchSelective - send Selective switch command
- EXTERN_DECL RET_T coLssStoreConfig (UNSIGNED16 timeOutVal)

  coLssStoreConfig - store configuration
- EXTERN_DECL RET_T coLssInquireNodeId (UNSIGNED16 timeOutVal)

  coLssInquireNodeId - inquire actual node ID
- EXTERN_DECL RET_T coLssInquireIdentity (UNSIGNED8 subIndex, UNSIGNED16 timeOutVal)

  coLssInquireIdentity - inquire identity data
- EXTERN_DECL RET_T coLssIdentifyRemoteSlaves (UNSIGNED32 vendor, UNSIGNED32 productCode, UNSIGNED32 revisionLow, UNSIGNED32 revisionHigh, UNSIGNED32 serialNumberLow, UNSIGNED32 serialNumberHigh, UNSIGNED16 timeOutVal)

  coLssIdentifyRemoteSlaves - identify remote slaves
- EXTERN_DECL void coLssMasterDisable (void)

  coLssMasterDisable - disable LSS master services
- EXTERN_DECL void coLssMasterEnable (void)

  coLssMasterEnable - enable LSS master services
- EXTERN_DECL UNSIGNED32 coLssMasterGetInquireData (void)

  coLssMasterGetInquireData - get requested inquire data

5.27.1 Detailed Description

defines for lss services

- contains defines for lss services

5.27.2 Typedef Documentation

5.27.2.1 typedef void(∗ CO_EVENT_LSS_MASTER_T) (CO_LSS_MASTER_SERVICE_T, UNSIGNED16 errorCode, UNSIGNED8 errorSpec, UNSIGNED32 ∗pIdentity)

function pointer to LSS master indication

Parameters

service	- answer for service LSS_MASTER_SERVICE_xxx
errorCode	== 65535 - timeout
errorCode	== 1..255 - error code
errorCode	== 0 - ok
errorSpec	- error spec (if errorCode == 65365)
pIdentity	== NULL - no data available
pIdentity	=! NULL - pIdentity valid

Returns

void

5.27.2.2 typedef void(∗ CO_EVENT_LSS_T) (CO_LSS_SERVICE_T service, UNSIGNED16 bitrate, UNSIGNED8 ∗pErrCode, UNSIGNED8 ∗pErrSpec)

function pointer to LSS indication

Parameters

service	- answer for service LSS_SERVICE_xxx
bitrate	- new bitrate / pending node id (only for CO_LSS_SERVICE_STORE) 1000, 500, ... 10 standard bitrates 0 autobaud 0 table specific, values in pErrCode and pErrSpec)
pErrCode	- pointer to error code
pErrSpec	- pointer to error spec

Returns

UNSIGNED8

Return values

0	- success

Return values

1	- store not supported
2	- media access error
255	- implementation specific (value in parameter pErr)

5.27.3 Enumeration Type Documentation

5.27.3.1 enum CO_LSS_MASTER_SERVICE_T

LSS master services for indication functions.

Enumerator

CO_LSS_MASTER_SERVICE_NON_CONFIG_SLAVE	LSS Master service non-config slave
CO_LSS_MASTER_SERVICE_SET_NODEID	LSS Master service set node id
CO_LSS_MASTER_SERVICE_SET_BITRATE	LSS Master service set bitrate
CO_LSS_MASTER_SERVICE_FASTSCAN	LSS Master service fastscan
CO_LSS_MASTER_SERVICE_STORE	LSS Master service store
CO_LSS_MASTER_SERVICE_INQUIRE_NODEID	LSS Master service inquire node
CO_LSS_MASTER_SERVICE_INQUIRE_VENDOR	LSS Master service inquire vendor
CO_LSS_MASTER_SERVICE_INQUIRE_PRODUCT	LSS Master service inquire product
CO_LSS_MASTER_SERVICE_INQUIRE_REVISION	LSS Master service inquire revision
CO_LSS_MASTER_SERVICE_INQUIRE_SERIAL	LSS Master service inquire serial
CO_LSS_MASTER_SERVICE_BITRATE_OFF	LSS Master service indication bitrate off
CO_LSS_MASTER_SERVICE_BITRATE_SET	LSS Master service indication set new bitrate
CO_LSS_MASTER_SERVICE_BITRATE_ACTIVE	LSS Master service indication bitrate active
CO_LSS_MASTER_SERVICE_SWITCH_SELECTIVE	LSS Master service switch selektive
CO_LSS_MASTER_SERVICE_IDENTITY	LSS Master service indentity
CO_LSS_MASTER_SERVICE_SWITCH_GLOBAL	LSS Master switch global

5.27.3.2 enum CO_LSS_SERVICE_T

LSS slave services for indication functions.

Enumerator

CO_LSS_SERVICE_STORE	LSS service indication store node id
CO_LSS_SERVICE_NEW_BITRATE	LSS service indication new bitrate
CO_LSS_SERVICE_BITRATE_OFF	LSS service indication bitrate off
CO_LSS_SERVICE_BITRATE_SET	LSS service indication set new bitrate
CO_LSS_SERVICE_BITRATE_ACTIVE	LSS service indication bitrate active
CO_LSS_SERVICE_NEW_NODE_ID	LSS service indication new node-id

5.27.3.3 enum CO_LSS_STATE_T

LSS slave states.

Enumerator

CO_LSS_STATE_WAITING	LSS state waiting
CO_LSS_STATE_CONFIGURATION	LSS state configuration

5.27.4 Function Documentation

5.27.4.1 EXTERN_DECL RET_T coEventRegister_LSS (CO_EVENT_LSS_T *pFunction*)

coEventRegister_LSS - register LSS event

This function registers an indication function for LSS events.

Returns

RET_T

Parameters

pFunction	pointer to function

5.27.4.2 EXTERN_DECL RET_T coEventRegister_LSS_MASTER (CO_EVENT_LSS_MASTER_T *pFunction*)

coEventRegister_LSS_MASTER - register LSS master event

This function registers an indication function for LSS Master events.

Returns

RET_T

Parameters

pFunction	pointer to function

5.27.4.3 EXTERN_DECL RET_T coLssActivateBitrate (UNSIGNED16 *switchDelay*)

coLssActivateBitrate - activate bitratenodes

Start service activate bitrate for remote and local node. The function transmits the command to the remote slaves and start the timer for bitrate switch itself. After the time was elapsed, the indication is called.

Returns

RET_T

Parameters

switchDelay	switch delay time

5.27.4.4 EXTERN_DECL RET_T coLssFastScan (UNSIGNED16 *timeOutVal*)

coLssFastScan - start fastscan

start fastscan for the given parameter if no node was found, the indication will be return 0

Returns

RET_T

Parameters

timeOutVal	timeout value in msec

5.27.4.5 EXTERN_DECL RET_T coLssFastScanKnownDevice (UNSIGNED32 *vendorId,* UNSIGNED32 *productCode,* UNSIGNED32 *versionNr,* UNSIGNED32 *serNr,* UNSIGNED16 *timeOutVal*)

coLssFastScanKnownDevice - start fastscan for known device

start fastscan for a well known device if no node was found, the indication will be return 0

Returns

RET_T

Parameters

vendorId	vendor number
productCode	product code
versionNr	version number
serNr	serial number
timeOutVal	timeout value in msec

5.27.4.6 EXTERN_DECL RET_T coLssIdentifyNonConfiguredSlaves (UNSIGNED16 *timeOutVal,* UNSIGNED16 *interval*)

coLssIdentifyNonConfiguredSlaves - identify unconfigured remote slaves

Identify unconfigured remote slaves by sending the LSS command. If no slave is available, the indication function is called after the time is up, given by the parameter timeoutvalue.

The LSS command is transmitted in a cycle of n seconds, given by parameter interval. If the parameter is 0, the LSS command is transmitted only once.

Returns

RET_T

Parameters

timeOutVal	timeout value in msec
interval	interval in seconds

5.27.4.7 EXTERN_DECL RET_T coLssIdentifyRemoteSlaves (UNSIGNED32 *vendorId,* UNSIGNED32 *productCode,* UNSIGNED32 *revisionLow,* UNSIGNED32 *revisionHigh,* UNSIGNED32 *serialNumberLow,* UNSIGNED32 *serialNumberHigh,* UNSIGNED16 *timeOutVal*)

coLssIdentifyRemoteSlaves - identify remote slaves

Identify remote slaves by sending the LSS command with the given identity parameter. If no slave is available, the indication function is called after the time is up, given by the parameter timeoutvalue.

Returns

RET_T

Parameters

vendorId	vendor id
productCode	product code
revisionLow	revision low
revisionHigh	revision high
serialNumberLow	serialNumber low
serialNumberHigh	serialNumber high
timeOutVal	timeout value in msec

5.27.4.8 EXTERN_DECL RET_T coLssInit (void)

coLssInit - init LSS functionality

This function initializes the LSS functionality, depending on the define CO_LSS_SLAVE_SUPPORTED or CO_L↩ SS_MASTER_SUPPORTED as slave or master.

Returns

RET_T

5.27.4.9 EXTERN_DECL RET_T coLssInquireIdentity (UNSIGNED8 *subIndex*, UNSIGNED16 *timeOutVal*)

coLssInquireIdentity - inquire identity data

Send the inquire identity command

Please note - the indication function called after the request is finished only indicates an erroronous or error free execution of the request. It doesn't delivered the requested data. To get the requested data the function coLss↩ MasterGetInquireData() have to be used.

Returns

RET_T

Parameters

subIndex	subIndex of requested identity parameter (1..4)
timeOutVal	timeout value in msec

5.27.4.10 EXTERN_DECL RET_T coLssInquireNodeId (UNSIGNED16 *timeOutVal*)

coLssInquireNodeId - inquire actual node ID

Send the inquire node id command

Returns

RET_T

Parameters

timeOutVal	timeout value in msec

5.27.4.11 EXTERN_DECL void coLssMasterDisable (void)

coLssMasterDisable - disable LSS master services

Returns

none

5.27.4.12 EXTERN_DECL void coLssMasterEnable (void)

coLssMasterEnable - enable LSS master services

(Re)enable LSS master services after the was disabled by coLssMasterDisable()

Returns

none

5.27.4.13 EXTERN_DECL UNSIGNED32 coLssMasterGetInquireData (void)

coLssMasterGetInquireData - get requested inquire data

This function returns the requested data for a inquire request started by coLssInquireIdentity() before. The data are only valid, if the indication function with the parameter given to coLssInquireIdentity() was indicated before without any error.

Returns

UNSIGNED32 identity value

5.27.4.14 EXTERN_DECL RET_T coLssMasterInit (void)

coLssMasterInit - init LSS functionality

This function initializes the LSS functionality, depending on the define CO_LSS_SLAVE_SUPPORTED or CO_L↩
SS_MASTER_SUPPORTED as slave or master.

Returns

RET_T

5.27.4.15 EXTERN_DECL void coLssNonConfigSlave (void)

coLssNonConfigSlave - request for unconfigured slaves

get answer, if node-id == 255

Returns

none

5.27.4.16 EXTERN_DECL RET_T coLssSetBitrate (UNSIGNED16 *bitRate*, UNSIGNED16 *timeOutVal*)

coLssSetBitrate - set bitrate for remote nodes

Send a new bitrate to an remote slaves. Allowed bitrates are: 1000, 800, 500, 250, 125, 50, 20, 10, 0 (for autobaud)

The nodes have to be set before in configuration mode

Returns

RET_T

Parameters

bitRate	new bitrate
timeOutVal	timeout value in msec

5.27.4.17 EXTERN_DECL RET_T coLssSetBitrateTable (UNSIGNED8 *tableSelector,* UNSIGNED8 *tableIndex,* UNSIGNED16 *timeOutVal*)

coLssSetBitrate - set bitrate for remote nodes

Send a new bitrate to an remote slaves. Parameter are not checked! The nodes have to be set before in configuration mode

Returns

RET_T

Parameters

tableSelector	table selector
tableIndex	table index
timeOutVal	timeout value in msec

5.27.4.18 EXTERN_DECL RET_T coLssSetNodeId (UNSIGNED8 *nodeId,* UNSIGNED16 *timeOutVal*)

coLssNodeId - set node id for remote node

Send a new node id to an remote slave The node has to be set before in configuration mode

Returns

RET_T

Parameters

nodeId	new node id
timeOutVal	timeout value in msec

5.27.4.19 EXTERN_DECL RET_T coLssStoreConfig (UNSIGNED16 *timeOutVal*)

coLssStoreConfig - store configuration

Send the LSS store configuration command to a slave.

Returns

RET_T

Parameters

timeOutVal	timeout value in msec

5.27.4.20 EXTERN_DECL RET_T coLssSwitchGlobal (CO_LSS_STATE_T *mode*)

coLssSwitchGlobal - send global switch command

Send the global switch command - no answer expected

Returns

RET_T

Parameters

mode	mode

5.27.4.21 EXTERN_DECL RET_T coLssSwitchSelective (UNSIGNED32 *vendorId,* UNSIGNED32 *productCode,* UNSIGNED32 *versionNr,* UNSIGNED32 *serNr,* UNSIGNED16 *timeOutVal*)

coLssSwitchSelective - send Selective switch command

Send the Selective switch command - the detected node should be answer and go into CONFIGURATION mode

Returns

RET_T

Parameters

vendorId	vendor number
productCode	product code
versionNr	version number
serNr	serial number
timeOutVal	timeout value in msec

5.28 co_lssmaster.c File Reference

LSS master handling.

Functions

- RET_T coLssIdentifyNonConfiguredSlaves (UNSIGNED16 timeOutVal, UNSIGNED16 interval)

  coLssIdentifyNonConfiguredSlaves - identify unconfigured remote slaves
- RET_T coLssIdentifyRemoteSlaves (UNSIGNED32 vendorId, UNSIGNED32 productCode, UNSIGNED32 revisionLow, UNSIGNED32 revisionHigh, UNSIGNED32 serialNumberLow, UNSIGNED32 serialNumber←High, UNSIGNED16 timeOutVal)

  coLssIdentifyRemoteSlaves - identify remote slaves
- RET_T coLssFastScan (UNSIGNED16 timeOutVal)

coLssFastScan - start fastscan
- RET_T coLssFastScanKnownDevice (UNSIGNED32 vendorId, UNSIGNED32 productCode, UNSIGNED32 versionNr, UNSIGNED32 serNr, UNSIGNED16 timeOutVal)

  coLssFastScanKnownDevice - start fastscan for known device
- RET_T coLssSetNodeId (UNSIGNED8 nodeId, UNSIGNED16 timeOutVal)

  coLssNodeId - set node id for remote node
- RET_T coLssSetBitrate (UNSIGNED16 bitRate, UNSIGNED16 timeOutVal)

  coLssSetBitrate - set bitrate for remote nodes
- RET_T coLssSetBitrateTable (UNSIGNED8 tableSelector, UNSIGNED8 tableIndex, UNSIGNED16 time↩OutVal)

  coLssSetBitrate - set bitrate for remote nodes
- RET_T coLssActivateBitrate (UNSIGNED16 switchDelay)

  coLssActivateBitrate - activate bitratenodes
- RET_T coLssStoreConfig (UNSIGNED16 timeOutVal)

  coLssStoreConfig - store configuration
- RET_T coLssSwitchGlobal (CO_LSS_STATE_T mode)

  coLssSwitchGlobal - send global switch command
- RET_T coLssSwitchSelective (UNSIGNED32 vendorId, UNSIGNED32 productCode, UNSIGNED32 versionNr, UNSIGNED32 serNr, UNSIGNED16 timeOutVal)

  coLssSwitchSelective - send Selective switch command
- RET_T coLssInquireNodeId (UNSIGNED16 timeOutVal)

  coLssInquireNodeId - inquire actual node ID
- RET_T coLssInquireIdentity (UNSIGNED8 subIndex, UNSIGNED16 timeOutVal)

  coLssInquireIdentity - inquire identity data
- RET_T coEventRegister_LSS_MASTER (CO_EVENT_LSS_MASTER_T pFunction)

  coEventRegister_LSS_MASTER - register LSS master event
- UNSIGNED32 coLssMasterGetInquireData (void)

  coLssMasterGetInquireData - get requested inquire data
- void coLssMasterDisable (void)

  coLssMasterDisable - disable LSS master services
- void coLssMasterEnable (void)

  coLssMasterEnable - enable LSS master services
- RET_T coLssMasterInit (void)

  coLssMasterInit - init LSS functionality

5.28.1 Detailed Description

LSS master handling.

contains LSS master services

5.28.2 Function Documentation

5.28.2.1 RET_T coEventRegister_LSS_MASTER (CO_EVENT_LSS_MASTER_T *pFunction*)

coEventRegister_LSS_MASTER - register LSS master event

This function registers an indication function for LSS Master events.

Returns

RET_T

Parameters

pFunction	pointer to function

5.28.2.2 RET_T coLssActivateBitrate (UNSIGNED16 *switchDelay*)

coLssActivateBitrate - activate bitratenodes

Start service activate bitrate for remote and local node. The function transmits the command to the remote slaves and start the timer for bitrate switch itself. After the time was elapsed, the indication is called.

Returns

RET_T

Parameters

switchDelay	switch delay time

5.28.2.3 RET_T coLssFastScan (UNSIGNED16 *timeOutVal*)

coLssFastScan - start fastscan

start fastscan for the given parameter if no node was found, the indication will be return 0

Returns

RET_T

Parameters

timeOutVal	timeout value in msec

5.28.2.4 RET_T coLssFastScanKnownDevice (UNSIGNED32 *vendorId,* UNSIGNED32 *productCode,* UNSIGNED32 *versionNr,* UNSIGNED32 *serNr,* UNSIGNED16 *timeOutVal*)

coLssFastScanKnownDevice - start fastscan for known device

start fastscan for a well known device if no node was found, the indication will be return 0

Returns

RET_T

Parameters

vendorId	vendor number
productCode	product code
versionNr	version number
serNr	serial number
timeOutVal	timeout value in msec

5.28.2.5 RET_T coLssIdentifyNonConfiguredSlaves (UNSIGNED16 *timeOutVal,* UNSIGNED16 *interval*)

coLssIdentifyNonConfiguredSlaves - identify unconfigured remote slaves

Identify unconfigured remote slaves by sending the LSS command. If no slave is available, the indication function is called after the time is up, given by the parameter timeoutvalue.

The LSS command is transmitted in a cycle of n seconds, given by parameter interval. If the parameter is 0, the LSS command is transmitted only once.

Returns

RET_T

Parameters

timeOutVal	timeout value in msec
interval	interval in seconds

5.28.2.6 RET_T coLssIdentifyRemoteSlaves (UNSIGNED32 *vendorId,* UNSIGNED32 *productCode,* UNSIGNED32 *revisionLow,* UNSIGNED32 *revisionHigh,* UNSIGNED32 *serialNumberLow,* UNSIGNED32 *serialNumberHigh,* UNSIGNED16 *timeOutVal*)

coLssIdentifyRemoteSlaves - identify remote slaves

Identify remote slaves by sending the LSS command with the given identity parameter. If no slave is available, the indication function is called after the time is up, given by the parameter timeoutvalue.

Returns

RET_T

Parameters

vendorId	vendor id
productCode	product code
revisionLow	revision low
revisionHigh	revision high
serialNumberLow	serialNumber low
serialNumberHigh	serialNumber high
timeOutVal	timeout value in msec

5.28.2.7 RET_T coLssInquireIdentity (UNSIGNED8 *subIndex*, UNSIGNED16 *timeOutVal*)

coLssInquireIdentity - inquire identity data

Send the inquire identity command

Please note - the indication function called after the request is finished only indicates an erroronous or error free execution of the request. It doesn't delivered the requested data. To get the requested data the function coLss↵
MasterGetInquireData() have to be used.

Returns

RET_T

Parameters

subIndex	subIndex of requested identity parameter (1..4)
timeOutVal	timeout value in msec

5.28.2.8 RET_T coLssInquireNodeId (UNSIGNED16 *timeOutVal*)

coLssInquireNodeId - inquire actual node ID

Send the inquire node id command

Returns

RET_T

Parameters

timeOutVal	timeout value in msec

5.28.2.9 void coLssMasterDisable (void)

coLssMasterDisable - disable LSS master services

Returns

none

5.28.2.10 void coLssMasterEnable (void)

coLssMasterEnable - enable LSS master services

(Re)enable LSS master services after the was disabled by coLssMasterDisable()

Returns

none

5.28.2.11 UNSIGNED32 coLssMasterGetInquireData (void)

coLssMasterGetInquireData - get requested inquire data

This function returns the requested data for a inquire request started by coLssInquireIdentity() before. The data are only valid, if the indication function with the parameter given to coLssInquireIdentity() was indicated before without any error.

Returns

UNSIGNED32 identity value

5.28.2.12 RET_T coLssMasterInit (void)

coLssMasterInit - init LSS functionality

This function initializes the LSS functionality, depending on the define CO_LSS_SLAVE_SUPPORTED or CO_L↩
SS_MASTER_SUPPORTED as slave or master.

Returns

RET_T

5.28.2.13 RET_T coLssSetBitrate (UNSIGNED16 *bitRate,* UNSIGNED16 *timeOutVal*)

coLssSetBitrate - set bitrate for remote nodes

Send a new bitrate to an remote slaves. Allowed bitrates are: 1000, 800, 500, 250, 125, 50, 20, 10, 0 (for autobaud)

The nodes have to be set before in configuration mode

Returns

RET_T

Parameters

bitRate	new bitrate
timeOutVal	timeout value in msec

5.28.2.14 RET_T coLssSetBitrateTable (UNSIGNED8 *tableSelector,* UNSIGNED8 *tableIndex,* UNSIGNED16 *timeOutVal*)

coLssSetBitrate - set bitrate for remote nodes

Send a new bitrate to an remote slaves. Parameter are not checked! The nodes have to be set before in configuration mode

Returns

RET_T

Parameters

tableSelector	table selector
tableIndex	table index
timeOutVal	timeout value in msec

5.28.2.15 RET_T coLssSetNodeId (UNSIGNED8 *nodeId,* UNSIGNED16 *timeOutVal*)

coLssNodeId - set node id for remote node

Send a new node id to an remote slave The node has to be set before in configuration mode

Returns

RET_T

Parameters

nodeId	new node id
timeOutVal	timeout value in msec

5.28.2.16 RET_T coLssStoreConfig (UNSIGNED16 *timeOutVal*)

coLssStoreConfig - store configuration

Send the LSS store configuration command to a slave.

Returns

RET_T

Parameters

timeOutVal	timeout value in msec

5.28.2.17 RET_T coLssSwitchGlobal (CO_LSS_STATE_T *mode*)

coLssSwitchGlobal - send global switch command

Send the global switch command - no answer expected

Returns

RET_T

Parameters

mode	mode

5.28.2.18 RET_T coLssSwitchSelective (UNSIGNED32 *vendorId,* UNSIGNED32 *productCode,* UNSIGNED32 *versionNr,* UNSIGNED32 *serNr,* UNSIGNED16 *timeOutVal*)

coLssSwitchSelective - send Selective switch command

Send the Selective switch command - the detected node should be answer and go into CONFIGURATION mode

Returns

RET_T

Parameters

vendorId	vendor number
productCode	product code
versionNr	version number
serNr	serial number
timeOutVal	timeout value in msec

5.29 co_manager.c File Reference

Manager handling according to CiA 302-2.

Functions

- RET_T coManagerStart (void)

  coManagerStart
- RET_T coManagerContinueSwUpdate (UNSIGNED8 slave, RET_T result)

  coManagerContinueSwUpdate - continue after software update
- RET_T coManagerContinueConfigUpdate (UNSIGNED8 slave, RET_T result)

  coManagerContinueConfigUpdate - continue configuration
- RET_T coManagerContinueOperational (void)

  coManagerContinueOperational - continue to OPERATIONAL
- RET_T coEventRegister_MANAGER_BOOTUP (CO_EVENT_MANAGER_BOOTUP_T pFunction)

  coEventRegister_MANAGER_BOOTUP - register MANAGER_BOOTUP event

5.29.1 Detailed Description

Manager handling according to CiA 302-2.

contains CANopen Manager handling according to CiA 302-2

5.29.2 Function Documentation

5.29.2.1 RET_T coEventRegister_MANAGER_BOOTUP (CO_EVENT_MANAGER_BOOTUP_T *pFunction*)

coEventRegister_MANAGER_BOOTUP - register MANAGER_BOOTUP event

register indication function for MANAGER_BOOTUP Procedure events

Returns

RET_T

Parameters

pFunction	pointer to function

5.29.2.2 RET_T coManagerContinueConfigUpdate (UNSIGNED8 *slave,* RET_T *result*)

coManagerContinueConfigUpdate - continue configuration

This function has to be called from application after configuration for the given node was finished The result should be RET_OK or another error code.

If state of this node is not in correct state, the function returns RET_INVALID_PARAMETER

Parameters

slave	slave (1.. 127)
result	result of configuration process

5.29.2.3 RET_T coManagerContinueOperational (void)

coManagerContinueOperational - continue to OPERATIONAL

This function continues Bootup Procedure to state OPERATIONAL, if the start bit at 0x1f80 is not set. The application can start the nodes itself, or call this function to do that.

If state of this node is not in correct state, the function returns RET_INVALID_PARAMETER

5.29.2.4 RET_T coManagerContinueSwUpdate (UNSIGNED8 *slave,* RET_T *result*)

coManagerContinueSwUpdate - continue after software update

This function continues startup for the given node after software was updates by application

If state of this node is not in correct state, the function returns RET_INVALID_PARAMETER

Parameters

slave	slave
result	result of software update

5.29.2.5 RET_T coManagerStart (void)

coManagerStart

This function starts the CANopen manager process. All necessary parameter for mandatory and optional slaves has to be available at the object dictionary.

Returns

RET_T

5.30 co_manager.h File Reference

defines for bootup manager services

Typedefs

- typedef void(∗ CO_EVENT_MANAGER_BOOTUP_T) (UNSIGNED8, CO_MANAGER_EVENT_T)

  function pointer to NMT event function

Enumerations

Functions

- EXTERN_DECL RET_T coEventRegister_MANAGER_BOOTUP (CO_EVENT_MANAGER_BOOTUP_T pFunction)

  coEventRegister_MANAGER_BOOTUP - register MANAGER_BOOTUP event
- EXTERN_DECL RET_T coManagerStart (void)

  coManagerStart
- EXTERN_DECL RET_T coManagerContinueSwUpdate (UNSIGNED8 slave, RET_T result)

  coManagerContinueSwUpdate - continue after software update
- EXTERN_DECL RET_T coManagerContinueConfigUpdate (UNSIGNED8 slave, RET_T result)

  coManagerContinueConfigUpdate - continue configuration
- EXTERN_DECL RET_T coManagerContinueOperational (void)

  coManagerContinueOperational - continue to OPERATIONAL

5.30.1 Detailed Description

defines for bootup manager services

- contains defines for bootup manager services

5.30.2 Typedef Documentation

5.30.2.1 typedef void(∗ CO_EVENT_MANAGER_BOOTUP_T) (UNSIGNED8, CO_MANAGER_EVENT_T)

function pointer to NMT event function

Parameters

node	- node id for event (0 = manager event)
event	- event type from CO_MANAGER_EVENT_T

Returns

void

5.30.3 Enumeration Type Documentation

5.30.3.1 enum CO_MANAGER_EVENT_T

manager bootup events

Enumerator

CO_MANAGER_EVENT_BOOT	node x start boot
CO_MANAGER_EVENT_ERROR_B	node x read 0x1000 failed
CO_MANAGER_EVENT_ERROR_C	node x check device type failed
CO_MANAGER_EVENT_ERROR_D	node x check vendor id type failed
CO_MANAGER_EVENT_ERROR_J	node x check configuration failed
CO_MANAGER_EVENT_ERROR_G	node x update configuration failed
CO_MANAGER_EVENT_ERROR_K	node x start errctl failed
CO_MANAGER_EVENT_ERROR_M	node x check product code failed
CO_MANAGER_EVENT_ERROR_N	node x check version nr failed
CO_MANAGER_EVENT_ERROR_O	node x check serial nr failed
CO_MANAGER_EVENT_UPDATE_SW	node x software update necessary
CO_MANAGER_EVENT_UPDATE_CONFIG	node x update config necessary
CO_MANAGER_EVENT_BOOTED	node x boot successfully
CO_MANAGER_EVENT_ERROR_NODE	node x failure heartbeat
CO_MANAGER_EVENT_RDY_OPERATIONAL	manager ready for OPERATIONAL
CO_MANAGER_EVENT_FAILURE	bootup failure about mandatory slave
CO_MANAGER_EVENT_FINISHED	bootup finished without errors

5.30.4 Function Documentation

5.30.4.1 EXTERN_DECL RET_T coEventRegister_MANAGER_BOOTUP (CO_EVENT_MANAGER_BOOTUP_T *pFunction*)

coEventRegister_MANAGER_BOOTUP - register MANAGER_BOOTUP event

register indication function for MANAGER_BOOTUP Procedure events

Returns

RET_T

Parameters

pFunction	pointer to function

5.30.4.2 EXTERN_DECL RET_T coManagerContinueConfigUpdate (UNSIGNED8 *slave*, RET_T *result*)

coManagerContinueConfigUpdate - continue configuration

This function has to be called from application after configuration for the given node was finished The result should be RET_OK or another error code.

If state of this node is not in correct state, the function returns RET_INVALID_PARAMETER

Parameters

slave	slave (1.. 127)
result	result of configuration process

5.30.4.3 EXTERN_DECL RET_T coManagerContinueOperational (void)

coManagerContinueOperational - continue to OPERATIONAL

This function continues Bootup Procedure to state OPERATIONAL, if the start bit at 0x1f80 is not set. The application can start the nodes itself, or call this function to do that.

If state of this node is not in correct state, the function returns RET_INVALID_PARAMETER

5.30.4.4 EXTERN_DECL RET_T coManagerContinueSwUpdate (UNSIGNED8 *slave*, RET_T *result*)

coManagerContinueSwUpdate - continue after software update

This function continues startup for the given node after software was updates by application

If state of this node is not in correct state, the function returns RET_INVALID_PARAMETER

Parameters

slave	slave
result	result of software update

5.30.4.5 EXTERN_DECL RET_T coManagerStart (void)

coManagerStart

This function starts the CANopen manager process. All necessary parameter for mandatory and optional slaves has to be available at the object dictionary.

Returns

RET_T

5.31 co_mpdo.c File Reference

MPDO handling.

5.31.1 Detailed Description

MPDO handling.

contains MPDO services

5.32 co_network.c File Reference

multi level networking handling

Functions

- RET_T coNetworkGet (UNSIGNED16 network, UNSIGNED8 ∗pNetworkIf, UNSIGNED8 ∗pRouterNode)

  coNetworkGet - get network interface and router node
- UNSIGNED16 icoNetworkLocalId (void)

  icoNetworkLocal - get local network id

5.32.1 Detailed Description

multi level networking handling

contains multi level network services

5.32.2 Function Documentation

5.32.2.1 RET_T coNetworkGet (UNSIGNED16 *network*, UNSIGNED8 ∗ *pNetworkIf*, UNSIGNED8 ∗ *pRouterNode*)

coNetworkGet - get network interface and router node

Get Network interface und router node for requested destination network from object 0x1f2c

Parameters

network	destination network
pNetworkIf	network interface number
pRouterNode	router node id

5.33 co_network.h File Reference

defines for network services

Typedefs

- typedef UNSIGNED8(∗ CO_EVENT_GW_SDOCLIENT_FCT_T) (UNSIGNED16 network, UNSIGNED8 node, UNSIGNED32 ∗pCobClSrv, UNSIGNED32 ∗pCobSrvCl)

  function pointer to get sdo channel number

Functions

- EXTERN_DECL RET_T coNetworkGet (UNSIGNED16 network, UNSIGNED8 ∗pNetworkIf, UNSIGNED8 ∗pRouterNode)

  coNetworkGet - get network interface and router node

5.33.1 Detailed Description

defines for network services

- contains defines for network services

5.33.2 Typedef Documentation

5.33.2.1 typedef UNSIGNED8(∗ CO_EVENT_GW_SDOCLIENT_FCT_T) (UNSIGNED16 network, UNSIGNED8 node, UNSIGNED32 ∗pCobClSrv, UNSIGNED32 ∗pCobSrvCl)

function pointer to get sdo channel number

Parameters

network	- target network
node	- target nodeid
pCobClSrv	- pointer for cob-id client server (0 - use default)
pCob↩ SrvCl	- pointer for cob-id server client (0 - use default)

Returns

SDO channel

5.33.3 Function Documentation

5.33.3.1 EXTERN_DECL RET_T coNetworkGet (UNSIGNED16 *network,* UNSIGNED8 ∗ *pNetworkIf,* UNSIGNED8 ∗ *pRouterNode*)

coNetworkGet - get network interface and router node

Get Network interface und router node for requested destination network from object 0x1f2c

Parameters

network	destination network
pNetworkIf	network interface number
pRouterNode	router node id

5.34 co_nmt.c File Reference

Network Managment(NMT) handler.

Functions

- RET_T coEventRegister_NMT (CO_EVENT_NMT_T pFunction)
  coEventRegister_NMT - register NMT event
- UNSIGNED8 coNmtGetNodeId (void)
  coNmtGetNodeId - returns actual node id
- CO_NMT_STATE_T coNmtGetState (void)
  coGetNmtState - returns current NMT state
- RET_T coNmtLocalStateReq (CO_NMT_STATE_T reqState)
  coNmtLocalStateReq - request local NMT state change
- RET_T coNmtInit (UNSIGNED8 master)
  coInitNmt - init NMT functionality

5.34.1 Detailed Description

Network Managment(NMT) handler.

contains routines for NMT handling

5.34.2 Function Documentation

5.34.2.1 RET_T coEventRegister_NMT (CO_EVENT_NMT_T *pFunction*)

coEventRegister_NMT - register NMT event

register indication function for NMT events

Returns

RET_T

Parameters

pFunction	pointer to function

5.34.2.2 UNSIGNED8 coNmtGetNodeId (void)

coNmtGetNodeId - returns actual node id

Returns

node-id

5.34.2.3 CO_NMT_STATE_T coNmtGetState (void)

coGetNmtState - returns current NMT state

This function returns the current NMT state of the local node.

Returns

NMT state

5.34.2.4 RET_T coNmtInit (UNSIGNED8 *master*)

coInitNmt - init NMT functionality

This function initializes the NMT functionality and calls an internal reset communication.

If parameter master is unequal 0 the node will be initialized as NMT master. If flying master is enabled, the decision for master/slave or flying master is done by check object 0x1f80. (parameter master is not used!) In this case, the node starts as slave and wait for the master negotiation.

Returns

RET_T

Parameters

master	master mode

5.34.2.5 RET_T coNmtLocalStateReq (CO_NMT_STATE_T *reqState*)

coNmtLocalStateReq - request local NMT state change

Be carfule - NMT commands should be generated only by the master

Returns

RET_T

Parameters

reqState	new requested state

5.35 co_nmt.h File Reference

defines for nmt services

Typedefs

- typedef UNSIGNED8(∗ CO_NODE_ID_T) (void)

  function pointer to get node id function This function shall get the node id for the device
- typedef void(∗ CO_EVENT_ERRCTRL_T) (UNSIGNED8, CO_ERRCTRL_T, CO_NMT_STATE_T)

  function pointer to error control event function
- typedef RET_T(∗ CO_EVENT_NMT_T) (BOOL_T, CO_NMT_STATE_T)

  function pointer to NMT event function

Enumerations

Functions

- EXTERN_DECL RET_T coNmtInit (UNSIGNED8)

  coInitNmt - init NMT functionality
- EXTERN_DECL UNSIGNED8 coNmtGetNodeId (void)

  coNmtGetNodeId - returns actual node id
- EXTERN_DECL BOOL_T coNmtInhibitActive (void)

  icoNmtInhibitActive - check if inhibit is active
- EXTERN_DECL RET_T coErrorCtrlInit (UNSIGNED16, UNSIGNED8)

  coInitNmt - init error control
- EXTERN_DECL RET_T coEventRegister_ERRCTRL (CO_EVENT_ERRCTRL_T pFunction)

  coEventRegister_ERRCTRL - register error control event
- EXTERN_DECL RET_T coEventRegister_NMT (CO_EVENT_NMT_T pFunction)

  coEventRegister_NMT - register NMT event
- EXTERN_DECL CO_NMT_STATE_T coNmtGetState (void)

  coGetNmtState - returns current NMT state
- EXTERN_DECL CO_NMT_STATE_T coNmtGetRemoteNodeState (UNSIGNED8 nodeId)

  coNmtGetRemoteNodeState - get remote node state
- EXTERN_DECL RET_T coNmtStateReq (UNSIGNED8 node, CO_NMT_STATE_T reqState, BOOL_T master)

  coNmtStateReq - request NMT state change
- EXTERN_DECL RET_T coNmtLocalStateReq (CO_NMT_STATE_T reqState)

  coNmtLocalStateReq - request local NMT state change
- EXTERN_DECL BOOL_T coNmtNodeIsMaster (void)

coNmtNodeIsMaster - detect if node is master
- EXTERN_DECL RET_T coHbConsumerSet (UNSIGNED8 node, UNSIGNED16 hbTime)

  coHbConsumerSet - setup heartbeat consumer
- EXTERN_DECL RET_T coHbConsumerStart (UNSIGNED8 node)

  coHbConsumerStart - start heartbeat consumer monitoring
- EXTERN_DECL RET_T coGuardingMasterStart (UNSIGNED8 node)

  coGuardingMasterStart - start master node guarding
- EXTERN_DECL RET_T coGuardingMasterStop (UNSIGNED8 node)

  coGuardingMasterStop - stop master node guarding

5.35.1 Detailed Description

defines for nmt services

- contains defines for nmt services

5.35.2 Typedef Documentation

5.35.2.1 typedef void(∗ CO_EVENT_ERRCTRL_T) (UNSIGNED8, CO_ERRCTRL_T, CO_NMT_STATE_T)

function pointer to error control event function

Parameters

nodeId	- node Id
errCtrlState	- error control state
nmtState	- actual NMT state

Returns

void

5.35.2.2 typedef RET_T(∗ CO_EVENT_NMT_T) (BOOL_T, CO_NMT_STATE_T)

function pointer to NMT event function

Parameters

execute	- execute status change y/n
nmtState	- new NMT state

Returns

RET_T

Return values

RET_OK	- state change allowed (only valid for OPERATIONAL)
*RET_*	- state change not allowed (only valid for OPERATIONAL)

5.35.2.3 typedef UNSIGNED8(∗ CO_NODE_ID_T) (void)

function pointer to get node id function This function shall get the node id for the device

Returns

node id

5.35.3 Enumeration Type Documentation

5.35.3.1 enum CO_ERRCTRL_T

error control states

Enumerator

CO_ERRCTRL_BOOTUP	bootup
CO_ERRCTRL_NEW_STATE	NMT state changed
CO_ERRCTRL_HB_STARTED	heartbeat started
CO_ERRCTRL_HB_FAILED	heartbeat failed
CO_ERRCTRL_GUARD_FAILED	Lifetime failure from master detected
CO_ERRCTRL_MGUARD_TOGGLE	Master guarding toggle failure detected
CO_ERRCTRL_MGUARD_FAILED	Master guarding failure detected
CO_ERRCTRL_BOOTUP_FAILURE	error transmit bootup
CO_ERRCTRL_DOUBLE_ID	error double node id received

5.35.3.2 enum CO_NMT_REQ_STATE_T

NMT REQ states

Enumerator

CO_NMT_REQ_STATE_STOPPED	STOPPED
CO_NMT_REQ_STATE_OPERATIONAL	OPERATIONAL
CO_NMT_REQ_STATE_RESET_NODE	Reset NODE
CO_NMT_REQ_STATE_RESET_COMM	Reset Communication
CO_NMT_REQ_STATE_PREOP	PRE-OPERATIONAL

5.35.3.3 enum CO_NMT_STATE_T

NMT states

Enumerator

CO_NMT_STATE_UNKNOWN	unknown
CO_NMT_STATE_OPERATIONAL	OPERATIONAL
CO_NMT_STATE_STOPPED	STOPPED
CO_NMT_STATE_PREOP	PRE-OPERATIONAL
CO_NMT_STATE_RESET_NODE	Reset NODE
CO_NMT_STATE_RESET_COMM	Reset Communication

5.35.4 Function Documentation

5.35.4.1 EXTERN_DECL RET_T coErrorCtrlInit (UNSIGNED16 *hbTime,* UNSIGNED8 *hbConsCnt*)

coInitNmt - init error control

Setup error control handling for local node (transmit heartbeat) and remote node (heartbeat monitoring)

Returns

RET_T

Parameters

hbTime	heartbeat producer time
hbConsCnt	heartbeat consumer count

5.35.4.2 EXTERN_DECL RET_T coEventRegister_ERRCTRL (CO_EVENT_ERRCTRL_T *pFunction*)

coEventRegister_ERRCTRL - register error control event

Returns

RET_T

Parameters

pFunction	pointer to function

5.35.4.3 EXTERN_DECL RET_T coEventRegister_NMT (CO_EVENT_NMT_T *pFunction*)

coEventRegister_NMT - register NMT event

register indication function for NMT events

Returns

RET_T

Parameters

pFunction	pointer to function

5.35.4.4 EXTERN_DECL RET_T coGuardingMasterStart (UNSIGNED8 *node*)

coGuardingMasterStart - start master node guarding

This function starts the master node guarding monitoring for the given node-id and the configured monitoring time from object dictionary.

Please note: The NMT state is set to unknown until next guarding was received

Returns

RET_T

Return values

RET_PARAMETER_INCOMPATIBLE	invalid node id

Parameters

node	node id

5.35.4.5 EXTERN_DECL RET_T coGuardingMasterStop (UNSIGNED8 *node*)

coGuardingMasterStop - stop master node guarding

This function stops the master node guarding monitoring for the given node-id

Returns

RET_T

Return values

RET_PARAMETER_INCOMPATIBLE	invalid node id

Parameters

node	node id

5.35.4.6 EXTERN_DECL RET_T coHbConsumerSet (UNSIGNED8 *node*, UNSIGNED16 *hbTime*)

coHbConsumerSet - setup heartbeat consumer

This function configures a hearbeat consumer for the given node-id and the monitoring time. The data are automatically saved at the object dictionary. If an entry at the object dictionary already exist, then it will be overwritten.
The parameter node have to be valid, otherwise the function returns an error.

Returns

RET_T

Return values

RET_PARAMETER_INCOMPATIBLE	invalid node id

Parameters

node	node id
hbTime	heartbeat monitoring time

5.35.4.7 EXTERN_DECL RET_T coHbConsumerStart (UNSIGNED8 *node*)

coHbConsumerStart - start heartbeat consumer monitoring

This function starts a hearbeat consumer monitoring for the given node-id and the configured monitoring time from object dictionary.

Please note: The NMT state is set to unknown until next HB was received

Returns

RET_T

Return values

RET_PARAMETER_INCOMPATIBLE	invalid node id

Parameters

node	node id

5.35.4.8 EXTERN_DECL UNSIGNED8 coNmtGetNodeId (void)

coNmtGetNodeId - returns actual node id

Returns

node-id

5.35.4.9 EXTERN_DECL CO_NMT_STATE_T coNmtGetRemoteNodeState (UNSIGNED8 *nodeId*)

coNmtGetRemoteNodeState - get remote node state

This function returns the NMT state of a remote node. If heartbeat monitoring of this node is disabled or has been failed, CO_NMT_STATE_UNKNOWN is returned.

Returns

CO_NMT_STATE_T

Parameters

node↩ Id	remote node id

5.35.4.10 EXTERN_DECL CO_NMT_STATE_T coNmtGetState (void)

coGetNmtState - returns current NMT state

This function returns the current NMT state of the local node.

Returns

NMT state

5.35.4.11 EXTERN_DECL BOOL_T coNmtInhibitActive (void)

icoNmtInhibitActive - check if inhibit is active

Returns

inhibit state

5.35.4.12 EXTERN_DECL RET_T coNmtInit (UNSIGNED8 *master*)

coInitNmt - init NMT functionality

This function initializes the NMT functionality and calls an internal reset communication.

If parameter master is unequal 0 the node will be initialized as NMT master. If flying master is enabled, the decision for master/slave or flying master is done by check object 0x1f80. (parameter master is not used!) In this case, the node starts as slave and wait for the master negotiation.

Returns

RET_T

Parameters

master	master mode

5.35.4.13 EXTERN_DECL RET_T coNmtLocalStateReq (CO_NMT_STATE_T *reqState*)

coNmtLocalStateReq - request local NMT state change

Be carfule - NMT commands should be generated only by the master

Returns

RET_T

Parameters

reqState	new requested state

5.35.4.14 EXTERN_DECL BOOL_T coNmtNodeIsMaster (void)

coNmtNodeIsMaster - detect if node is master

Return values

CO_TRUE	- node is master
CO_FALSE	- node is not master

5.35.4.15 EXTERN_DECL RET_T coNmtStateReq (UNSIGNED8 *node*, CO_NMT_STATE_T *reqState*, BOOL_T *master*)

coNmtStateReq - request NMT state change

Request NMT state change for the given node 1..127. If *node* == 0, the NMT request is sent to all nodes. If the NMT sending master should enter the same NMT state as the addressed node the *master* flag has to be set to CO_TRUE. This is true for *node* == 0 too.

If *node* == the masters own nodeId, the requested state is only valid for the own node.

If the inhibit time is set (see object 0x102a, NMT inhibit time), NMT command is not sent if time hasn't been elapsed.

Returns

RET_T

Parameters

node	node
reqState	new requested state
master	valid for master

5.36 co_nmtmaster.c File Reference

NMT master services.

Functions

- RET_T coNmtStateReq (UNSIGNED8 node, CO_NMT_STATE_T reqState, BOOL_T master)

  coNmtStateReq - request NMT state change
- BOOL_T coNmtInhibitActive (void)

  icoNmtInhibitActive - check if inhibit is active
- BOOL_T coNmtNodeIsMaster (void)

  coNmtNodeIsMaster - detect if node is master

5.36.1 Detailed Description

NMT master services.

contains NMT master services

5.36.2 Function Documentation

5.36.2.1 BOOL_T coNmtInhibitActive (void)

icoNmtInhibitActive - check if inhibit is active

Returns

inhibit state

5.36.2.2 BOOL_T coNmtNodeIsMaster (void)

coNmtNodeIsMaster - detect if node is master

Return values

CO_TRUE	- node is master
CO_FALSE	- node is not master

5.36.2.3 RET_T coNmtStateReq (UNSIGNED8 *node*, CO_NMT_STATE_T *reqState*, BOOL_T *master*)

coNmtStateReq - request NMT state change

Request NMT state change for the given node 1..127. If *node* == 0, the NMT request is sent to all nodes. If the NMT sending master should enter the same NMT state as the addressed node the *master* flag has to be set to CO_TRUE. This is true for *node* == 0 too.

If *node* == the masters own nodeId, the requested state is only valid for the own node.

If the inhibit time is set (see object 0x102a, NMT inhibit time), NMT command is not sent if time hasn't been elapsed.

Returns

RET_T

Parameters

node	node
reqState	new requested state
master	valid for master

5.37 co_nmtslave.c File Reference

NMT slave services.

5.37.1 Detailed Description

NMT slave services.

contains NMT slave services for self starting devices

5.38 co_odaccess.c File Reference

object dictionary access

Functions

- RET_T coOdSetCobid (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED32 newCobId)

  coOdSetCobid - set cob id
- void ∗ coOdGetObjAddr (UNSIGNED16 index, UNSIGNED8 subIndex)

  coOdGetObjAddr - get address of an object
- RET_T coOdGetObj_u8 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pObj)

  coOdGetObj_u8 - get UNSIGNED8 object
- RET_T coOdGetObj_u16 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED16 ∗pObj)

  coOdGetObj_u16 - get UNSIGNED16 object
- RET_T coOdGetObj_u32 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED32 ∗pObj)

  coOdGetObj_u32 - get UNSIGNED32 object
- RET_T coOdGetObj_u24 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED24 ∗pObj)

  coOdGetObj_u24 - get UNSIGNED24 object
- RET_T coOdGetObj_u40 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED40 ∗pObj)

  coOdGetObj_u40 - get UNSIGNED40 object
- RET_T coOdGetObj_u48 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED48 ∗pObj)

coOdGetObj_u48 - get UNSIGNED48 object
- RET_T coOdGetObj_u64 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED64 ∗pObj)

  coOdGetObj_u64 - get UNSIGNED64 object
- RET_T coOdGetObj_i8 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER8 ∗pObj)

  coOdGetObj_i8 - get INTEGER8 object
- RET_T coOdGetObj_i16 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER16 ∗pObj)

  coOdGetObj_i16 - get INTEGER16 object
- RET_T coOdGetObj_i32 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER32 ∗pObj)

  coOdGetObj_i32 - get INTEGER32 object
- RET_T coOdGetObj_r32 (UNSIGNED16 index, UNSIGNED8 subIndex, REAL32 ∗pObj)

  coOdGetObj_r32 - get REAL32 object
- RET_T coOdPutObj_u8 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 newVal)

  coOdPutObj_u8 - put UNSIGNED8 value to object
- RET_T coOdPutObj_u16 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED16 newVal)

  coOdPutObj_u16 - put UNSIGNED16 value to object
- RET_T coOdPutObj_u32 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED32 newVal)

  coOdPutObj_u32 - put UNSIGNED32 value to object
- RET_T coOdPutObj_i8 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER8 newVal)

  coOdPutObj_i8 - Put INTEGER8 object
- RET_T coOdPutObj_i16 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER16 newVal)

  coOdPutObj_i16 - Put INTEGER16 object
- RET_T coOdPutObj_i32 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER32 newVal)

  coOdPutObj_i32 - Put INTEGER32 object
- RET_T coOdPutObj_r32 (UNSIGNED16 index, UNSIGNED8 subIndex, REAL32 newVal)

  coOdPutObj_r32 - Put REAL32 object
- RET_T coOdPutObj_u24 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED24 newVal)

  coOdPutObj_u24 - Put UNSIGNED24 Object
- RET_T coOdPutObj_u40 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED40 newVal)

  coOdPutObj_u40 - Put UNSIGNED40 Object
- RET_T coOdPutObj_u48 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED48 newVal)

  coOdPutObj_u48 - Put UNSIGNED48 Object
- RET_T coOdPutObj_u64 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED64 newVal)

  coOdPutObj_u64 - Put UNSIGNED64 Object
- UNSIGNED16 coOdGetObjAttribute (CO_CONST CO_OBJECT_DESC_T ∗pObjDesc)

  coOdGetObjAttribute - get object attribute
- RET_T coOdDomainAddrSet (UNSIGNED16 index, UNSIGNED8 subIndex, CO_DOMAIN_PTR pAddr, U↩
NSIGNED32 size)

  coOdDomainAddrSet - set domain address
- RET_T coOdVisStringSet (UNSIGNED16 index, UNSIGNED8 subIndex, VIS_STRING pAddr, UNSIGNED32 size)

  coOdVisStringSet - set address and len for visible string
- UNSIGNED32 coOdGetObjSize (CO_CONST CO_OBJECT_DESC_T ∗pDesc)

  coOdGetObjSize - get object size
- RET_T icoOdGetObjRecMapData (UNSIGNED16 index, UNSIGNED8 subIndex, void ∗∗pVar, UNSIGNED8 ∗pLen, BOOL_T ∗pNumeric)

  icoOdGetObjRecMapData - get data of receive mapping entry
- RET_T icoOdGetObjTrMapData (UNSIGNED16 index, UNSIGNED8 subIndex, CO_CONST void ∗∗pVar, U↩
NSIGNED8 ∗pLen, BOOL_T ∗pNumeric)

  icoOdGetObjTrMapData - get data of transmit mapping entry
- RET_T icoOdCheckObjAttr (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED16 checkAttr)

  icoOdCheckObjAttr - check object for given attributes

- RET_T coOdGetDefaultVal_u8 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pDefVal)

    coOdGetDefaultVal_u8 - get default value for specific object
- RET_T coOdGetDefaultVal_u16 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED16 ∗pDefVal)

    coOdGetDefaultVal_u16 - get default value for specific object
- RET_T coOdGetDefaultVal_u32 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED32 ∗pDefVal)

    coOdGetDefaultVal_u32 - get default value for specific object
- RET_T coOdGetObjDescPtr (UNSIGNED16 index, UNSIGNED8 subIndex, CO_CONST CO_OBJECT_D←ESC_T ∗∗pDescPtr)

    coOdGetObjDescPtr - get object description pointer
- RET_T coEventRegister_OBJECT_CHANGED (CO_EVENT_OBJECT_CHANGED_FCT_T pFunction, U←NSIGNED16 index, UNSIGNED8 subIndex)

    coEventRegister_OBJECT_CHANGED - register object changed function
- void coOdInitOdPtr (const CO_OD_ASSIGN_T ∗pOdAssing, UNSIGNED16 odCnt, const CO_OBJECT_←DESC_T ∗pObjdesc, UNSIGNED16 descCnt, CO_EVENT_OBJECT_CHANGED_FCT_T ∗pEventPtr, const CO_OD_DATA_VARIABLES_T ∗pOdVarPointers)

    coOdInitOdPtr - init all object dictionary and variable pointers

5.38.1 Detailed Description

object dictionary access

contains routines for object dictionary access

5.38.2 Function Documentation

5.38.2.1 RET_T coEventRegister_OBJECT_CHANGED (CO_EVENT_OBJECT_CHANGED_FCT_T *pFunction,* UNSIGNED16 *index,* UNSIGNED8 *subIndex*)

coEventRegister_OBJECT_CHANGED - register object changed function

This function registered a indication function for a given object. Each time, this object is changed by PDO, SDO or by coOdPutObj_xx() the given function is called.

If the subindex == 255, then the indication is called for each subindex.

Returns

RET_T

Parameters

pFunction	pointer to function
index	index
subIndex	subIndex

5.38.2.2 RET_T coOdDomainAddrSet (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* CO_DOMAIN_PTR *pAddr,* UNSIGNED32 *size*)

coOdDomainAddrSet - set domain address

This function sets the adress and the size of a domain.
At the initialization time, domains are not initialized at the object dictionary. This has to be done by this function.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pAddr	pointer to domain
size	domain length

5.38.2.3 RET_T coOdGetDefaultVal_u16 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED16 ∗ *pDefVal*)

coOdGetDefaultVal_u16 - get default value for specific object

This function returns the default value of an UNSIGNED16 object.

Returns

RET_T

Parameters

index	index
subIndex	sub index
pDefVal	pointer to default val

5.38.2.4 RET_T coOdGetDefaultVal_u32 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED32 ∗ *pDefVal*)

coOdGetDefaultVal_u32 - get default value for specific object

This function returns the default value of an UNSIGNED32 object.

Returns

RET_T

Parameters

index	index
subIndex	sub index
pDefVal	pointer to default val

5.38.2.5 RET_T coOdGetDefaultVal_u8 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED8 ∗ *pDefVal*)

coOdGetDefaultVal_u8 - get default value for specific object

This function returns the default value of an UNSIGNED8 object.

Returns

RET_T

Parameters

index	index
subIndex	sub index
pDefVal	pointer to default val

5.38.2.6 RET_T coOdGetObj_i16 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, INTEGER16 ∗ *pObj*)

coOdGetObj_i16 - get INTEGER16 object

Get an object from the object dictionary from type INTEGER16.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.38.2.7 RET_T coOdGetObj_i32 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, INTEGER32 ∗ *pObj*)

coOdGetObj_i32 - get INTEGER32 object

Get an object from the object dictionary from type INTEGER32.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.38.2.8 RET_T coOdGetObj_i8 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* INTEGER8 ∗ *pObj*)

coOdGetObj_i8 - get INTEGER8 object

Get an object from the object dictionary from type INTEGER8.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.38.2.9 RET_T coOdGetObj_r32 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* REAL32 ∗ *pObj*)

coOdGetObj_r32 - get REAL32 object

Get an object from the object dictionary from type REAL32.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.38.2.10 RET_T coOdGetObj_u16 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED16 ∗ *pObj*)

coOdGetObj_u16 - get UNSIGNED16 object

Get an object from the object dictionary from type UNSIGNED16.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.38.2.11 RET_T coOdGetObj_u24 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED24 ∗ *pObj*)

coOdGetObj_u24 - get UNSIGNED24 object

Get an object from the object dictionary from type UNSIGNED24.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.38.2.12 RET_T coOdGetObj_u32 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED32 ∗ *pObj*)

coOdGetObj_u32 - get UNSIGNED32 object

Get an object from the object dictionary from type UNSIGNED32.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.38.2.13 RET_T coOdGetObj_u40 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED40 ∗ *pObj*)

coOdGetObj_u40 - get UNSIGNED40 object

Get an object from the object dictionary from type UNSIGNED40

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.38.2.14 RET_T coOdGetObj_u48 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED48 ∗ *pObj*)

coOdGetObj_u48 - get UNSIGNED48 object

Get an object from the object dictionary from type UNSIGNED48

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.38.2.15 RET_T coOdGetObj_u64 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED64 ∗ *pObj*)

coOdGetObj_u64 - get UNSIGNED64 object

Get an object from the object dictionary from type UNSIGNED64

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.38.2.16 RET_T coOdGetObj_u8 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pObj*)

coOdGetObj_u8 - get UNSIGNED8 object

Get an object from the object dictionary from type UNSIGNED8.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.38.2.17 void∗ coOdGetObjAddr (UNSIGNED16 *index,* UNSIGNED8 *subIndex*)

coOdGetObjAddr - get address of an object

Get the address of an object from the object dictionary.

Returns

pointer to object address, NULL if object not found

Parameters

index	index of object
subIndex	subindex of object

5.38.2.18 UNSIGNED16 coOdGetObjAttribute (CO_CONST CO_OBJECT_DESC_T ∗ *pObjDesc*)

coOdGetObjAttribute - get object attribute

This function returns the attribute of the object from the given object description.

Returns

attribute

Parameters

pObjDesc	pointer to object description

5.38.2.19 RET_T coOdGetObjDescPtr (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* CO_CONST CO_OBJECT_DESC_T ∗∗ *pDescPtr*)

coOdGetObjDescPtr - get object description pointer

This function returns a pointer to the object description of an object of the object dictionary.

Returns

RET_T

Parameters

index	index
subIndex	sub index
pDescPtr	pointer for description index

5.38.2.20 UNSIGNED32 coOdGetObjSize (CO_CONST CO_OBJECT_DESC_T ∗ *pDesc*)

coOdGetObjSize - get object size

This function returns the size of an object (in bytes), given by the object description.

Returns

object size

Parameters

pDesc	pointer for description index

5.38.2.21 void coOdInitOdPtr (const CO_OD_ASSIGN_T ∗ *pOdAssing,* UNSIGNED16 *odCnt,* const CO_OBJECT_DESC_T ∗ *pObjdesc,* UNSIGNED16 *descCnt,* CO_EVENT_OBJECT_CHANGED_FCT_T ∗ *pEventPtr,* const CO_OD_DATA_VARIABLES_T ∗ *pOdVarPointers*)

coOdInitOdPtr - init all object dictionary and variable pointers

This function initializes the object dictionary with all variables.

Returns

none

Parameters

pOdAssing	pointer to OD assign
odCnt	number of objects
pObjdesc	pointer to obj descr
descCnt	number of obj desc
pEventPtr	pointer to obj events
pOdVarPointers	pointer to variable types

5.38.2.22 RET_T coOdPutObj_i16 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* INTEGER16 *newVal*)

coOdPutObj_i16 - Put INTEGER16 object

Put value from type INTEGER16 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.38.2.23 RET_T coOdPutObj_i32 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, INTEGER32 *newVal*)

coOdPutObj_i32 - Put INTEGER32 object

Put value from type INTEGER32 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.38.2.24 RET_T coOdPutObj_i8 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, INTEGER8 *newVal*)

coOdPutObj_i8 - Put INTEGER8 object

Put value from type INTEGER8 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.38.2.25 RET_T coOdPutObj_r32 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, REAL32 *newVal*)

coOdPutObj_r32 - Put REAL32 object

Put value from type REAL32 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.38.2.26 RET_T coOdPutObj_u16 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED16 *newVal*)

coOdPutObj_u16 - put UNSIGNED16 value to object

Put value from type UNSIGNED16 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.38.2.27 RET_T coOdPutObj_u24 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED24 *newVal*)

coOdPutObj_u24 - Put UNSIGNED24 Object

Put value from type UNSIGNED24 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.38.2.28 RET_T coOdPutObj_u32 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED32 *newVal*)

coOdPutObj_u32 - put UNSIGNED32 value to object

Put value from type UNSIGNED32 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.38.2.29 RET_T coOdPutObj_u40 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED40 *newVal*)

coOdPutObj_u40 - Put UNSIGNED40 Object

Put value from type UNSIGNED24 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.38.2.30 RET_T coOdPutObj_u48 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED48 *newVal*)

coOdPutObj_u48 - Put UNSIGNED48 Object

Put value from type UNSIGNED24 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.38.2.31 RET_T coOdPutObj_u64 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED64 *newVal*)

coOdPutObj_u64 - Put UNSIGNED64 Object

Put value from type UNSIGNED24 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.38.2.32 RET_T coOdPutObj_u8 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 *newVal*)

coOdPutObj_u8 - put UNSIGNED8 value to object

Put value from type UNSIGNED8 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.38.2.33 RET_T coOdSetCobid (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED32 *newCobId*)

coOdSetCobid - set cob id

This function set the COB-Id for a service indicated by index and subindex.

According to the standard, the COB-ID is disabled first by this function, and then the new COB-ID is set.

Returns

RET_T

Parameters

index	index for the cob
subIndex	subIndex for the cob
new↩ CobId	new cob-id

5.38.2.34 RET_T coOdVisStringSet (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* VIS_STRING *pAddr,* UNSIGNED32 *size*)

coOdVisStringSet - set address and len for visible string

This function change the address and length if a visible string object. It can only be used for non-constant strings, defined as user variable.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pAddr	pointer to string
size	string length

5.38.2.35 RET_T icoOdCheckObjAttr (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED16 *checkAttr*)

icoOdCheckObjAttr - check object for given attributes

internal

This function checks an object for the given attributes

Returns

RET_T

5.38.2.36 RET_T icoOdGetObjRecMapData (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* void ∗∗ *pVar,* UNSIGNED8 ∗ *pLen,* BOOL_T ∗ *pNumeric*)

icoOdGetObjRecMapData - get data of receive mapping entry

internal

This function returns data of mapping entry If index is not mapable, returns error

Returns

RET_T

5.38.2.37 RET_T icoOdGetObjTrMapData (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* CO_CONST void ∗∗ *pVar,* UNSIGNED8 ∗ *pLen,* BOOL_T ∗ *pNumeric*)

icoOdGetObjTrMapData - get data of transmit mapping entry

internal

This function returns data of mapping entry If index is not mapable, returns error

Returns

RET_T

5.39 co_odaccess.h File Reference

defines for OD access

Macros

- #define CO_OS_LOCK_OD()
- #define CO_OS_UNLOCK_OD()
- #define CO_ATTR_READ 0x0001u
- #define CO_ATTR_WRITE 0x0002u
- #define CO_ATTR_NUM 0x0004u
- #define CO_ATTR_MAP 0x0008u
- #define CO_ATTR_MAP_TR 0x0010u
- #define CO_ATTR_MAP_REC 0x0020u
- #define CO_ATTR_DEFVAL 0x0040u
- #define CO_ATTR_LIMIT 0x0080u
- #define CO_ATTR_DYNOD 0x0100u
- #define CO_ATTR_STORE 0x0200u
- #define CO_ATTR_COMPACT 0x0400u

Typedefs

- typedef RET_T(∗ CO_EVENT_OBJECT_CHANGED_FCT_T) (UNSIGNED16, UNSIGNED8)

  function pointer to object changed function

Enumerations

Functions

- void coOdInitOdPtr (const CO_OD_ASSIGN_T ∗pOdAssing, UNSIGNED16 odCnt, const CO_OBJECT_↩ DESC_T ∗pObjdesc, UNSIGNED16 descCnt, CO_EVENT_OBJECT_CHANGED_FCT_T ∗pEventPtr, const CO_OD_DATA_VARIABLES_T ∗pOdVarPointers)

  coOdInitOdPtr - init all object dictionary and variable pointers
- EXTERN_DECL RET_T coOdGetObj_u32 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED32 ∗pObj)

  coOdGetObj_u32 - get UNSIGNED32 object
- EXTERN_DECL RET_T coOdGetObj_u16 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED16 ∗pObj)

  coOdGetObj_u16 - get UNSIGNED16 object
- EXTERN_DECL RET_T coOdGetObj_u8 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pObj)

  coOdGetObj_u8 - get UNSIGNED8 object
- EXTERN_DECL RET_T coOdGetObj_i32 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER32 ∗pObj)

  coOdGetObj_i32 - get INTEGER32 object
- EXTERN_DECL RET_T coOdGetObj_i16 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER16 ∗pObj)

  coOdGetObj_i16 - get INTEGER16 object
- EXTERN_DECL RET_T coOdGetObj_i8 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER8 ∗pObj)

  coOdGetObj_i8 - get INTEGER8 object
- EXTERN_DECL RET_T coOdGetObj_r32 (UNSIGNED16 index, UNSIGNED8 subIndex, REAL32 ∗pObj)

  coOdGetObj_r32 - get REAL32 object

- EXTERN_DECL RET_T coOdGetObj_u24 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED24 ∗pObj)

  coOdGetObj_u24 - get UNSIGNED24 object
- EXTERN_DECL RET_T coOdGetObj_u40 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED40 ∗pObj)

  coOdGetObj_u40 - get UNSIGNED40 object
- EXTERN_DECL RET_T coOdGetObj_u48 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED48 ∗pObj)

  coOdGetObj_u48 - get UNSIGNED48 object
- EXTERN_DECL RET_T coOdGetObj_u64 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED64 ∗pObj)

  coOdGetObj_u64 - get UNSIGNED64 object
- EXTERN_DECL UNSIGNED16 coOdGetObjAttribute (CO_CONST CO_OBJECT_DESC_T ∗pObjDesc)

  coOdGetObjAttribute - get object attribute
- EXTERN_DECL RET_T coOdGetObjDescPtr (UNSIGNED16 index, UNSIGNED8 subIndex, CO_CONST CO_OBJECT_DESC_T ∗∗pDescPtr)

  coOdGetObjDescPtr - get object description pointer
- EXTERN_DECL UNSIGNED32 coOdGetObjSize (CO_CONST CO_OBJECT_DESC_T ∗pDesc)

  coOdGetObjSize - get object size
- EXTERN_DECL RET_T coOdPutObj_u32 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED32 newVal)

  coOdPutObj_u32 - put UNSIGNED32 value to object
- EXTERN_DECL RET_T coOdPutObj_u16 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED16 newVal)

  coOdPutObj_u16 - put UNSIGNED16 value to object
- EXTERN_DECL RET_T coOdPutObj_u8 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 new←Val)

  coOdPutObj_u8 - put UNSIGNED8 value to object
- EXTERN_DECL RET_T coOdPutObj_i32 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER32 new←Val)

  coOdPutObj_i32 - Put INTEGER32 object
- EXTERN_DECL RET_T coOdPutObj_i16 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER16 new←Val)

  coOdPutObj_i16 - Put INTEGER16 object
- EXTERN_DECL RET_T coOdPutObj_i8 (UNSIGNED16 index, UNSIGNED8 subIndex, INTEGER8 newVal)

  coOdPutObj_i8 - Put INTEGER8 object
- EXTERN_DECL RET_T coOdPutObj_r32 (UNSIGNED16 index, UNSIGNED8 subIndex, REAL32 newVal)

  coOdPutObj_r32 - Put REAL32 object
- EXTERN_DECL RET_T coOdPutObj_u24 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED24 newVal)

  coOdPutObj_u24 - Put UNSIGNED24 Object
- EXTERN_DECL RET_T coOdPutObj_u40 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED40 newVal)

  coOdPutObj_u40 - Put UNSIGNED40 Object
- EXTERN_DECL RET_T coOdPutObj_u48 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED48 newVal)

  coOdPutObj_u48 - Put UNSIGNED48 Object
- EXTERN_DECL RET_T coOdPutObj_u64 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED64 newVal)

  coOdPutObj_u64 - Put UNSIGNED64 Object
- EXTERN_DECL void ∗ coOdGetObjAddr (UNSIGNED16 index, UNSIGNED8 subIndex)

  coOdGetObjAddr - get address of an object

- EXTERN_DECL RET_T coOdGetDefaultVal_u32 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGN↩ ED32 ∗pDefVal)

    coOdGetDefaultVal_u32 - get default value for specific object
- EXTERN_DECL RET_T coOdGetDefaultVal_u16 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGN↩ ED16 ∗pDefVal)

    coOdGetDefaultVal_u16 - get default value for specific object
- EXTERN_DECL RET_T coOdGetDefaultVal_u8 (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pDefVal)

    coOdGetDefaultVal_u8 - get default value for specific object
- EXTERN_DECL RET_T coOdSetCobid (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED32 new↩ CobId)

    coOdSetCobid - set cob id
- EXTERN_DECL RET_T coOdDomainAddrSet (UNSIGNED16 index, UNSIGNED8 subIndex, CO_DOMAI↩ N_PTR pAddr, UNSIGNED32 size)

    coOdDomainAddrSet - set domain address
- EXTERN_DECL RET_T coOdVisStringSet (UNSIGNED16 index, UNSIGNED8 subIndex, VIS_STRING p↩ Addr, UNSIGNED32 size)

    coOdVisStringSet - set address and len for visible string
- EXTERN_DECL RET_T coEventRegister_OBJECT_CHANGED (CO_EVENT_OBJECT_CHANGED_FCT↩ _T, UNSIGNED16, UNSIGNED8)

    coEventRegister_OBJECT_CHANGED - register object changed function
- EXTERN_DECL RET_T coDynOdInit (UNSIGNED16 objCnt, UNSIGNED16 u8Cnt, UNSIGNED16 u16Cnt, UNSIGNED16 u32Cnt, UNSIGNED16 i8Cnt, UNSIGNED16 i16Cnt, UNSIGNED16 i32Cnt, UNSIGNED16 u64Cnt)

    coDynOdInit - init dynamic object dictionary
- EXTERN_DECL RET_T coDynOdRelease (void)

    coDynOdRelease - release dynamic object dictionary
- EXTERN_DECL RET_T coDynOdAddIndex (UNSIGNED16 index, UNSIGNED8 nrOfSubs, CO_ODTYPE_T odType)

    coDynOdAddIndex - add a new object index
- EXTERN_DECL RET_T coDynOdAddSubIndex (UNSIGNED16 index, UNSIGNED8 subIndex, CO_DATA↩ _TYPE_T dataType, UNSIGNED16 attr, void ∗pVar)

    coDynOdAddSubIndex - add new subindex
- EXTERN_DECL RET_T coDynOdSetSubIndexAddr (UNSIGNED16 index, UNSIGNED8 subIndex, CO_D↩ ATA_TYPE_T dataType, void ∗pVar)

    coDynOdSetSubIndexAddr - set new pointer for subindex

5.39.1 Detailed Description

defines for OD access

- contains defines for object dictionary access

5.39.2 Macro Definition Documentation

5.39.2.1 #define CO_ATTR_COMPACT 0x0400u

object is compact array

5.39.2.2 #define CO_ATTR_DEFVAL 0x0040u

object has a default value

5.39.2.3 #define CO_ATTR_DYNOD 0x0100u

object is a dynamic created object

5.39.2.4 #define CO_ATTR_LIMIT 0x0080u

object has limits

5.39.2.5 #define CO_ATTR_MAP 0x0008u

object can be mapped into a PDO

5.39.2.6 #define CO_ATTR_MAP_REC 0x0020u

object can be mapped into a receive PDO

5.39.2.7 #define CO_ATTR_MAP_TR 0x0010u

object can be mapped into a transmit PDO

5.39.2.8 #define CO_ATTR_NUM 0x0004u

object is numeric

5.39.2.9 #define CO_ATTR_READ 0x0001u

object attributes object readable

5.39.2.10 #define CO_ATTR_STORE 0x0200u

object is supposed to be stored

5.39.2.11 #define CO_ATTR_WRITE 0x0002u

object writeable

5.39.2.12 #define CO_OS_LOCK_OD( )

function call to lock object dictionary

5.39.2.13 #define CO_OS_UNLOCK_OD( )

function call to unlock object dictionary

5.39.3 Typedef Documentation

5.39.3.1 typedef RET_T(∗ CO_EVENT_OBJECT_CHANGED_FCT_T) (UNSIGNED16, UNSIGNED8)

function pointer to object changed function

Parameters

index	- object index
subindex	- object subindex

Returns

RET_T

5.39.4 Enumeration Type Documentation

5.39.4.1 enum CO_DATA_TYPE_T

object datatypes

5.39.4.2 enum CO_ODTYPE_T

Object type

Enumerator

CO_ODTYPE_VAR	variable
CO_ODTYPE_ARRAY	array
CO_ODTYPE_STRUCT	structure

5.39.5 Function Documentation

5.39.5.1 EXTERN_DECL RET_T coDynOdAddIndex (UNSIGNED16 *index,* UNSIGNED8 *nrOfSubs,* CO_ODTYPE_T *odType*)

coDynOdAddIndex - add a new object index

Return values

RET_IDX_NOT_FOUND	index < 0x2000 are not allowed
RET_INVALID_PARAMETER	index already exist
RET_EVENT_NO_RESSOURCE	no resource available

Parameters

index	index
nrOfSubs	number of subindex
odType	variable, array, struct

5.39.5.2 EXTERN_DECL RET_T coDynOdAddSubIndex (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, CO_DATA_TYPE_T *dataType*, UNSIGNED16 *attr*, void * *pVar*)

coDynOdAddSubIndex - add new subindex

no check for to many data or duplicate subindex

Return values

RET_DATA_TYPE_MISMATCH	data type not supported (only U8, U16, U32, I8, I16, I32 allowed)
RET_IDX_NOT_FOUND	index not found

Parameters

index	index
subIndex	number of subindex
dataType	data type
attr	attribute
pVar	pointer to variable

5.39.5.3 EXTERN_DECL RET_T coDynOdInit (UNSIGNED16 *objCnt*, UNSIGNED16 *u8Cnt*, UNSIGNED16 *u16Cnt*, UNSIGNED16 *u32Cnt*, UNSIGNED16 *i8Cnt*, UNSIGNED16 *i16Cnt*, UNSIGNED16 *i32Cnt*, UNSIGNED16 *u64Cnt*)

coDynOdInit - init dynamic object dictionary

Return values

RET_OK	initialisation OK
RET_EVENT_NO_RESSOURCE	error at malloc()

Parameters

objCnt	number of new objects for can line
u8Cnt	number of U8 vars for can line
u16Cnt	number of U16 vars for can line

Parameters

u32Cnt	number of U32 vars for can line
i8Cnt	number of i8 vars for can line
i16Cnt	number of i16 vars for can line
i32Cnt	number of i32 vars for can line
u64Cnt	number of U64 vars for can line

5.39.5.4 EXTERN_DECL RET_T coDynOdRelease (void)

coDynOdRelease - release dynamic object dictionary

Deinit dynamic object dictionary and release all requested memory

Return values

RET_OK	deinitialisation OK

5.39.5.5 EXTERN_DECL RET_T coDynOdSetSubIndexAddr (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* CO_DATA_TYPE_T *dataType,* void ∗ *pVar*)

coDynOdSetSubIndexAddr - set new pointer for subindex

set a new data pointer for a given sub index

Return values

RET_DATA_TYPE_MISMATCH	data type not supported (only U8, U16, U32, I8, I16, I32 allowed)
RET_IDX_NOT_FOUND	index not found

Parameters

index	index
subIndex	number of subindex
dataType	data type
pVar	pointer to variable

5.39.5.6 EXTERN_DECL RET_T coEventRegister_OBJECT_CHANGED (CO_EVENT_OBJECT_CHANGED_FCT_T *pFunction,* UNSIGNED16 *index,* UNSIGNED8 *subIndex*)

coEventRegister_OBJECT_CHANGED - register object changed function

This function registered a indication function for a given object. Each time, this object is changed by PDO, SDO or by coOdPutObj_xx() the given function is called.

If the subindex == 255, then the indication is called for each subindex.

Returns

RET_T

Parameters

pFunction	pointer to function
index	index
subIndex	subIndex

5.39.5.7 EXTERN_DECL RET_T coOdDomainAddrSet (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, CO_DOMAIN_PTR *pAddr*, UNSIGNED32 *size*)

coOdDomainAddrSet - set domain address

This function sets the adress and the size of a domain.
At the initialization time, domains are not initialized at the object dictionary. This has to be done by this function.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pAddr	pointer to domain
size	domain length

5.39.5.8 EXTERN_DECL RET_T coOdGetDefaultVal_u16 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED16 ∗ *pDefVal*)

coOdGetDefaultVal_u16 - get default value for specific object

This function returns the default value of an UNSIGNED16 object.

Returns

RET_T

Parameters

index	index
subIndex	sub index
pDefVal	pointer to default val

5.39.5.9 EXTERN_DECL RET_T coOdGetDefaultVal_u32 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED32 ∗ *pDefVal*)

coOdGetDefaultVal_u32 - get default value for specific object

This function returns the default value of an UNSIGNED32 object.

Returns

RET_T

Parameters

index	index
subIndex	sub index
pDefVal	pointer to default val

5.39.5.10 EXTERN_DECL RET_T coOdGetDefaultVal_u8 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pDefVal*)

coOdGetDefaultVal_u8 - get default value for specific object

This function returns the default value of an UNSIGNED8 object.

Returns

RET_T

Parameters

index	index
subIndex	sub index
pDefVal	pointer to default val

5.39.5.11 EXTERN_DECL RET_T coOdGetObj_i16 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* INTEGER16 ∗ *pObj*)

coOdGetObj_i16 - get INTEGER16 object

Get an object from the object dictionary from type INTEGER16.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.39.5.12 EXTERN_DECL RET_T coOdGetObj_i32 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, INTEGER32 ∗ *pObj*)

coOdGetObj_i32 - get INTEGER32 object

Get an object from the object dictionary from type INTEGER32.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.39.5.13 EXTERN_DECL RET_T coOdGetObj_i8 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, INTEGER8 ∗ *pObj*)

coOdGetObj_i8 - get INTEGER8 object

Get an object from the object dictionary from type INTEGER8.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.39.5.14 EXTERN_DECL RET_T coOdGetObj_r32 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, REAL32 ∗ *pObj*)

coOdGetObj_r32 - get REAL32 object

Get an object from the object dictionary from type REAL32.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.39.5.15 EXTERN_DECL RET_T coOdGetObj_u16 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED16 ∗ *pObj*)

coOdGetObj_u16 - get UNSIGNED16 object

Get an object from the object dictionary from type UNSIGNED16.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.39.5.16 EXTERN_DECL RET_T coOdGetObj_u24 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED24 ∗ *pObj*)

coOdGetObj_u24 - get UNSIGNED24 object

Get an object from the object dictionary from type UNSIGNED24.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.39.5.17 EXTERN_DECL RET_T coOdGetObj_u32 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED32 ∗ *pObj*)

coOdGetObj_u32 - get UNSIGNED32 object

Get an object from the object dictionary from type UNSIGNED32.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.39.5.18 EXTERN_DECL RET_T coOdGetObj_u40 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED40 ∗ *pObj*)

coOdGetObj_u40 - get UNSIGNED40 object

Get an object from the object dictionary from type UNSIGNED40

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.39.5.19 EXTERN_DECL RET_T coOdGetObj_u48 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED48 ∗ *pObj*)

coOdGetObj_u48 - get UNSIGNED48 object

Get an object from the object dictionary from type UNSIGNED48

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.39.5.20 EXTERN_DECL RET_T coOdGetObj_u64 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED64 ∗ *pObj*)

coOdGetObj_u64 - get UNSIGNED64 object

Get an object from the object dictionary from type UNSIGNED64

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.39.5.21 EXTERN_DECL RET_T coOdGetObj_u8 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pObj*)

coOdGetObj_u8 - get UNSIGNED8 object

Get an object from the object dictionary from type UNSIGNED8.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pObj	pointer to object

5.39.5.22 EXTERN_DECL void∗ coOdGetObjAddr (UNSIGNED16 *index,* UNSIGNED8 *subIndex*)

coOdGetObjAddr - get address of an object

Get the address of an object from the object dictionary.

Returns

pointer to object address, NULL if object not found

Parameters

index	index of object
subIndex	subindex of object

5.39.5.23 EXTERN_DECL UNSIGNED16 coOdGetObjAttribute (CO_CONST CO_OBJECT_DESC_T ∗ *pObjDesc*)

coOdGetObjAttribute - get object attribute

This function returns the attribute of the object from the given object description.

Returns

attribute

Parameters

pObjDesc	pointer to object description

5.39.5.24 EXTERN_DECL RET_T coOdGetObjDescPtr (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, CO_CONST CO_OBJECT_DESC_T ∗∗ *pDescPtr*)

coOdGetObjDescPtr - get object description pointer

This function returns a pointer to the object description of an object of the object dictionary.

Returns

RET_T

Parameters

index	index
subIndex	sub index
pDescPtr	pointer for description index

5.39.5.25 EXTERN_DECL UNSIGNED32 coOdGetObjSize (CO_CONST CO_OBJECT_DESC_T ∗ *pDesc*)

coOdGetObjSize - get object size

This function returns the size of an object (in bytes), given by the object description.

Returns

object size

Parameters

pDesc	pointer for description index

5.39.5.26 void coOdInitOdPtr (const CO_OD_ASSIGN_T ∗ *pOdAssing*, UNSIGNED16 *odCnt*, const CO_OBJECT_DESC_T ∗ *pObjdesc*, UNSIGNED16 *descCnt*, CO_EVENT_OBJECT_CHANGED_FCT_T ∗ *pEventPtr*, const CO_OD_DATA_VARIABLES_T ∗ *pOdVarPointers*)

coOdInitOdPtr - init all object dictionary and variable pointers

This function initializes the object dictionary with all variables.

Returns

none

Parameters

pOdAssing	pointer to OD assign
odCnt	number of objects
pObjdesc	pointer to obj descr
descCnt	number of obj desc
pEventPtr	pointer to obj events
pOdVarPointers	pointer to variable types

5.39.5.27 EXTERN_DECL RET_T coOdPutObj_i16 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, INTEGER16 *newVal*)

coOdPutObj_i16 - Put INTEGER16 object

Put value from type INTEGER16 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.39.5.28 EXTERN_DECL RET_T coOdPutObj_i32 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, INTEGER32 *newVal*)

coOdPutObj_i32 - Put INTEGER32 object

Put value from type INTEGER32 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.39.5.29 EXTERN_DECL RET_T coOdPutObj_i8 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, INTEGER8 *newVal*)

coOdPutObj_i8 - Put INTEGER8 object

Put value from type INTEGER8 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.39.5.30 EXTERN_DECL RET_T coOdPutObj_r32 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* REAL32 *newVal*)

coOdPutObj_r32 - Put REAL32 object

Put value from type REAL32 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.39.5.31 EXTERN_DECL RET_T coOdPutObj_u16 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED16 *newVal*)

coOdPutObj_u16 - put UNSIGNED16 value to object

Put value from type UNSIGNED16 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.39.5.32 EXTERN_DECL RET_T coOdPutObj_u24 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED24 *newVal*)

coOdPutObj_u24 - Put UNSIGNED24 Object

Put value from type UNSIGNED24 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.39.5.33 EXTERN_DECL RET_T coOdPutObj_u32 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED32 *newVal*)

coOdPutObj_u32 - put UNSIGNED32 value to object

Put value from type UNSIGNED32 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.39.5.34 EXTERN_DECL RET_T coOdPutObj_u40 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED40 *newVal*)

coOdPutObj_u40 - Put UNSIGNED40 Object

Put value from type UNSIGNED24 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.39.5.35 EXTERN_DECL RET_T coOdPutObj_u48 (UNSIGNED16 *index*, UNSIGNED8 *subIndex*, UNSIGNED48 *newVal*)

coOdPutObj_u48 - Put UNSIGNED48 Object

Put value from type UNSIGNED24 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.39.5.36 EXTERN_DECL RET_T coOdPutObj_u64 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED64 *newVal*)

coOdPutObj_u64 - Put UNSIGNED64 Object

Put value from type UNSIGNED24 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.39.5.37 EXTERN_DECL RET_T coOdPutObj_u8 (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 *newVal*)

coOdPutObj_u8 - put UNSIGNED8 value to object

Put value from type UNSIGNED8 to the object dictionary

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
newVal	new value

5.39.5.38 EXTERN_DECL RET_T coOdSetCobid (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED32 *newCobId*)

coOdSetCobid - set cob id

This function set the COB-Id for a service indicated by index and subindex.

According to the standard, the COB-ID is disabled first by this function, and then the new COB-ID is set.

Returns

RET_T

Parameters

index	index for the cob
subIndex	subIndex for the cob
new↵ CobId	new cob-id

5.39.5.39 EXTERN_DECL RET_T coOdVisStringSet (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* VIS_STRING *pAddr,* UNSIGNED32 *size*)

coOdVisStringSet - set address and len for visible string

This function change the address and length if a visible string object. It can only be used for non-constant strings, defined as user variable.

Returns

RET_T

Parameters

index	index of object
subIndex	subindex of object
pAddr	pointer to string
size	string length

5.40 co_odindex.h File Reference

defines for OD index

5.40.1 Detailed Description

defines for OD index

- contains defines for OD index

5.41 co_pdo.c File Reference

PDO transmission and reception routines.

Functions

- RET_T coPdoReqNr (UNSIGNED16 pdoNr, UNSIGNED8 flags)

  coPdoReqNr - request PDO transmission by PDO number
- RET_T coPdoReqObj (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 flags)

  coPdoReqObj - request PDO transmission by object
- BOOL_T coPdoObjIsMapped (UNSIGNED16 pdoNr, UNSIGNED16 index, UNSIGNED8 subIndex)

  coPdoObjIsMapped - check, if object mapped to given PDO
- RET_T coEventRegister_PDO (CO_EVENT_PDO_T pFunction)

  coEventRegister_PDO - register asynchronous PDO event
- RET_T coEventRegister_PDO_REC_EVENT (CO_EVENT_PDO_T pFunction)

coEventRegister_PDO_REC_EVENT - register receive PDO event
- RET_T coEventRegister_PDO_SYNC (CO_EVENT_PDO_T pFunction)

  coEventRegister_PDO_SYNC - register PDO SYNC event
- RET_T coEventRegister_PDO_UPDATE (CO_EVENT_PDO_UPDATE_T pFunction)

  coEventRegister_PDO_UPDATE - register PDO Data Update event
- void icoPdoVarInit (UNSIGNED16 ∗pTrCnt, UNSIGNED16 ∗pRecCnt)

  icoPdoVarInit - init pdo variables
- RET_T coPdoTransmitInit (UNSIGNED16 pdoNr, UNSIGNED8 transType, UNSIGNED16 inhibit, UNSIGN←ED16 eventTime, UNSIGNED8 syncStartVal, CO_CONST PDO_TR_MAP_TABLE_T ∗mapTable)

  coPdoTransmitInit - init transmit pdo functionality
- RET_T coPdoReceiveInit (UNSIGNED16 pdoNr, UNSIGNED8 transType, UNSIGNED16 inhibit, UNSIGN←ED16 eventTime, CO_CONST PDO_REC_MAP_TABLE_T ∗mapTable)

  coPdoReceiveInit - init receive pdo functionality

5.41.1 Detailed Description

PDO transmission and reception routines.

contains PDO handling routines.

5.41.2 Function Documentation

5.41.2.1 RET_T coEventRegister_PDO (CO_EVENT_PDO_T *pFunction*)

coEventRegister_PDO - register asynchronous PDO event

Register an indication function for asynchrounous PDOs.

After a PDO has been received, the data are stored in the object dictionary, and then the given indication function is called. This function is only valid for asynchronous PDOs.

Returns

RET_T

Parameters

pFunction	pointer to function

5.41.2.2 RET_T coEventRegister_PDO_REC_EVENT (CO_EVENT_PDO_T *pFunction*)

coEventRegister_PDO_REC_EVENT - register receive PDO event

Register an indication function for receive PDO event.

For monitoring of receive PDOs the event timer can be used. If the event timer value is unequal 0 then after the reception of a PDO the monitoring is started automatically. if no further PDO in the given time was received, the indication function given to this function is called.

Returns

RET_T

Parameters

pFunction	pointer to function

5.41.2.3 RET_T coEventRegister_PDO_SYNC (CO_EVENT_PDO_T *pFunction*)

coEventRegister_PDO_SYNC - register PDO SYNC event

Register an indication function for received synchronous PDOs.

After the SYNC was received, the received data are stored in the object dictionary, and then this given indication function is called.

Returns

RET_T

Parameters

pFunction	pointer to function

5.41.2.4 RET_T coEventRegister_PDO_UPDATE (CO_EVENT_PDO_UPDATE_T *pFunction*)

coEventRegister_PDO_UPDATE - register PDO Data Update event

Register an indication function for PDO Data Update.

Before a PDO is transmitted, this function is called. Application can now update the given objects at the OD before the data are transmitted with a PDO.

Returns

RET_T

Parameters

pFunction	pointer to function

5.41.2.5 BOOL_T coPdoObjIsMapped (UNSIGNED16 *pdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex*)

coPdoObjIsMapped - check, if object mapped to given PDO

This function checks the PDO, if the given object is actual mapped to this PDO. (Only valid for receive PDOs)

Returns

BOOL_T

Return values

CO_TRUE	object is mapped
CO_FALSE	object is not mapped

Parameters

pdoNr	PDO number
index	index of mapped object
subIndex	subindex of mapped object

5.41.2.6 RET_T coPdoReceiveInit (UNSIGNED16 *pdoNr,* UNSIGNED8 *transType,* UNSIGNED16 *inhibit,* UNSIGNED16 *eventTime,* CO_CONST PDO_REC_MAP_TABLE_T ∗ *mapTable*)

coPdoReceiveInit - init receive pdo functionality

This function initializes a receive PDO. The COB-ID is set at reset communication or at load parameter.

Note: All parameter are reset by their default values at reset communication.

Returns

RET_T

Parameters

pdoNr	PDO number
transType	transmission type
inhibit	inhibit time 100 usec
eventTime	event timer in msec
mapTable	pointer to mapping table

5.41.2.7 RET_T coPdoReqNr (UNSIGNED16 *pdoNr,* UNSIGNED8 *flags*)

coPdoReqNr - request PDO transmission by PDO number

This function requests the transmission of an PDO given by its number.
All mapped objects are automatically copied into the CAN message. If the inhibit time is not active, then the message is transmitted immediately.

If the inhibit time is not ellapsed yet, the transmission depends on the parameter flags:

0 - PDO will be transmitted after inhibit is ellapsed (if data are not changed, PDO will not be transmitted more than once!) MSG_OVERWRITE - if the last PDO is not transmitted yet, overwrite the last data with the new data MSG_RET_INHIBIT - return the function with RET_INHIBIT_ACTIVE, if the inhibit is not ellapsed yet

with the same or

Returns

RET_T

Return values

RET_INVALID_NMT_STATE	invalid NMT state
RET_INVALID_PARAMETER	unknown PDO number
RET_COB_DISABLED	PDO is disabled
RET_INHIBIT_ACTIVE	inhibit time is not yet ellapsed
RET_OK	all function are ok, but have not to be transmitted yet

Parameters

pdoNr	PDO number
flags	transmit flags

5.41.2.8 RET_T coPdoReqObj (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 *flags*)

coPdoReqObj - request PDO transmission by object

This function requests the transmission of the PDO, which the given object is mapped into.
All mapped objects are automatically copied into the CAN message. If the inhibit time is not active, then the message is transmitted immediately.

If the inhibit time is not ellapsed yet, the transmission depends on the parameter flags:

0 - PDO will be transmitted after inhibit is ellapsed MSG_OVERWRITE - if the last PDO is not transmitted yet, overwrite the last data with the new data MSG_RET_INHIBIT - return the function with RET_INHIBIT_ACTIVE, if the inhibit is not ellapsed yet

Returns

RET_T

Return values

RET_INVALID_NMT_STATE	invalid NMT state
RET_INVALID_PARAMETER	unknown PDO number
RET_COB_DISABLED	PDO is disabled
RET_INHIBIT_ACTIVE	inhibit time is not yet ellapsed
RET_OK	all function are ok, but have not to be transmitted yet

Parameters

index	index of mapped object
subIndex	subindex of mapped object
flags	transmit flags

5.41.2.9 RET_T coPdoTransmitInit (UNSIGNED16 *pdoNr,* UNSIGNED8 *transType,* UNSIGNED16 *inhibit,* UNSIGNED16 *eventTime,* UNSIGNED8 *syncStartVal,* CO_CONST PDO_TR_MAP_TABLE_T ∗ *mapTable*)

coPdoTransmitInit - init transmit pdo functionality

This function initializes a transmit PDO. The COB-ID is set at reset communication or at load parameter.

Note: All parameters are reset to their default values at reset communication.

Returns

RET_T

Parameters

pdoNr	PDO number
transType	transmission type
inhibit	inhibit time 100 usec
eventTime	event timer in msec
syncStartVal	sync start value
mapTable	pointer to mapping table

5.42 co_pdo.h File Reference

defines for pdo service

Data Structures

- struct PDO_TR_MAP_ENTRY_T
- struct PDO_REC_MAP_ENTRY_T
- struct PDO_TR_MAP_TABLE_T
- struct PDO_REC_MAP_TABLE_T

Typedefs

- typedef void(∗ CO_EVENT_PDO_T) (UNSIGNED16)

  function pointer to PDO indication
- typedef void(∗ CO_EVENT_PDO_UPDATE_T) (UNSIGNED16, UNSIGNED8)

  function pointer to PDO update indication
- typedef void(∗ CO_EVENT_MPDO_T) (UNSIGNED16, UNSIGNED16, UNSIGNED8)

  function pointer to MPDO indication

Functions

- EXTERN_DECL RET_T coPdoTransmitInit (UNSIGNED16 pdoNr, UNSIGNED8 transType, UNSIGNED16 inhibit, UNSIGNED16 eventTime, UNSIGNED8 syncStartVal, CO_CONST PDO_TR_MAP_TABLE_T ∗map↩ Table)

    coPdoTransmitInit - init transmit pdo functionality
- EXTERN_DECL RET_T coPdoReceiveInit (UNSIGNED16 pdoNr, UNSIGNED8 transType, UNSIGNED16 inhibit, UNSIGNED16 eventTime, CO_CONST PDO_REC_MAP_TABLE_T ∗mapTable)

    coPdoReceiveInit - init receive pdo functionality
- EXTERN_DECL RET_T coPdoReqNr (UNSIGNED16 pdoNr, UNSIGNED8 flags)

    coPdoReqNr - request PDO transmission by PDO number
- EXTERN_DECL RET_T coPdoReqObj (UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 flags)

    coPdoReqObj - request PDO transmission by object
- EXTERN_DECL BOOL_T coPdoObjIsMapped (UNSIGNED16 pdoNr, UNSIGNED16 index, UNSIGNED8 subIndex)

    coPdoObjIsMapped - check, if object mapped to given PDO
- EXTERN_DECL RET_T coEventRegister_PDO (CO_EVENT_PDO_T pFunction)

    coEventRegister_PDO - register asynchronous PDO event
- EXTERN_DECL RET_T coEventRegister_PDO_SYNC (CO_EVENT_PDO_T pFunction)

    coEventRegister_PDO_SYNC - register PDO SYNC event
- EXTERN_DECL RET_T coEventRegister_PDO_REC_EVENT (CO_EVENT_PDO_T pFunction)

    coEventRegister_PDO_REC_EVENT - register receive PDO event
- EXTERN_DECL RET_T coEventRegister_PDO_UPDATE (CO_EVENT_PDO_UPDATE_T pFunction)

    coEventRegister_PDO_UPDATE - register PDO Data Update event

5.42.1 Detailed Description

defines for pdo service

- contains defines for pdo service

5.42.2 Typedef Documentation

5.42.2.1 typedef void(∗ CO_EVENT_MPDO_T) (UNSIGNED16, UNSIGNED16, UNSIGNED8)

function pointer to MPDO indication

Parameters

pdoNr	- PDO number
index	- Index
subIndex	- subIndex

Returns

void

5.42.2.2 typedef void(∗ CO_EVENT_PDO_T) (UNSIGNED16)

function pointer to PDO indication

Parameters

pdoNr	- PDO number

Returns

void

5.42.2.3 typedef void(∗ CO_EVENT_PDO_UPDATE_T) (UNSIGNED16, UNSIGNED8)

function pointer to PDO update indication

Parameters

index	
subindex	

Returns

void

5.42.3 Function Documentation

5.42.3.1 EXTERN_DECL RET_T coEventRegister_PDO (CO_EVENT_PDO_T *pFunction*)

coEventRegister_PDO - register asynchronous PDO event

Register an indication function for asynchrounous PDOs.

After a PDO has been received, the data are stored in the object dictionary, and then the given indication function is called. This function is only valid for asynchronous PDOs.

Returns

RET_T

Parameters

pFunction	pointer to function

5.42.3.2 EXTERN_DECL RET_T coEventRegister_PDO_REC_EVENT (CO_EVENT_PDO_T *pFunction*)

coEventRegister_PDO_REC_EVENT - register receive PDO event

Register an indication function for receive PDO event.

For monitoring of receive PDOs the event timer can be used. If the event timer value is unequal 0 then after the reception of a PDO the monitoring is started automatically. if no further PDO in the given time was received, the indication function given to this function is called.

Returns

RET_T

Parameters

pFunction	pointer to function

5.42.3.3 EXTERN_DECL RET_T coEventRegister_PDO_SYNC (CO_EVENT_PDO_T *pFunction*)

coEventRegister_PDO_SYNC - register PDO SYNC event

Register an indication function for received synchronous PDOs.

After the SYNC was received, the received data are stored in the object dictionary, and then this given indication function is called.

Returns

RET_T

Parameters

pFunction	pointer to function

5.42.3.4 EXTERN_DECL RET_T coEventRegister_PDO_UPDATE (CO_EVENT_PDO_UPDATE_T *pFunction*)

coEventRegister_PDO_UPDATE - register PDO Data Update event

Register an indication function for PDO Data Update.

Before a PDO is transmitted, this function is called. Application can now update the given objects at the OD before the data are transmitted with a PDO.

Returns

RET_T

Parameters

pFunction	pointer to function

5.42.3.5 EXTERN_DECL BOOL_T coPdoObjIsMapped (UNSIGNED16 *pdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex*)

coPdoObjIsMapped - check, if object mapped to given PDO

This function checks the PDO, if the given object is actual mapped to this PDO. (Only valid for receive PDOs)

Returns

BOOL_T

Return values

CO_TRUE	object is mapped
CO_FALSE	object is not mapped

Parameters

pdoNr	PDO number
index	index of mapped object
subIndex	subindex of mapped object

5.42.3.6 EXTERN_DECL RET_T coPdoReceiveInit (UNSIGNED16 *pdoNr,* UNSIGNED8 *transType,* UNSIGNED16 *inhibit,* UNSIGNED16 *eventTime,* CO_CONST PDO_REC_MAP_TABLE_T $*$ *mapTable*)

coPdoReceiveInit - init receive pdo functionality

This function initializes a receive PDO. The COB-ID is set at reset communication or at load parameter.

Note: All parameter are reset by their default values at reset communication.

Returns

RET_T

Parameters

pdoNr	PDO number
transType	transmission type
inhibit	inhibit time 100 usec
eventTime	event timer in msec
mapTable	pointer to mapping table

5.42.3.7 EXTERN_DECL RET_T coPdoReqNr (UNSIGNED16 *pdoNr,* UNSIGNED8 *flags*)

coPdoReqNr - request PDO transmission by PDO number

This function requests the transmission of an PDO given by its number.
All mapped objects are automatically copied into the CAN message. If the inhibit time is not active, then the message is transmitted immediately.

If the inhibit time is not ellapsed yet, the transmission depends on the parameter flags:

0 - PDO will be transmitted after inhibit is ellapsed (if data are not changed, PDO will not be transmitted more than once!) MSG_OVERWRITE - if the last PDO is not transmitted yet, overwrite the last data with the new data MSG_RET_INHIBIT - return the function with RET_INHIBIT_ACTIVE, if the inhibit is not ellapsed yet

with the same or

Returns

RET_T

Return values

RET_INVALID_NMT_STATE	invalid NMT state
RET_INVALID_PARAMETER	unknown PDO number
RET_COB_DISABLED	PDO is disabled
RET_INHIBIT_ACTIVE	inhibit time is not yet ellapsed
RET_OK	all function are ok, but have not to be transmitted yet

Parameters

pdoNr	PDO number
flags	transmit flags

5.42.3.8 EXTERN_DECL RET_T coPdoReqObj (UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 *flags*)

coPdoReqObj - request PDO transmission by object

This function requests the transmission of the PDO, which the given object is mapped into.
All mapped objects are automatically copied into the CAN message. If the inhibit time is not active, then the message is transmitted immediately.

If the inhibit time is not ellapsed yet, the transmission depends on the parameter flags:

0 - PDO will be transmitted after inhibit is ellapsed MSG_OVERWRITE - if the last PDO is not transmitted yet, overwrite the last data with the new data MSG_RET_INHIBIT - return the function with RET_INHIBIT_ACTIVE, if the inhibit is not ellapsed yet

Returns

RET_T

Return values

RET_INVALID_NMT_STATE	invalid NMT state
RET_INVALID_PARAMETER	unknown PDO number
RET_COB_DISABLED	PDO is disabled
RET_INHIBIT_ACTIVE	inhibit time is not yet ellapsed
RET_OK	all function are ok, but have not to be transmitted yet

Parameters

index	index of mapped object
subIndex	subindex of mapped object
flags	transmit flags

5.42.3.9 EXTERN_DECL RET_T coPdoTransmitInit (UNSIGNED16 *pdoNr*, UNSIGNED8 *transType*, UNSIGNED16 *inhibit*, UNSIGNED16 *eventTime*, UNSIGNED8 *syncStartVal*, CO_CONST PDO_TR_MAP_TABLE_T ∗ *mapTable*)

coPdoTransmitInit - init transmit pdo functionality

This function initializes a transmit PDO. The COB-ID is set at reset communication or at load parameter.

Note: All parameters are reset to their default values at reset communication.

Returns

RET_T

Parameters

pdoNr	PDO number
transType	transmission type
inhibit	inhibit time 100 usec
eventTime	event timer in msec
syncStartVal	sync start value
mapTable	pointer to mapping table

5.43 co_queue.c File Reference

Queue handling.

Functions

- BOOL_T coQueueReceiveMessageAvailable (void)

  coQueueReceiveMessageAvailable - receive messages available
- CO_CAN_MSG_T ∗ coQueueGetNextTransmitMessage (void)

coQueueGetNextTransmitMessage - get next message to transmit
- void coQueueMsgTransmitted (const CO_CAN_MSG_T ∗pBuf)

  coQueueMsgTransmitted - message was transmitted
- void coQueueInit (void)

  coQueueInit - (re)init queues

5.43.1 Detailed Description

Queue handling.

contains functions for queue handling

5.43.2 Function Documentation

5.43.2.1 CO_CAN_MSG_T∗ coQueueGetNextTransmitMessage (void)

coQueueGetNextTransmitMessage - get next message to transmit

This function returns the next available transmit message from the transmit queue. It increments also trBufferRdCnt.

Returns

CO_CAN_MSG_T∗ pointer to next tx message

Return values

!NULL	pointer to transmit queue entry
NULL	no message available

5.43.2.2 void coQueueInit (void)

coQueueInit - (re)init queues

This function clears the transmit and the receive queue

Returns

none

5.43.2.3 void coQueueMsgTransmitted (const CO_CAN_MSG_T ∗ *pBuf*)

coQueueMsgTransmitted - message was transmitted

This function is called after a message was succesfull transmitted.

Returns

none

Parameters

pBuf	pointer to transmitted message

5.43.2.4 BOOL_T coQueueReceiveMessageAvailable (void)

coQueueReceiveMessageAvailable - receive messages available

This functions checks the receive queue for new messages. Are new messages available, return CO_TRUE. Otherwise CO_FALSE

Return values

CO_FALSE	no data available
CO_FALSE	data available

5.44 co_sdo.h File Reference

defines for sdo service

Typedefs

- typedef RET_T(∗ CO_EVENT_SDO_SERVER_T) (BOOL_T, UNSIGNED8, UNSIGNED16, UNSIGNED8)

  function pointer to SDO server event
- typedef RET_T(∗ CO_EVENT_SDO_SERVER_CHECK_WRITE_T) (BOOL_T, UNSIGNED8, UNSIGNE↩ D16, UNSIGNED8, const UNSIGNED8 ∗)

  function pointer to SDO server write check event
- typedef void(∗ CO_EVENT_SDO_SERVER_DOMAIN_WRITE_T) (UNSIGNED16, UNSIGNED8, UNSIGN↩ ED32, UNSIGNED32)

  function pointer to SDO server write domain event
- typedef void(∗ CO_EVENT_SDO_CLIENT_READ_T) (UNSIGNED8, UNSIGNED16, UNSIGNED8, UNSI↩ GNED32, UNSIGNED32)

  function pointer to SDO client read event
- typedef void(∗ CO_EVENT_SDO_CLIENT_WRITE_T) (UNSIGNED8, UNSIGNED16, UNSIGNED8, UNSI↩ GNED32)

  function pointer to SDO client write event
- typedef RET_T(∗ CO_EVENT_SDO_CLIENT_DOMAIN_WRITE_T) (UNSIGNED8, UNSIGNED16, UNSI↩ GNED8, UNSIGNED32, void ∗)

  function pointer to SDO client domain write event

Functions

- EXTERN_DECL RET_T coSdoServerInit (UNSIGNED8)

  coInitSdoServer - init sdo server functionality
- EXTERN_DECL RET_T coEventRegister_SDO_SERVER_READ (CO_EVENT_SDO_SERVER_T p↩ Function)

coEventRegister_SdoServer - register SDO server event

- EXTERN_DECL RET_T coEventRegister_SDO_SERVER_WRITE (CO_EVENT_SDO_SERVER_T p↩
Function)

    coEventRegister_SdoServerWrite - register SDO server write event

- EXTERN_DECL RET_T coEventRegister_SDO_SERVER_CHECK_WRITE (CO_EVENT_SDO_SERVER↩
_CHECK_WRITE_T pFunction)

    coEventRegister_SdoServerCheckWrite - register SDO server write event

- EXTERN_DECL RET_T coSdoClientInit (UNSIGNED8)

    coInitSdoClient - init SDO client functionality

- EXTERN_DECL RET_T coSdoRead (UNSIGNED8 sdoNr, UNSIGNED16 index, UNSIGNED8 subIndex, U↩
NSIGNED8 ∗pData, UNSIGNED32 dataLen, UNSIGNED16 numeric, UNSIGNED32 timeout)

    coSdoRead - read value by SDO

- EXTERN_DECL RET_T coSdoWrite (UNSIGNED8 sdoNr, UNSIGNED16 index, UNSIGNED8 subIndex, U↩
NSIGNED8 ∗pData, UNSIGNED32 dataLen, UNSIGNED16 numeric, UNSIGNED32 timeout)

    coSdoWrite - Write value by SDO

- EXTERN_DECL RET_T coSdoQueueAddTransfer (BOOL_T write, UNSIGNED8 sdoNr, UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pData, UNSIGNED32 dataLen, CO_SDO_QUEUE_IND_T pFct, void ∗pFctPara)

    coSdoQueueAddTransfer - add sdo transfer to sdo queue handler

- EXTERN_DECL RET_T coSdoQueueAddOdTransfer (BOOL_T write, UNSIGNED8 sdoNr, UNSIGNED16 remoteIndex, UNSIGNED8 remoteSubIndex, UNSIGNED16 localIndex, UNSIGNED8 localSubIndex, CO_↩
SDO_QUEUE_IND_T pFct, void ∗pFctPara)

    coSdoQueueAddOdTransfer - add sdo transfer to sdo queue handler

- EXTERN_DECL RET_T coSdoReadSeg (UNSIGNED8 sdoNr, UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pData, UNSIGNED32 dataLen, UNSIGNED16 numeric, UNSIGNED32 timeout)

    coSdoReadSeg - read value by segmented SDO

- EXTERN_DECL RET_T coSdoWriteSeg (UNSIGNED8 sdoNr, UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pData, UNSIGNED32 dataLen, UNSIGNED16 numeric, UNSIGNED32 timeout)

    coSdoWriteSeg - Write value by segmented SDO

- EXTERN_DECL RET_T coSdoClientAbortTransfer (UNSIGNED8 sdoNr, RET_T errorReason)

    coSdoClientAbortTransfer - abort SDO transfer

- EXTERN_DECL RET_T coEventRegister_SDO_CLIENT_READ (CO_EVENT_SDO_CLIENT_READ_T p↩
Function)

    coEventRegister_SdoClientRead - register SDO client read event

- EXTERN_DECL RET_T coEventRegister_SDO_CLIENT_WRITE (CO_EVENT_SDO_CLIENT_WRITE_↩
T pFunction)

    coEventRegister_SdoClientWrite - register SDO client write event

- EXTERN_DECL RET_T coEventUnregister_SDO_CLIENT_READ (CO_EVENT_SDO_CLIENT_READ_↩
T pFunction)

    coEventUnregister_SDO_CLIENT_READ - unregister SDO client read event

- EXTERN_DECL RET_T coEventUnregister_SDO_CLIENT_WRITE (CO_EVENT_SDO_CLIENT_WRITE_T pFunction)

    coEventUnregister_SDO_CLIENT_WRITE - unregister SDO client write event

- EXTERN_DECL RET_T coSdoServerReadIndCont (UNSIGNED8 sdoNr, RET_T result)

    coSdoServerReadIndCont - continue SDO server Read indication

- EXTERN_DECL RET_T coSdoServerWriteIndCont (UNSIGNED8 sdoNr, RET_T result)

    coSdoServerWriteIndCont - continue SDO server Write indication

- EXTERN_DECL RET_T coSdoNetworkRead (UNSIGNED8 sdoNr, UNSIGNED16 network, UNSIGNED8 node, UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pData, UNSIGNED32 dataLen, UNS↩
IGNED16 numeric, UNSIGNED32 timeout)

    coSdoNetworkRead - read network value by SDO

- EXTERN_DECL RET_T coSdoNetworkWrite (UNSIGNED8 sdoNr, UNSIGNED16 network, UNSIGNED8 node, UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pData, UNSIGNED32 dataLen, UNS↩
IGNED16 numeric, UNSIGNED32 timeout)

    coSdoNetworkWrite - Write value by SDO

5.44.1 Detailed Description

defines for sdo service

- contains defines for sdo service

5.44.2 Typedef Documentation

5.44.2.1 typedef RET_T(∗ CO_EVENT_SDO_CLIENT_DOMAIN_WRITE_T) (UNSIGNED8, UNSIGNED16, UNSIGNED8, UNSIGNED32, void ∗)

function pointer to SDO client domain write event

Parameters

sdoNr	- sdo number
index	- object index
subindex	- object subindex
transfered	- bytes transfered
pointer	- pointer to application data

Returns

RET_T

5.44.2.2 typedef void(∗ CO_EVENT_SDO_CLIENT_READ_T) (UNSIGNED8, UNSIGNED16, UNSIGNED8, UNSIGNED32, UNSIGNED32)

function pointer to SDO client read event

Parameters

sdoNr	- sdo number
index	- object index
subindex	- object subindex
size	- size of received data
result	- result of transfer

Returns

void

5.44.2.3 typedef void(∗ CO_EVENT_SDO_CLIENT_WRITE_T) (UNSIGNED8, UNSIGNED16, UNSIGNED8, UNSIGNED32)

function pointer to SDO client write event

Parameters

sdoNr	- sdo number
index	- object index
subindex	- object subindex
result	- result of transfer

Returns

void

5.44.2.4 typedef RET_T(∗ CO_EVENT_SDO_SERVER_CHECK_WRITE_T) (BOOL_T, UNSIGNED8, UNSIGNED16, UNSIGNED8, const UNSIGNED8 ∗)

function pointer to SDO server write check event

Parameters

execute	- execute or test only
sdoNr	- sdo number
index	- object index
subindex	- object subindex
pData	- pointer to receive buffer

Returns

RET_T

5.44.2.5 typedef void(∗ CO_EVENT_SDO_SERVER_DOMAIN_WRITE_T) (UNSIGNED16, UNSIGNED8, UNSIGNED32, UNSIGNED32)

function pointer to SDO server write domain event

Parameters

index	- object index
subindex	- object subindex
domainBufSize	- actual size at domain buffer
transferSize	- actual transfered size

Returns

RET_T

5.44.2.6 typedef RET_T(∗ CO_EVENT_SDO_SERVER_T) (BOOL_T, UNSIGNED8, UNSIGNED16, UNSIGNED8)

function pointer to SDO server event

Parameters

execute	- execute or test only
sdoNr	- sdo number
index	- object index
subindex	- object subindex

Returns

RET_T

5.44.3 Function Documentation

5.44.3.1 EXTERN_DECL RET_T coEventRegister_SDO_CLIENT_READ (CO_EVENT_SDO_CLIENT_READ_T *pFunction*)

coEventRegister_SdoClientRead - register SDO client read event

This function registers the sdo read indication function. It is called after a SDO read, started by coSdoRead() was finished.

Returns

RET_T

Parameters

pFunction	pointer to function

5.44.3.2 EXTERN_DECL RET_T coEventRegister_SDO_CLIENT_WRITE (CO_EVENT_SDO_CLIENT_WRITE_T *pFunction*)

coEventRegister_SdoClientWrite - register SDO client write event

This function registers the sdo write indication function. It is called after a SDO write, started by coSdoWrite() was finished.

Returns

RET_T

5.44.3.3 EXTERN_DECL RET_T coEventRegister_SDO_SERVER_CHECK_WRITE (CO_EVENT_SDO_SERVER_CHE↩ CK_WRITE_T *pFunction*)

coEventRegister_SdoServerCheckWrite - register SDO server write event

This function register a sdo server indication function, which is called before SDO write access is executed, so the application can reject an SDO write access.

Returns

RET_T

Parameters

pFunction	pointer to function

5.44.3.4 EXTERN_DECL RET_T coEventRegister_SDO_SERVER_READ (CO_EVENT_SDO_SERVER_T *pFunction*)

coEventRegister_SdoServer - register SDO server event

This function registers a sdo server indication function, which is called before a SDO read request is executed, so the application can update the data before the response is sent.

Returns

RET_T

Parameters

pFunction	pointer to function

5.44.3.5 EXTERN_DECL RET_T coEventRegister_SDO_SERVER_WRITE (CO_EVENT_SDO_SERVER_T *pFunction*)

coEventRegister_SdoServerWrite - register SDO server write event

This function registers a SDO server write indication function. It is called, after a SDO write access was finished.

Returns

RET_T

Parameters

pFunction	pointer to function

5.44.3.6 EXTERN_DECL RET_T coEventUnregister_SDO_CLIENT_READ (CO_EVENT_SDO_CLIENT_READ_T *pFunction*)

coEventUnregister_SDO_CLIENT_READ - unregister SDO client read event

This function unregisters the sdo read indication function.

Returns

RET_T

Parameters

pFunction	pointer to function

5.44.3.7 EXTERN_DECL RET_T coEventUnregister_SDO_CLIENT_WRITE (CO_EVENT_SDO_CLIENT_WRITE_T *pFunction*)

coEventUnregister_SDO_CLIENT_WRITE - unregister SDO client write event

This function unregisters the sdo write indication function.

Returns

RET_T

Parameters

pFunction	pointer to function

5.44.3.8 EXTERN_DECL RET_T coSdoClientAbortTransfer (UNSIGNED8 *sdoNr*, RET_T *errorReason*)

coSdoClientAbortTransfer - abort SDO transfer

This function aborts a running SDO transfer with the given abort reason.

Returns

RET_T

Parameters

sdoNr	sdo number
errorReason	error reason

5.44.3.9 EXTERN_DECL RET_T coSdoClientInit (UNSIGNED8 *clientNr*)

coInitSdoClient - init SDO client functionality

This function initializes the SDO client with the given number.

Returns

RET_T

Parameters

clientNr	sdo client number

5.44.3.10 EXTERN_DECL RET_T coSdoNetworkRead (UNSIGNED8 *sdoNr,* UNSIGNED16 *network,* UNSIGNED8 *node,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoNetworkRead - read network value by SDO

This function starts a sdo read transfer over a network to the given network/node and SDO parameters.

As first, the network connection to the router is etablished and than the normal SDO transfer ist started.

The result is given by the standard SDO client indication functions.

The data are stored at the given pointer pData with a maximal length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. Initialization is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

Returns

RET_T

Parameters

sdoNr	sdo number
network	network number
node	node number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.44.3.11 EXTERN_DECL RET_T coSdoNetworkWrite (UNSIGNED8 *sdoNr,* UNSIGNED16 *network,* UNSIGNED8 *node,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoNetworkWrite - Write value by SDO

This function starts a sdo read transfer over a network to the given network/node and SDO parameters.

As first, the network connection to the router is etablished and than the normal SDO transfer ist started.

The result is given by the standard SDO client indication functions.

The data are written from the given pointer pData and with a length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. This is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

If SDO block transfer is enabled, it will be used automatically if dataLen is larger than CO_SDO_BLOCK_MIN_SIZE. If the server doesn't support block transfer, segmented transfer will be used instead.

Returns

RET_T

Parameters

sdoNr	sdo number
network	network number
node	node number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.44.3.12 EXTERN_DECL RET_T coSdoQueueAddOdTransfer (BOOL_T *write*, UNSIGNED8 *sdoNr*, UNSIGNED16 *remoteIndex*, UNSIGNED8 *remoteSubIndex*, UNSIGNED16 *localIndex*, UNSIGNED8 *localSubIndex*, CO_SDO_QUEUE_IND_T *pFct*, void ∗ *pFctPara*)

coSdoQueueAddOdTransfer - add sdo transfer to sdo queue handler

This function can be used to add sdo transfers from local object dicationary to remote dictionary to a queue. If a tranfer was finished, the next will start automatically. After each transfer, the given function with the parameter are called.

Please note: Only allowed for expedited transfers with initialized sdo channel. Transmit data are saved internally.

Returns

RET_T

Parameters

write	write/read access
sdoNr	sdo number
remoteIndex	remote index
remoteSubIndex	remote subIndex
localIndex	local index
localSubIndex	local subindex
pFct	pointer to finish function
pFctPara	pointer to data field for finish function

5.44.3.13 EXTERN_DECL RET_T coSdoQueueAddTransfer (BOOL_T *write,* UNSIGNED8 *sdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* CO_SDO_QUEUE_IND_T *pFct,* void ∗ *pFctPara*)

coSdoQueueAddTransfer - add sdo transfer to sdo queue handler

This function can be used to add sdo transfers to a queue. If a tranfer was finished, the next will start automatically. After each transfer, the given function with the parameter are called.

Please note: Only allowed for expedited transfers with initialized sdo channel. Transmit data are saved internally.

Returns

RET_T

Parameters

write	write/read access
sdoNr	sdo number
index	index
subIndex	subIndex
pData	pointer to transfer data
dataLen	len of transfer data
pFct	pointer to finish function
pFctPara	pointer to data field for finish function

5.44.3.14 EXTERN_DECL RET_T coSdoRead (UNSIGNED8 *sdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoRead - read value by SDO

This function starts a sdo transfer with the given parameters. The data are stored at the given pointer pData with a maximal length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. Initialization is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

If SDO block transfer is enabled, it will be used automatically if dataLen is larger than CO_SDO_BLOCK_MIN_SIZE. If the server doesn't support block transfer, segmented transfer will be used instead.

Returns

RET_T

Parameters

sdoNr	sdo number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.44.3.15 EXTERN_DECL RET_T coSdoReadSeg (UNSIGNED8 *sdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoReadSeg - read value by segmented SDO

This function starts a sdo transfer with the given parameters. The data are stored at the given pointer pData with a maximal length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. Initialization is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

The segmented transfer will be used.

Returns

RET_T

Parameters

sdoNr	sdo number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.44.3.16 EXTERN_DECL RET_T coSdoServerInit (UNSIGNED8 *sdoServerNr*)

coInitSdoServer - init sdo server functionality

This function initializes the given sdo server. If the sdo number = 1, then the default COB-IDs are set for this SDO.

Returns

RET_T

Parameters

sdoServerNr	sdo server number

5.44.3.17 EXTERN_DECL RET_T coSdoServerReadIndCont (UNSIGNED8 *sdoNr,* RET_T *result*)

coSdoServerReadIndCont - continue SDO server Read indication

This function has to be called, after the sdoServerReadInd function has returned RET_SDO_SPLIT_INDICATION to continue and finish the SDO transfer

The result parameter should contain the result for the transfer

Returns

RET_T

Parameters

sdoNr	sdo number
result	result for transfer

5.44.3.18 EXTERN_DECL RET_T coSdoServerWriteIndCont (UNSIGNED8 *sdoNr,* RET_T *result*)

coSdoServerWriteIndCont - continue SDO server Write indication

This function has to be called, after the sdoServerWriteInd function has returned RET_SDO_SPLIT_INDICATION to continue and finish the SDO transfer

The result parameter should contain the result for the transfer.

Returns

RET_T

Parameters

sdoNr	sdo number
result	result indication

5.44.3.19 EXTERN_DECL RET_T coSdoWrite (UNSIGNED8 *sdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoWrite - Write value by SDO

This function starts a sdo write transfer with the given parameter. The data are read from the given pointer pData and with a length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. This is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

If SDO block transfer is enabled, it will be used automatically if dataLen is larger than CO_SDO_BLOCK_MIN_SIZE. If the server doesn't support block transfer, segmented transfer will be used instead.

Returns

RET_T

Parameters

sdoNr	sdo number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.44.3.20 EXTERN_DECL RET_T coSdoWriteSeg (UNSIGNED8 *sdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoWriteSeg - Write value by segmented SDO

This function starts a sdo write transfer with the given parameter. The data are read from the given pointer pData and with a length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. This is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

The segmented transfer will be used.

Returns

RET_T

Parameters

sdoNr	sdo number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.45 co_sdoblockclient.c File Reference

sdo block routines

5.45.1 Detailed Description

sdo block routines

contains sdo block transfer routines for client

5.46 co_sdoblockserver.c File Reference

sdo block routines

5.46.1 Detailed Description

sdo block routines

contains sdo block transfer routines for server

5.47 co_sdoclient.c File Reference

sdo client routines

Functions

- RET_T coSdoRead (UNSIGNED8 sdoNr, UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗p↩ Data, UNSIGNED32 dataLen, UNSIGNED16 numeric, UNSIGNED32 timeout)

    coSdoRead - read value by SDO
- RET_T coSdoReadSeg (UNSIGNED8 sdoNr, UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pData, UNSIGNED32 dataLen, UNSIGNED16 numeric, UNSIGNED32 timeout)

    coSdoReadSeg - read value by segmented SDO
- RET_T coSdoWrite (UNSIGNED8 sdoNr, UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗p↩ Data, UNSIGNED32 dataLen, UNSIGNED16 numeric, UNSIGNED32 timeout)

    coSdoWrite - Write value by SDO
- RET_T coSdoWriteSeg (UNSIGNED8 sdoNr, UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pData, UNSIGNED32 dataLen, UNSIGNED16 numeric, UNSIGNED32 timeout)

    coSdoWriteSeg - Write value by segmented SDO
- RET_T coSdoClientAbortTransfer (UNSIGNED8 sdoNr, RET_T errorReason)

    coSdoClientAbortTransfer - abort SDO transfer
- RET_T coEventRegister_SDO_CLIENT_READ (CO_EVENT_SDO_CLIENT_READ_T pFunction)

    coEventRegister_SdoClientRead - register SDO client read event
- RET_T coEventUnregister_SDO_CLIENT_READ (CO_EVENT_SDO_CLIENT_READ_T pFunction)

    coEventUnregister_SDO_CLIENT_READ - unregister SDO client read event
- RET_T coEventRegister_SDO_CLIENT_WRITE (CO_EVENT_SDO_CLIENT_WRITE_T pFunction)

    coEventRegister_SdoClientWrite - register SDO client write event
- RET_T coEventUnregister_SDO_CLIENT_WRITE (CO_EVENT_SDO_CLIENT_WRITE_T pFunction)

    coEventUnregister_SDO_CLIENT_WRITE - unregister SDO client write event
- RET_T coSdoClientInit (UNSIGNED8 clientNr)

    coInitSdoClient - init SDO client functionality

5.47.1 Detailed Description

sdo client routines

contains sdo client routines

5.47.2 Function Documentation

5.47.2.1 RET_T coEventRegister_SDO_CLIENT_READ (CO_EVENT_SDO_CLIENT_READ_T *pFunction*)

coEventRegister_SdoClientRead - register SDO client read event

This function registers the sdo read indication function. It is called after a SDO read, started by coSdoRead() was finished.

Returns

RET_T

Parameters

pFunction	pointer to function

5.47.2.2 RET_T coEventRegister_SDO_CLIENT_WRITE (CO_EVENT_SDO_CLIENT_WRITE_T *pFunction*)

coEventRegister_SdoClientWrite - register SDO client write event

This function registers the sdo write indication function. It is called after a SDO write, started by coSdoWrite() was finished.

Returns

RET_T

5.47.2.3 RET_T coEventUnregister_SDO_CLIENT_READ (CO_EVENT_SDO_CLIENT_READ_T *pFunction*)

coEventUnregister_SDO_CLIENT_READ - unregister SDO client read event

This function unregisters the sdo read indication function.

Returns

RET_T

Parameters

pFunction	pointer to function

5.47.2.4 RET_T coEventUnregister_SDO_CLIENT_WRITE (CO_EVENT_SDO_CLIENT_WRITE_T *pFunction*)

coEventUnregister_SDO_CLIENT_WRITE - unregister SDO client write event

This function unregisters the sdo write indication function.

Returns

RET_T

Parameters

pFunction	pointer to function

5.47.2.5 RET_T coSdoClientAbortTransfer (UNSIGNED8 *sdoNr,* RET_T *errorReason*)

coSdoClientAbortTransfer - abort SDO transfer

This function aborts a running SDO transfer with the given abort reason.

Returns

RET_T

Parameters

sdoNr	sdo number
errorReason	error reason

5.47.2.6 RET_T coSdoClientInit (UNSIGNED8 *clientNr*)

coInitSdoClient - init SDO client functionality

This function initializes the SDO client with the given number.

Returns

RET_T

Parameters

clientNr	sdo client number

5.47.2.7 RET_T coSdoRead (UNSIGNED8 *sdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoRead - read value by SDO

This function starts a sdo transfer with the given parameters. The data are stored at the given pointer pData with a maximal length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. Initialization is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

If SDO block transfer is enabled, it will be used automatically if dataLen is larger than CO_SDO_BLOCK_MIN_SIZE. If the server doesn't support block transfer, segmented transfer will be used instead.

Returns

RET_T

Parameters

sdoNr	sdo number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.47.2.8 RET_T coSdoReadSeg (UNSIGNED8 *sdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoReadSeg - read value by segmented SDO

This function starts a sdo transfer with the given parameters. The data are stored at the given pointer pData with a maximal length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. Initialization is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

The segmented transfer will be used.

Returns

RET_T

Parameters

sdoNr	sdo number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.47.2.9 RET_T coSdoWrite (UNSIGNED8 *sdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoWrite - Write value by SDO

This function starts a sdo write transfer with the given parameter. The data are read from the given pointer pData and with a length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. This is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

If SDO block transfer is enabled, it will be used automatically if dataLen is larger than CO_SDO_BLOCK_MIN_SIZE. If the server doesn't support block transfer, segmented transfer will be used instead.

Returns

RET_T

Parameters

sdoNr	sdo number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.47.2.10 RET_T coSdoWriteSeg (UNSIGNED8 *sdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoWriteSeg - Write value by segmented SDO

This function starts a sdo write transfer with the given parameter. The data are read from the given pointer pData and with a length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. This is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

The segmented transfer will be used.

Returns

RET_T

Parameters

sdoNr	sdo number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.48 co_sdonetwork.c File Reference

sdo network routines

Functions

- RET_T coSdoNetworkRead (UNSIGNED8 sdoNr, UNSIGNED16 network, UNSIGNED8 node, UNSIGNE↩ D16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pData, UNSIGNED32 dataLen, UNSIGNED16 numeric, UNSIGNED32 timeout)

    coSdoNetworkRead - read network value by SDO
- RET_T coSdoNetworkWrite (UNSIGNED8 sdoNr, UNSIGNED16 network, UNSIGNED8 node, UNSIGNE↩ D16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pData, UNSIGNED32 dataLen, UNSIGNED16 numeric, UNSIGNED32 timeout)

    coSdoNetworkWrite - Write value by SDO

5.48.1 Detailed Description

sdo network routines

contains sdo network transfer routines for server

5.48.2 Function Documentation

5.48.2.1 RET_T coSdoNetworkRead (UNSIGNED8 *sdoNr,* UNSIGNED16 *network,* UNSIGNED8 *node,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoNetworkRead - read network value by SDO

This function starts a sdo read transfer over a network to the given network/node and SDO parameters.

As first, the network connection to the router is etablished and than the normal SDO transfer ist started.

The result is given by the standard SDO client indication functions.

The data are stored at the given pointer pData with a maximal length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. Initialization is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

Returns

RET_T

Parameters

sdoNr	sdo number
network	network number
node	node number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.48.2.2 RET_T coSdoNetworkWrite (UNSIGNED8 *sdoNr,* UNSIGNED16 *network,* UNSIGNED8 *node,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* UNSIGNED16 *numeric,* UNSIGNED32 *timeout*)

coSdoNetworkWrite - Write value by SDO

This function starts a sdo read transfer over a network to the given network/node and SDO parameters.

As first, the network connection to the router is etablished and than the normal SDO transfer ist started.

The result is given by the standard SDO client indication functions.

The data are written from the given pointer pData and with a length of dataLen.
The timeout value given in msec is started with each message transmission.
The numeric flag is only valid for big-endian transfers. If this parameter is set, the data are changed to little endian format.

Before an SDO can be started, it has to be initialized. This is done by setup the COB-Ids of this SDO at index 0x128x:1 and 0x128x:2

If SDO block transfer is enabled, it will be used automatically if dataLen is larger than CO_SDO_BLOCK_MIN_SIZE.
If the server doesn't support block transfer, segmented transfer will be used instead.

Returns

RET_T

Parameters

sdoNr	sdo number
network	network number
node	node number
index	index at server OD
subIndex	index at server OD
pData	pointer to transfer data
dataLen	data len for transfer
numeric	numeric flag (only for big endian)
timeout	timeout in msec

5.49 co_sdoqueue.c File Reference

SDO handling with queuing.

Functions

- RET_T coSdoQueueAddTransfer (BOOL_T write, UNSIGNED8 sdoNr, UNSIGNED16 index, UNSIGNED8 subIndex, UNSIGNED8 ∗pData, UNSIGNED32 dataLen, CO_SDO_QUEUE_IND_T pFct, void ∗pFctPara)

  coSdoQueueAddTransfer - add sdo transfer to sdo queue handler
- RET_T coSdoQueueAddOdTransfer (BOOL_T write, UNSIGNED8 sdoNr, UNSIGNED16 remoteIndex, U←NSIGNED8 remoteSubIndex, UNSIGNED16 localIndex, UNSIGNED8 localSubIndex, CO_SDO_QUEUE_←IND_T pFct, void ∗pFctPara)

  coSdoQueueAddOdTransfer - add sdo transfer to sdo queue handler

5.49.1 Detailed Description

SDO handling with queuing.

5.49.2 Function Documentation

5.49.2.1 RET_T coSdoQueueAddOdTransfer (BOOL_T *write,* UNSIGNED8 *sdoNr,* UNSIGNED16 *remoteIndex,* UNSIGNED8 *remoteSubIndex,* UNSIGNED16 *localIndex,* UNSIGNED8 *localSubIndex,* CO_SDO_QUEUE_IND_T *pFct,* void ∗ *pFctPara*)

coSdoQueueAddOdTransfer - add sdo transfer to sdo queue handler

This function can be used to add sdo transfers from local object dicationary to remote dictionary to a queue. If a tranfer was finished, the next will start automatically. After each transfer, the given function with the parameter are called.

Please note: Only allowed for expedited transfers with initialized sdo channel. Transmit data are saved internally.

Returns

RET_T

Parameters

write	write/read access
sdoNr	sdo number
remoteIndex	remote index
remoteSubIndex	remote subIndex
localIndex	local index
localSubIndex	local subindex
pFct	pointer to finish function
pFctPara	pointer to data field for finish function

5.49.2.2 RET_T coSdoQueueAddTransfer (BOOL_T *write,* UNSIGNED8 *sdoNr,* UNSIGNED16 *index,* UNSIGNED8 *subIndex,* UNSIGNED8 ∗ *pData,* UNSIGNED32 *dataLen,* CO_SDO_QUEUE_IND_T *pFct,* void ∗ *pFctPara*)

coSdoQueueAddTransfer - add sdo transfer to sdo queue handler

This function can be used to add sdo transfers to a queue. If a tranfer was finished, the next will start automatically. After each transfer, the given function with the parameter are called.

Please note: Only allowed for expedited transfers with initialized sdo channel. Transmit data are saved internally.

Returns

RET_T

Parameters

write	write/read access
sdoNr	sdo number
index	index
subIndex	subIndex
pData	pointer to transfer data
dataLen	len of transfer data
pFct	pointer to finish function
pFctPara	pointer to data field for finish function

5.50 co_sdoserv.c File Reference

SDO server routines.

Functions

- RET_T coSdoServerReadIndCont (UNSIGNED8 sdoNr, RET_T result)

  coSdoServerReadIndCont - continue SDO server Read indication
- RET_T coSdoServerWriteIndCont (UNSIGNED8 sdoNr, RET_T result)

  coSdoServerWriteIndCont - continue SDO server Write indication
- RET_T coEventRegister_SDO_SERVER_READ (CO_EVENT_SDO_SERVER_T pFunction)

  coEventRegister_SdoServer - register SDO server event
- RET_T coEventRegister_SDO_SERVER_CHECK_WRITE (CO_EVENT_SDO_SERVER_CHECK_WRIT←E_T pFunction)

  coEventRegister_SdoServerCheckWrite - register SDO server write event
- RET_T coEventRegister_SDO_SERVER_WRITE (CO_EVENT_SDO_SERVER_T pFunction)

  coEventRegister_SdoServerWrite - register SDO server write event
- RET_T coSdoServerInit (UNSIGNED8 sdoServerNr)

  coInitSdoServer - init sdo server functionality

5.50.1 Detailed Description

SDO server routines.

contains sdo server routines

5.50.2 Function Documentation

5.50.2.1 RET_T coEventRegister_SDO_SERVER_CHECK_WRITE (CO_EVENT_SDO_SERVER_CHECK_WRITE_T *pFunction*)

coEventRegister_SdoServerCheckWrite - register SDO server write event

This function register a sdo server indication function, which is called before SDO write access is executed, so the application can reject an SDO write access.

Returns

RET_T

Parameters

pFunction	pointer to function

5.50.2.2 RET_T coEventRegister_SDO_SERVER_READ (CO_EVENT_SDO_SERVER_T *pFunction*)

coEventRegister_SdoServer - register SDO server event

This function registers a sdo server indication function, which is called before a SDO read request is executed, so the application can update the data before the response is sent.

Returns

RET_T

Parameters

pFunction	pointer to function

5.50.2.3 RET_T coEventRegister_SDO_SERVER_WRITE (CO_EVENT_SDO_SERVER_T *pFunction*)

coEventRegister_SdoServerWrite - register SDO server write event

This function registers a SDO server write indication function. It is called, after a SDO write access was finished.

Returns

RET_T

Parameters

pFunction	pointer to function

5.50.2.4 RET_T coSdoServerInit (UNSIGNED8 *sdoServerNr*)

coInitSdoServer - init sdo server functionality

This function initializes the given sdo server. If the sdo number = 1, then the default COB-IDs are set for this SDO.

Returns

RET_T

Parameters

sdoServerNr	sdo server number

5.50.2.5 RET_T coSdoServerReadIndCont (UNSIGNED8 *sdoNr*, RET_T *result*)

coSdoServerReadIndCont - continue SDO server Read indication

This function has to be called, after the sdoServerReadInd function has returned RET_SDO_SPLIT_INDICATION to continue and finish the SDO transfer

The result parameter should contain the result for the transfer

Returns

RET_T

Parameters

sdoNr	sdo number
result	result for transfer

5.50.2.6 RET_T coSdoServerWriteIndCont (UNSIGNED8 *sdoNr*, RET_T *result*)

coSdoServerWriteIndCont - continue SDO server Write indication

This function has to be called, after the sdoServerWriteInd function has returned RET_SDO_SPLIT_INDICATION to continue and finish the SDO transfer

The result parameter should contain the result for the transfer.

Returns

RET_T

Parameters

sdoNr	sdo number
result	result indication

5.51 co_sleep.c File Reference

Sleep and Wakeup Handling.

Functions

- void coSleepModeStart (UNSIGNED16 waitTime)

    coSleepModeStart - start sleep mode
- void coSleepAwake (BOOL_T master, UNSIGNED8 status, UNSIGNED8 reason, UNSIGNED16 repeatTime)

    coSleepAwake - awake from sleep
- void coSleepWakeUp (BOOL_T master, UNSIGNED8 status, UNSIGNED8 reason, UNSIGNED16 repeat↩ Time)

    coSleepWakeUp - awake from sleep
- BOOL_T coSleepModeActive (void)

    coSleepModeActive - check if sleep mode is active
- void coSleepRequestSleep (void)

    coSleepRequestSleep - request sleep mode to master
- RET_T coEventRegister_SLEEP (CO_EVENT_SLEEP_T pFunction)

    coEventRegister_SLEEP - register SLEEP event

5.51.1 Detailed Description

Sleep and Wakeup Handling.

contains routines for sleep/wakeup handling

5.51.2 Function Documentation

5.51.2.1 RET_T coEventRegister_SLEEP (CO_EVENT_SLEEP_T *pFunction*)

coEventRegister_SLEEP - register SLEEP event

register indication function for SLEEP events

Returns

RET_T

Parameters

pFunction	pointer to function

5.51.2.2 void coSleepAwake (BOOL_T *master,* UNSIGNED8 *status,* UNSIGNED8 *reason,* UNSIGNED16 *repeatTime*)

coSleepAwake - awake from sleep

This function have to called after the processor is awaked. It transmit the wake up message, repeat it after the given repeatTime and reinitializes the communication handling.

Returns

none

Parameters

master	wake up master
status	wake up state (only for slaves)
reason	wake up reason
repeatTime	time interval for repeat wake up message

5.51.2.3 BOOL_T coSleepModeActive (void)

coSleepModeActive - check if sleep mode is active

Returns

none

5.51.2.4 void coSleepModeStart (UNSIGNED16 *waitTime*)

coSleepModeStart - start sleep mode

This function starts the sleep mode. First a timer is started, then the CAN traffic is stopped and the CPU is going to sleep.

Each step is indicated by the function registered by coEventRegister_SLEEP().

Returns

none

Parameters

waitTime	wait time before stop CAN in ms

5.51.2.5 void coSleepRequestSleep (void)

coSleepRequestSleep - request sleep mode to master

Request sleep mode from master by sending sleep request.

Returns

none

5.51.2.6 void coSleepWakeUp (BOOL_T *master,* UNSIGNED8 *status,* UNSIGNED8 *reason,* UNSIGNED16 *repeatTime*)

coSleepWakeUp - awake from sleep

This function can be called to send the wake up message independ form the actual sleep state. It transmit the wake up message, repeat it after the given repeatTime and reinitializes the communication handling.

Returns

none

Parameters

master	wake up master
status	wake up state (only for slaves)
reason	wake up reason
repeatTime	time interval for repeat wake up message

5.52 co_sleep.h File Reference

defines for sleep services

Typedefs

- typedef UNSIGNED8(∗ CO_EVENT_SLEEP_T) (CO_SLEEP_MODE_T, UNSIGNED8)

  function pointer to sleep event function

Enumerations

Functions

- EXTERN_DECL void coSleepModeStart (UNSIGNED16 waitTime)

  coSleepModeStart - start sleep mode
- EXTERN_DECL RET_T coEventRegister_SLEEP (CO_EVENT_SLEEP_T pFunction)

  coEventRegister_SLEEP - register SLEEP event
- EXTERN_DECL BOOL_T coSleepModeActive (void)

  coSleepModeActive - check if sleep mode is active
- EXTERN_DECL void coSleepAwake (BOOL_T master, UNSIGNED8 status, UNSIGNED8 reason, UNSIG↩ NED16 repeatTime)

  coSleepAwake - awake from sleep
- EXTERN_DECL void coSleepWakeUp (BOOL_T master, UNSIGNED8 status, UNSIGNED8 reason, UNSI↩ GNED16 repeatTime)

  coSleepWakeUp - awake from sleep
- EXTERN_DECL void coSleepRequestSleep (void)

  coSleepRequestSleep - request sleep mode to master

5.52.1 Detailed Description

defines for sleep services

- contains defines for sleep services

5.52.2 Typedef Documentation

5.52.2.1 typedef UNSIGNED8(∗ CO_EVENT_SLEEP_T) (CO_SLEEP_MODE_T, UNSIGNED8)

function pointer to sleep event function

Parameters

sleep	mode
node	id

Return values

0	- ok
!=0	- error

5.52.3 Enumeration Type Documentation

5.52.3.1 enum CO_SLEEP_MODE_T

SLEEP states

Enumerator

CO_SLEEP_MODE_CHECK	check if sleep mode is possible
CO_SLEEP_MODE_OBJECTION	slave has send an objection
CO_SLEEP_MODE_PREPARE	automatic start sleep mode
CO_SLEEP_MODE_SILENT	sleep mode silent
CO_SLEEP_MODE_DOZE	sleep mode doze
CO_SLEEP_MODE_REQUEST_SLEEP	sleep mode reuqest sleep

5.52.4 Function Documentation

5.52.4.1 EXTERN_DECL RET_T coEventRegister_SLEEP (CO_EVENT_SLEEP_T *pFunction*)

coEventRegister_SLEEP - register SLEEP event

register indication function for SLEEP events

Returns

RET_T

Parameters

pFunction	pointer to function

5.52.4.2 EXTERN_DECL void coSleepAwake (BOOL_T *master*, UNSIGNED8 *status*, UNSIGNED8 *reason*, UNSIGNED16 *repeatTime*)

coSleepAwake - awake from sleep

This function have to called after the processor is awaked. It transmit the wake up message, repeat it after the given repeatTime and reinitializes the communication handling.

Returns

none

Parameters

master	wake up master
status	wake up state (only for slaves)
reason	wake up reason
repeatTime	time interval for repeat wake up message

5.52.4.3 EXTERN_DECL BOOL_T coSleepModeActive (void)

coSleepModeActive - check if sleep mode is active

Returns

none

5.52.4.4 EXTERN_DECL void coSleepModeStart (UNSIGNED16 *waitTime*)

coSleepModeStart - start sleep mode

This function starts the sleep mode. First a timer is started, then the CAN traffic is stopped and the CPU is going to sleep.

Each step is indicated by the function registered by coEventRegister_SLEEP().

Returns

none

Parameters

waitTime	wait time before stop CAN in ms

5.52.4.5 EXTERN_DECL void coSleepRequestSleep (void)

coSleepRequestSleep - request sleep mode to master

Request sleep mode from master by sending sleep request.

Returns

none

5.52.4.6 EXTERN_DECL void coSleepWakeUp (BOOL_T *master*, UNSIGNED8 *status*, UNSIGNED8 *reason*, UNSIGNED16 *repeatTime*)

coSleepWakeUp - awake from sleep

This function can be called to send the wake up message independ form the actual sleep state. It transmit the wake up message, repeat it after the given repeatTime and reinitializes the communication handling.

Returns

none

Parameters

master	wake up master
status	wake up state (only for slaves)
reason	wake up reason
repeatTime	time interval for repeat wake up message

5.53 co_srd.c File Reference

Service Request Device (SDO Manager Slave)

Functions

- RET_T coSrdRequestRegister (CO_SRD_REQ_TYPE_T reqType, UNSIGNED8 sdoClientChannel, UNSI↩ GNED32 timeOut)

  coSrdRegister - register SRD at SDO manager
- RET_T coSrdRequestConnection (UNSIGNED8 sdoClientChannel, UNSIGNED8 remoteNodeId, UNSIGN↩ ED32 timeOut)

  coSrdRequestConnection - request connection to remote node
- RET_T coSrdReleaseConnection (UNSIGNED8 sdoClientChannel, UNSIGNED8 remoteNodeId, UNSIGN↩ ED32 timeOut)

  coSrdReleaseConnection - release connection to remote node
- RET_T coEventRegister_SRD (CO_EVENT_SRD_T pFunction)

  coEventRegister_SRD - register SRD event

- void icoSrdVarInit (void)
- void icoSrdReset (void)

    icoSrdReset
- RET_T coSrdInit (void)

    coInitSrd - init Srd functionality

5.53.1 Detailed Description

Service Request Device (SDO Manager Slave)

contains routines for SRD slave handling

5.53.2 Function Documentation

5.53.2.1 RET_T coEventRegister_SRD (CO_EVENT_SRD_T *pFunction*)

coEventRegister_SRD - register SRD event

register indication function for SRD events

Returns

RET_T

Parameters

pFunction	pointer to function

5.53.2.2 RET_T coSrdInit (void)

coInitSrd - init Srd functionality

Returns

RET_T

5.53.2.3 RET_T coSrdReleaseConnection (UNSIGNED8 *sdoClientChannel,* UNSIGNED8 *remoteNodeId,* UNSIGNED32 *timeOut*)

coSrdReleaseConnection - release connection to remote node

release SDO connection to remote node

If sdoClientChannel = 0, release all connections If remoteNodeId = 0 deregister at sdo manager

The answer will be done by calling function registered coEventRegister_SRD()

Parameters

sdoClientChannel	sdo client channel to node
remoteNodeId	node id of remote node
timeOut	time out until service is aborted

5.53.2.4 RET_T coSrdRequestConnection (UNSIGNED8 *sdoClientChannel,* UNSIGNED8 *remoteNodeId,* UNSIGNED32 *timeOut*)

coSrdRequestConnection - request connection to remote node

Request SDO connection to remote node

The answer will be done by calling function registered coEventRegister_SRD()

Parameters

sdoClientChannel	sdo client channel to node
remoteNodeId	node id of remote node
timeOut	time out until service is aborted

5.53.2.5 RET_T coSrdRequestRegister (CO_SRD_REQ_TYPE_T *reqType,* UNSIGNED8 *sdoClientChannel,* UNSIGNED32 *timeOut*)

coSrdRegister - register SRD at SDO manager

Request register as SRD at the SDO manager

If reqType == CO_SRD_REQ_TYPE_ALL_SDOS sdoClientChannel is ignored If reqType == CO_SRD_REQ_TY↩
PE_NORMAL SDO client channel have to be from 1..128 (0x1280..0x12ff) This channel will be used as SDO client to the SDO manager.

The answer will be done by calling function registered by coEventRegister_SRD()

Parameters

reqType	request type
sdoClientChannel	sdo client channel to SDO Manager
timeOut	time out until service is aborted in msec

5.53.2.6 void icoSrdReset (void)

icoSrdReset

Returns

none

5.53.2.7 void icoSrdVarInit (void)

Returns

none

5.54 co_srd.h File Reference

defines for srd services

Typedefs

- typedef void(∗ CO_EVENT_SRD_T) (CO_SRD_RESULT_T result, UNSIGNED8 errorCode)

    function pointer to srd result function

Enumerations

Functions

- EXTERN_DECL RET_T coSrdRequestRegister (CO_SRD_REQ_TYPE_T reqType, UNSIGNED8 sdo↩ClientChannel, UNSIGNED32 timeOut)

    coSrdRegister - register SRD at SDO manager
- EXTERN_DECL RET_T coSrdRequestConnection (UNSIGNED8 sdoClientChannel, UNSIGNED8 remote↩NodeId, UNSIGNED32 timeOut)

    coSrdRequestConnection - request connection to remote node
- EXTERN_DECL RET_T coSrdReleaseConnection (UNSIGNED8 sdoClientChannel, UNSIGNED8 remote↩NodeId, UNSIGNED32 timeOut)

    coSrdReleaseConnection - release connection to remote node
- EXTERN_DECL RET_T coEventRegister_SRD (CO_EVENT_SRD_T pFunction)

    coEventRegister_SRD - register SRD event
- EXTERN_DECL RET_T coSrdInit (void)

    coInitSrd - init Srd functionality

5.54.1 Detailed Description

defines for srd services

- contains defines for srd services

5.54.2 Typedef Documentation

5.54.2.1 typedef void(∗ CO_EVENT_SRD_T) (CO_SRD_RESULT_T result, UNSIGNED8 errorCode)

function pointer to srd result function

Parameters

result	- result status of action
errorcode	- errorcode if

Returns

void

5.54.3 Enumeration Type Documentation

5.54.3.1 enum CO_SRD_REQ_TYPE_T

request type for SDO register

Enumerator

CO_SRD_REQ_TYPE_ALL_SDOS	request all default Server SDOs
CO_SRD_REQ_TYPE_NORMAL	request one SDO connection

5.54.3.2 enum CO_SRD_RESULT_T

result values for indication function

Enumerator

CO_SRD_RESULT_SUCCESS	requested service ok
CO_SRD_RESULT_TIMEOUT	time out occured, fct aborted
CO_SRD_RESULT_ERROR	error
CO_SRD_RESULT_ALL_REQUEST_SUCCESS	request all sdos ok
CO_SRD_RESULT_NODE_REQUEST_SUCCESS	request connection ok

5.54.4 Function Documentation

5.54.4.1 EXTERN_DECL RET_T coEventRegister_SRD (CO_EVENT_SRD_T *pFunction*)

coEventRegister_SRD - register SRD event

register indication function for SRD events

Returns

RET_T

Parameters

pFunction	pointer to function

5.54.4.2 EXTERN_DECL RET_T coSrdInit (void)

coInitSrd - init Srd functionality

Returns

RET_T

5.54.4.3 EXTERN_DECL RET_T coSrdReleaseConnection (UNSIGNED8 *sdoClientChannel*, UNSIGNED8 *remoteNodeId*, UNSIGNED32 *timeOut*)

coSrdReleaseConnection - release connection to remote node

release SDO connection to remote node

If sdoClientChannel = 0, release all connections If remoteNodeId = 0 deregister at sdo manager

The answer will be done by calling function registered coEventRegister_SRD()

Parameters

sdoClientChannel	sdo client channel to node
remoteNodeId	node id of remote node
timeOut	time out until service is aborted

5.54.4.4 EXTERN_DECL RET_T coSrdRequestConnection (UNSIGNED8 *sdoClientChannel*, UNSIGNED8 *remoteNodeId*, UNSIGNED32 *timeOut*)

coSrdRequestConnection - request connection to remote node

Request SDO connection to remote node

The answer will be done by calling function registered coEventRegister_SRD()

Parameters

sdoClientChannel	sdo client channel to node
remoteNodeId	node id of remote node
timeOut	time out until service is aborted

5.54.4.5 EXTERN_DECL RET_T coSrdRequestRegister (CO_SRD_REQ_TYPE_T *reqType*, UNSIGNED8 *sdoClientChannel*, UNSIGNED32 *timeOut*)

coSrdRegister - register SRD at SDO manager

Request register as SRD at the SDO manager

If reqType == CO_SRD_REQ_TYPE_ALL_SDOS sdoClientChannel is ignored If reqType == CO_SRD_REQ_TY↩
PE_NORMAL SDO client channel have to be from 1..128 (0x1280..0x12ff) This channel will be used as SDO client to the SDO manager.

The answer will be done by calling function registered by coEventRegister_SRD()

Parameters

reqType	request type
sdoClientChannel	sdo client channel to SDO Manager
timeOut	time out until service is aborted in msec

5.55 co_srdo.c File Reference

srdo handling

5.55.1 Detailed Description

srdo handling

contains srdo services

5.56 co_srdo.h File Reference

defines for srdo services

5.56.1 Detailed Description

defines for srdo services

- contains defines for srdo services

5.57 co_stackinit.c File Reference

Functions for stack intialization handling.

Functions

- void coCanOpenStackVarInit (CO_SERVICE_INIT_VAL_T ∗pServiceInitVals)
    coCanOpenStackVarInit - init of variables of the stack

5.57.1 Detailed Description

Functions for stack intialization handling.

contains functions for initialization handling

5.57.2 Function Documentation

5.57.2.1 void coCanOpenStackVarInit (CO_SERVICE_INIT_VAL_T ∗ *pServiceInitVals*)

coCanOpenStackVarInit - init of variables of the stack

This function initializes all global and local variables of the stack.

It can also be used to reinitialize the stack.

Returns

nothing

Parameters

pServiceInitVals	pointer to init vals

5.58 co_store.c File Reference

Stroe/Restore functionality.

5.58.1 Detailed Description

Stroe/Restore functionality.

contains routines for handling store/restore OD data

5.59 co_store.h File Reference

defines for store services

Macros

- #define CO_STORE_AREA_ALL 1u
- #define CO_STORE_SIGNATURE_SAVE 0x65766173ul
- #define CO_STORE_SIGNATURE_LOAD 0x64616f6cul

Typedefs

- typedef RET_T(∗ CO_EVENT_STORE_T) (UNSIGNED8 subIndex)
  function pointer to save/load/clear function

5.59.1 Detailed Description

defines for store services

- contains defines for store services

5.59.2 Macro Definition Documentation

5.59.2.1 #define CO_STORE_AREA_ALL 1u

define for store/load/restore area all

5.59.2.2 #define CO_STORE_SIGNATURE_LOAD 0x64616f6cul

define for load command

5.59.2.3 #define CO_STORE_SIGNATURE_SAVE 0x65766173ul

define for save command

5.59.3 Typedef Documentation

5.59.3.1 typedef RET_T(∗ CO_EVENT_STORE_T) (UNSIGNED8 subIndex)

function pointer to save/load/clear function

Parameters

subIndex	- subindex parameter to point parameter area

Returns

none

5.60 co_sync.c File Reference

sync handling

Functions

- RET_T coEventRegister_SYNC (CO_EVENT_SYNC_T pFunction)

    coEventRegister_SYNC - register SYNC event
- RET_T coEventRegister_SYNC_FINISHED (CO_EVENT_SYNC_FINISHED_T pFunction)

    coEventRegister_SYNC_FINISHED - register SYNC finished event
- RET_T coSyncInit (UNSIGNED32 cobId)

    coSyncInit - init sync functionality

5.60.1 Detailed Description

sync handling

contains SYNC services

5.60.2 Function Documentation

5.60.2.1 RET_T coEventRegister_SYNC (CO_EVENT_SYNC_T *pFunction*)

coEventRegister_SYNC - register SYNC event

This function registers an indication function for SYNC events.

It is called every time a sync message was received or generated before PDOs are handled.

Returns

RET_T

Parameters

pFunction	pointer to function

5.60.2.2 RET_T coEventRegister_SYNC_FINISHED (CO_EVENT_SYNC_FINISHED_T *pFunction*)

coEventRegister_SYNC_FINISHED - register SYNC finished event

This function registers an indication function for finished SYNC handling.

It is called every time a sync message was received or generated and PDO handling is completed.

Returns

RET_T

Parameters

pFunction	pointer to function

5.60.2.3 RET_T coSyncInit (UNSIGNED32 *cobId*)

coSyncInit - init sync functionality

This function initializes the SYNC functionality.

If the node is a sync producer or a sync consumer depends on the value of the object dictionary index 0x1005. Sync counter value can also be set/reset by the value at the object dictionary at index 0x1019

Returns

RET_T

Parameters

cob↩ Id	sync cob-id

5.61 co_sync.h File Reference

defines for sync services

Typedefs

- typedef void(∗ CO_EVENT_SYNC_T) (UNSIGNED8)

  function pointer to SYNC indication
- typedef void(∗ CO_EVENT_SYNC_FINISHED_T) (UNSIGNED8)

  function pointer to SYNC Finished indication

Functions

- EXTERN_DECL RET_T coSyncInit (UNSIGNED32 cobId)

  coSyncInit - init sync functionality
- EXTERN_DECL RET_T coEventRegister_SYNC (CO_EVENT_SYNC_T pFunction)

  coEventRegister_SYNC - register SYNC event
- EXTERN_DECL RET_T coEventRegister_SYNC_FINISHED (CO_EVENT_SYNC_FINISHED_T pFunction)

  coEventRegister_SYNC_FINISHED - register SYNC finished event

5.61.1 Detailed Description

defines for sync services

- contains defines for sync services

5.61.2 Typedef Documentation

5.61.2.1 typedef void(∗ CO_EVENT_SYNC_FINISHED_T) (UNSIGNED8)

function pointer to SYNC Finished indication

Parameters

syncCounter	- actual SYNC counter

Returns

void

5.61.2.2 typedef void(∗ CO_EVENT_SYNC_T) (UNSIGNED8)

function pointer to SYNC indication

Parameters

syncCounter	- actual SYNC counter

Returns

void

5.61.3 Function Documentation

5.61.3.1 EXTERN_DECL RET_T coEventRegister_SYNC (CO_EVENT_SYNC_T *pFunction*)

coEventRegister_SYNC - register SYNC event

This function registers an indication function for SYNC events.

It is called every time a sync message was received or generated before PDOs are handled.

Returns

RET_T

Parameters

pFunction	pointer to function

5.61.3.2 EXTERN_DECL RET_T coEventRegister_SYNC_FINISHED (CO_EVENT_SYNC_FINISHED_T *pFunction*)

coEventRegister_SYNC_FINISHED - register SYNC finished event

This function registers an indication function for finished SYNC handling.

It is called every time a sync message was received or generated and PDO handling is completed.

Returns

RET_T

Parameters

pFunction	pointer to function

5.61.3.3 EXTERN_DECL RET_T coSyncInit (UNSIGNED32 *cobId*)

coSyncInit - init sync functionality

This function initializes the SYNC functionality.

If the node is a sync producer or a sync consumer depends on the value of the object dictionary index 0x1005. Sync counter value can also be set/reset by the value at the object dictionary at index 0x1019

Returns

RET_T

Parameters

cob↩ Id	sync cob-id

5.62 co_time.c File Reference

time handling

Functions

- RET_T coTimeWriteReq (const CO_TIME_T ∗pTimeData)

  coTimeWriteReq - write time request
- RET_T coEventRegister_TIME (CO_EVENT_TIME_T pFunction)

  coEventRegister_TIME - register TIME event
- RET_T coTimeInit (BOOL_T producer, BOOL_T consumer)

  coTimeInit - init time functionality

5.62.1 Detailed Description

time handling

contains TIME services

5.62.2 Function Documentation

5.62.2.1 RET_T coEventRegister_TIME (CO_EVENT_TIME_T *pFunction*)

coEventRegister_TIME - register TIME event

This function registers an indication function for TIME events.

Returns

RET_T

Parameters

pFunction	pointer to function

5.62.2.2 RET_T coTimeInit (BOOL_T *producer*, BOOL_T *consumer*)

coTimeInit - init time functionality

This function initializes the TIME functionality. The parameters give the possibilities to be producer and/or consumer, independ on the current value of the entry in the object dictionary.

Returns

RET_T

Parameters

producer	node can be time producer
consumer	node can be time consumer

5.62.2.3 RET_T coTimeWriteReq (const CO_TIME_T ∗ *pTimeData*)

coTimeWriteReq - write time request

This function sends a time message to the bus.

Returns

RET_T

Parameters

pTimeData	time data to transmit

5.63 co_time.h File Reference

defines for time services

Data Structures

- struct CO_TIME_T

Typedefs

- typedef void(∗ CO_EVENT_TIME_T) (CO_TIME_T ∗pTime)

  function pointer to time function

Functions

- EXTERN_DECL RET_T coTimeWriteReq (CO_TIME_T const ∗pTimeData)

  coTimeWriteReq - write time request
- EXTERN_DECL RET_T coTimeInit (BOOL_T producer, BOOL_T consumer)

  coTimeInit - init time functionality
- EXTERN_DECL RET_T coEventRegister_TIME (CO_EVENT_TIME_T pFunction)

  coEventRegister_TIME - register TIME event

5.63.1 Detailed Description

defines for time services

- contains defines for time services

5.63.2 Typedef Documentation

5.63.2.1 typedef void(∗ CO_EVENT_TIME_T) (CO_TIME_T ∗pTime)

function pointer to time function

Parameters

pTime	- time of day structure

Returns

void

5.63.3 Function Documentation

5.63.3.1 EXTERN_DECL RET_T coEventRegister_TIME (CO_EVENT_TIME_T *pFunction*)

coEventRegister_TIME - register TIME event

This function registers an indication function for TIME events.

Returns

RET_T

Parameters

pFunction	pointer to function

5.63.3.2 EXTERN_DECL RET_T coTimeInit (BOOL_T *producer,* BOOL_T *consumer*)

coTimeInit - init time functionality

This function initializes the TIME functionality. The parameters give the possibilities to be producer and/or consumer, independ on the current value of the entry in the object dictionary.

Returns

RET_T

Parameters

producer	node can be time producer
consumer	node can be time consumer

5.63.3.3 EXTERN_DECL RET_T coTimeWriteReq (const CO_TIME_T ∗ *pTimeData*)

coTimeWriteReq - write time request

This function sends a time message to the bus.

Returns

RET_T

Parameters

pTimeData	time data to transmit

5.64 co_timer.c File Reference

timer routines

Functions

- RET_T coTimerStart (CO_TIMER_T ∗pTimer, UNSIGNED32 timerTime, CO_TIMER_FCT_T pFct, void ∗p←Data, CO_TIMER_ATTR_T timerAttributes)

  coTimerStart - start a timer
- RET_T coTimerStop (CO_CONST CO_TIMER_T ∗pTimer)

  coTimerStop - stop a timer
- BOOL_T coTimerIsActive (CO_CONST CO_TIMER_T ∗pTimer)

  coTimerIsActive - check if timer is active
- void coTimerAttrChange (CO_TIMER_T ∗pTimer, CO_TIMER_ATTR_T timerAttributes)

  coTimerAttrChange - change timer attribute
- void coTimerTick (void)

  coTimerTick - timer tick elapsed
- void coTimerInit (UNSIGNED32 timerVal)

  coTimerInit - init timer interval

5.64.1 Detailed Description

timer routines

contains timer routines

5.64.2 Function Documentation

5.64.2.1 void coTimerAttrChange (CO_TIMER_T ∗ *pTimer,* CO_TIMER_ATTR_T *timerAttributes*)

coTimerAttrChange - change timer attribute

With this function timer attribute can be change.

Returns

none

Parameters

pTimer	pointer to timerstruct
timerAttributes	timer attributes

5.64.2.2 void coTimerInit (UNSIGNED32 *timerVal*)

coTimerInit - init timer interval

This function initializes the internal timer handling. It does nothing with the hardware timer and initializes only internal variables.
The given timer interval is used to calculate the timer period for timer depending functions started by coTimerStart().

Returns

none

Parameters

timerVal	timer interval in µsec

5.64.2.3 BOOL_T coTimerIsActive (CO_CONST CO_TIMER_T * *pTimer*)

coTimerIsActive - check if timer is active

With this function can be ckecked, if a timer is currently in the timer list.

Returns

BOOL_T

Return values

CO_TRUE	timer is active
CO_FALSE	timer is not active

Parameters

pTimer	pointer to timer struct

5.64.2.4 RET_T coTimerStart (CO_TIMER_T * *pTimer*, UNSIGNED32 *timerTime*, CO_TIMER_FCT_T *pFct*, void * *pData*, CO_TIMER_ATTR_T *timerAttributes*)

coTimerStart - start a timer

This function starts a timer with the given timer interval (in µsec). If the timer is elapsed, the indication function pointed by ptrToFct() with the parameter pData is called.

Single-shot or cyclic timer can be defined using the CO_TIMER_ATTR_T attribute.

Returns

RET_T

Parameters

pTimer	pointer to timerstruct
timerTime	timer time in µsec
pFct	function at timer elapsed
pData	pointer for own data
timerAttributes	timer attributes

5.64.2.5 RET_T coTimerStop (CO_CONST CO_TIMER_T ∗ *pTimer*)

coTimerStop - stop a timer

This function stops the given timer.

Returns

RET_T

Return values

RET_OK	timer successful removed
RET_INVALID_PARAMETER	timer not in timer list

Parameters

pTimer	pointer to timerstruct

5.64.2.6 void coTimerTick (void)

coTimerTick - timer tick elapsed

This function should be called, if the CANopen timer has been elapsed to signal a new timer interval to the stack.

It can be called at interrupt level.

Returns

none

5.65 co_timer.h File Reference

defines for timer

Data Structures

- struct co_timer

Typedefs

- typedef void(∗ CO_TIMER_FCT_T) (void ∗)

  function pointer to Timer indication
- typedef struct co_timer xTimer

Enumerations

Functions

- EXTERN_DECL void coTimerInit (UNSIGNED32 timerVal)

  coTimerInit - init timer interval
- EXTERN_DECL RET_T coTimerStart (CO_TIMER_T ∗pTimer, UNSIGNED32 timerTime, CO_TIMER_FC↩ T_T pFct, void ∗pData, CO_TIMER_ATTR_T timerAttributes)

  coTimerStart - start a timer
- EXTERN_DECL RET_T coTimerStop (CO_CONST CO_TIMER_T ∗pTimer)

  coTimerStop - stop a timer
- EXTERN_DECL BOOL_T coTimerIsActive (CO_CONST CO_TIMER_T ∗pTimer)

  coTimerIsActive - check if timer is active
- EXTERN_DECL void coTimerTick (void)

  coTimerTick - timer tick elapsed
- EXTERN_DECL void coTimerAttrChange (CO_TIMER_T ∗pTimer, CO_TIMER_ATTR_T timerAttributes)

  coTimerAttrChange - change timer attribute

5.65.1 Detailed Description

defines for timer

- contains defines for timer

5.65.2 Typedef Documentation

5.65.2.1 typedef void(∗ CO_TIMER_FCT_T) (void ∗)

function pointer to Timer indication

Parameters

pFct	- pointer to timer up function

Returns

void

5.65.2.2 typedef struct co_timer xTimer

timer structure

5.65.3 Enumeration Type Documentation

5.65.3.1 enum CO_TIMER_ATTR_T

timer attributes

Enumerator

CO_TIMER_ATTR_ROUNDUP	round up given timer value
CO_TIMER_ATTR_ROUNDUP_CYCLIC	round up and start timer again
CO_TIMER_ATTR_ROUNDDOWN	round down given timer value
CO_TIMER_ATTR_ROUNDDOWN_CYCLIC	round down and start timer again

5.65.4 Function Documentation

5.65.4.1 EXTERN_DECL void coTimerAttrChange (CO_TIMER_T $*$ *pTimer,* CO_TIMER_ATTR_T *timerAttributes*)

coTimerAttrChange - change timer attribute

With this function timer attribute can be change.

Returns

none

Parameters

pTimer	pointer to timerstruct
timerAttributes	timer attributes

5.65.4.2 EXTERN_DECL void coTimerInit (UNSIGNED32 *timerVal*)

coTimerInit - init timer interval

This function initializes the internal timer handling. It does nothing with the hardware timer and initializes only internal variables.
The given timer interval is used to calculate the timer period for timer depending functions started by coTimerStart().

Returns

none

Parameters

timerVal	timer interval in µsec

5.65.4.3 EXTERN_DECL BOOL_T coTimerIsActive (CO_CONST CO_TIMER_T ∗ *pTimer*)

coTimerIsActive - check if timer is active

With this function can be ckecked, if a timer is currently in the timer list.

Returns

BOOL_T

Return values

CO_TRUE	timer is active
CO_FALSE	timer is not active

Parameters

pTimer	pointer to timer struct

5.65.4.4 EXTERN_DECL RET_T coTimerStart (CO_TIMER_T ∗ *pTimer*, UNSIGNED32 *timerTime*, CO_TIMER_FCT_T *pFct*, void ∗ *pData*, CO_TIMER_ATTR_T *timerAttributes*)

coTimerStart - start a timer

This function starts a timer with the given timer interval (in µsec). If the timer is elapsed, the indication function pointed by ptrToFct() with the parameter pData is called.

Single-shot or cyclic timer can be defined using the CO_TIMER_ATTR_T attribute.

Returns

RET_T

Parameters

pTimer	pointer to timerstruct
timerTime	timer time in µsec
pFct	function at timer elapsed
pData	pointer for own data
timerAttributes	timer attributes

5.65.4.5 EXTERN_DECL RET_T coTimerStop (CO_CONST CO_TIMER_T ∗ *pTimer*)

coTimerStop - stop a timer

This function stops the given timer.

Returns

RET_T

Return values

RET_OK	timer successful removed
RET_INVALID_PARAMETER	timer not in timer list

Parameters

pTimer	pointer to timerstruct

5.65.4.6 EXTERN_DECL void coTimerTick (void)

coTimerTick - timer tick elapsed

This function should be called, if the CANopen timer has been elapsed to signal a new timer interval to the stack.

It can be called at interrupt level.

Returns

none

5.66 co_usdo.c File Reference

USDO routines.

5.66.1 Detailed Description

USDO routines.

contains usdo server routines

5.67 co_usdoserv.c File Reference

USDO server routines.

5.67.1 Detailed Description

USDO server routines.

contains usdo server routines

5.68 co_user.c File Reference

User CAN functionality.

5.68.1 Detailed Description

User CAN functionality.

Contain functions to send other data over CAN

5.69 co_user.h File Reference

defines for time services

Typedefs

- typedef void(∗ CO_EVENT_USER_T) (CO_CONST UNSIGNED16 msgNr, CO_CONST UNSIGNED8 data↩ Len, CO_CONST UNSIGNED8 ∗pRecData)

    function pointer to user function

5.69.1 Detailed Description

defines for time services

- contains defines for time services

5.69.2 Typedef Documentation

5.69.2.1 typedef void(∗ CO_EVENT_USER_T) (CO_CONST UNSIGNED16 msgNr, CO_CONST UNSIGNED8 dataLen, CO_CONST UNSIGNED8 ∗pRecData)

function pointer to user function

Parameters

msgNr	- message number
dataLen	- received data len
precData	- received data

Returns

void

5.70 codrv_can_generic.c File Reference

generic driver

Macros

- #define POLLING 1

Functions

- RET_T codrvCanInit (UNSIGNED16 bitRate)

    codrvCanInit - init CAN controller
- RET_T codrvCanReInit (UNSIGNED16 bitRate)

    codrvCanReInit - reinit CAN controller
- RET_T codrvCanSetBitRate (UNSIGNED16 bitRate)

    codrvCanSetBitRate - set CAN Bitrate
- RET_T codrvCanEnable (void)

    codrvCanEnable - enable CAN controller
- RET_T codrvCanDisable (void)

    codrvCanDisable - disable CAN controller
- RET_T codrvCanStartTransmission (void)

    codrvCanStartTransmission - start can transmission if not active
- void codrvCanTransmitInterrupt (void)

    codrvCanDriverTransmitInterrupt - can driver transmit interrupt
- void codrvCanReceiveInterrupt (void)

    codrvCanReceiveInterrupt - can driver receive interrupt
- void codrvCanDriverHandler (void)

    codrvCanDriverHandler - can driver handler

5.70.1 Detailed Description

generic driver

- generic driver for basic CAN

    Author

    emtas GmbH

    This module contains a skeleton for a basic can driver. It can be use to implement a new driver for CANopen library of emtas.

The official small API for Filter usage is contained. But it is not required for a basic CAN driver.

5.70.2 Macro Definition Documentation

5.70.2.1 #define POLLING 1

CO_DRV_FILTER This setting activates the filter functionality. But note, you need a lot of filter to use it effectively. For a slave for example for the following services:

- NMT
- SDO Request
- n RPDOs optionally a slave can receive other nodes
- Heart Beat (as consumer)
- EMCY (as consumer) Typical the CAN controller is called FullCAN controller if it has for each filtered out CAN frame Id a own hardware storage (message object).

It can also be a sophisticated CAN receiver, preferred with a hardware FIFO, with a sophisticated acceptance filter mechanism.

You have to set this define in gen_define.h! CO_DRV_GROUP_FILTER The group filter mechanism is a additional feature for the general filter mechanism. The most filter can set an acceptance mask. A often used mask enable a group for all NodeIds of a specific command group, e.g. heartbeat consumer. In this case only one filter is required for 128 message identifiers.

You have to set this define in gen_define.h! POLLING Often used driver internal define, e.g. during the development. In case this define is set, the driver don't use interrupts.

You have to use it driver internal, only. CODRV_DEBUG Often used driver internal define to activate the printf() output for debugging. A completely correct functionality is not ensured, if this define is set. Please deactivate it! DEBUG_SEND_TESTMESSAGE Often used driver internal #define to send a transmit CAN frame during the initialization. For measurement purpose the message ID is 0x555 and the data byte 0x01..0x8. If no other CAN node is connected, the CAN controller will send this frame endless. This can be used to measure the bit time using an oscilloscope. Please deactivate it in production code!

5.70.3 Function Documentation

5.70.3.1 RET_T codrvCanDisable (void)

codrvCanDisable - disable CAN controller

This function disables the CAN controller. The function waits for the CAN controller being disabled. Code calling this function typically expects that after returning the CAN controller is in Init mode.

But note, the time the CAN controller needs to enter the Init mode can be as long as the duration of one CAN frame.

Returns

RET_T

Return values

RET_OK	CAN controller is set to be disabled

5.70.3.2 void codrvCanDriverHandler (void)

codrvCanDriverHandler - can driver handler

This function is cyclically called from the CANopen stack to get the current CAN state (BUS_OFF, PASSIVE, AC↩
TIVE).

If a bus off event has occurred, this function should try to get to bus on again (activate the CAN controller).

Returns

void

5.70.3.3 RET_T codrvCanEnable (void)

codrvCanEnable - enable CAN controller

This function enables the CAN controller. At this point the enable bit is set. Typically the CAN controller requests 11 recessive bits to go in active mode. This will be checked later outside of this function.

Returns

RET_T

Return values

RET_OK	CAN controller, enabled was set

5.70.3.4 RET_T codrvCanInit (UNSIGNED16 *bitRate*)

codrvCanInit - init CAN controller

This function initializes the CAN controller and configures the bitrate. At the end of the function, the CAN controller should be in state disabled.

Returns

RET_T

Return values

RET_OK	initialization was OK

Parameters

bitRate	CAN bitrate

5.70.3.5 void codrvCanReceiveInterrupt (void)

codrvCanReceiveInterrupt - can driver receive interrupt

This function is called, if a new message was received. As first get the pointer to the receive buffer and save the message there. Then set the buffer as filled and inform the lib about new data.

Returns

void

5.70.3.6 RET_T codrvCanReInit (UNSIGNED16 *bitRate*)

codrvCanReInit - reinit CAN controller

This Function reinits the CAN controller after deactivation.

In Filter mode: After this function call all Filter are reset and must be reconfigured!

At the end of the function, the CAN controller should be in state disabled.

Parameters

bitRate	- CANopen bitrate

Returns

RET_T

Parameters

bitRate	CAN bitrate

5.70.3.7 RET_T codrvCanSetBitRate (UNSIGNED16 *bitRate*)

codrvCanSetBitRate - set CAN Bitrate

This function sets the CAN Bitrate to the given value. Changing the Bitrate is only allowed, if the CAN controller is in reset. The state at the start of the function is unknown, so the CAN controller should be switch to state reset.

At the end of the function the CAN controller should be stay in state reset.

Returns

RET_T

Return values

RET_OK	setting of Bitrate was OK

Parameters

bitRate	CAN Bitrate in kbit/s

5.70.3.8 RET_T codrvCanStartTransmission (void)

codrvCanStartTransmission - start can transmission if not active

Transmission of CAN messages should be interrupt driven. If a message was sent, the Transmit Interrupt is called and the next message can be transmitted. To start the transmission of the first message, this function is called from the CANopen stack.

The easiest way to implement this function is to trigger the transmit interrupt, but only of the transmission is not already active.

Returns

RET_T

Return values

RET_OK	start transmission was successful

5.70.3.9 void codrvCanTransmitInterrupt (void)

codrvCanDriverTransmitInterrupt - can driver transmit interrupt

This function is called, after a message was transmitted.

As first, inform stack about message transmission. Get the next message from the transmit buffer, write it to the CAN controller and transmit it.

Returns

void

5.71 codrv_cpu_generic.c File Reference

CPU specific routines.

Functions

- void codrvHardwareInit (void)

  codrvHardwareInit - hardware initialization
- void codrvHardwareCanInit (void)

  codrvInitCanHW - CAN related hardware initialization
- void codrvCanEnableInterrupt (void)

  codrvCanEnableInterrupt - enable the CAN interrupt
- void codrvCanDisableInterrupt (void)

  codrvCanDisableInterrupt - disable the CAN interrupt
- void codrvCanSetTxInterrupt (void)

  codrvCanSetTxInterrupt - set pending bit of the Transmit interrupt
- RET_T codrvTimerSetup (UNSIGNED32 timerInterval)

  codrvTimerSetup - init and configure the hardware Timer
- void codrvTimerISR (void)

  codrvTimerISR - Timer interrupt service routine

5.71.1 Detailed Description

CPU specific routines.

cpu specific routines

This module contains the cpu specific routines for initialization and timer handling.

Author

emtas GmbH

5.71.2 Function Documentation

5.71.2.1 void codrvCanSetTxInterrupt (void)

codrvCanSetTxInterrupt - set pending bit of the Transmit interrupt

This function set the interrupt pending bit. In case of the NVIC enable interrupt and the CAN specific enable TX Interrupt mask the CAN interrupt handler is calling.

5.71.2.2 void codrvHardwareCanInit (void)

codrvInitCanHW - CAN related hardware initialization

Within this function you find the CAN only hardware part. Goal of it is, that you can have your own hardware initialization like codrvHardwareInit(), but you can add our tested CAN initialization.

5.71.2.3 void codrvHardwareInit (void)

codrvHardwareInit - hardware initialization

This function initializes the hardware, incl. clock and CAN hardware.

5.71.2.4 void codrvTimerISR (void)

codrvTimerISR - Timer interrupt service routine

This function is normally called from timer interrupt or from an other system timer. It has to call the timer handling function at the library.

Returns

void

5.71.2.5 RET_T codrvTimerSetup (UNSIGNED32 *timerInterval*)

codrvTimerSetup - init and configure the hardware Timer

This function starts a cyclic hardware timer to provide a timing interval for the CANopen library. Alternativly it can be derived from an other system timer with the timer interval given by the function parameter.

Returns

RET_T

Return values

RET_OK	intialization of the timer was ok

Parameters

timerInterval	timer interval in usec

5.72 codrv_error.c File Reference

error state handling

Functions

- CAN_ERROR_FLAGS_T ∗ codrvCanErrorGetFlags (void)

  codrvCanErrorgetFlags - Reference to the error flags
- void codrvCanErrorInit (void)

  codrvCanErrorInit - init Error variables
- RET_T codrvCanErrorInformStack (void)

5.72.1 Detailed Description

error state handling

5.72.2 Function Documentation

5.72.2.1 CAN_ERROR_FLAGS_T∗ codrvCanErrorGetFlags (void)

codrvCanErrorgetFlags - Reference to the error flags

Return values

pointer	to error flags

5.72.2.2 RET_T codrvCanErrorInformStack (void)

codrvCanErrorInformStack - inform the stack about changes

Call outside of interrupts! Typical call in codrvCanDriverHandler().

Index